SHED: Shapley-Based Automated Dataset Refinement for Instruction Fine-Tuning

Anonymous ACL submission

Abstract

The pre-trained Large Language Models (LLMs) can be adapted for many downstream tasks and tailored to align with human preferences through fine-tuning. Recent studies have discovered that LLMs can achieve desirable performance with only a small amount of high-quality data, suggesting that a large amount of the data in these extensive datasets is redundant or even harmful. Identifying high-quality data from vast datasets to curate small yet effective datasets has emerged as a critical challenge. In this paper, we introduce SHED, an automated dataset refinement framework based on Shapley value for instruction fine-tuning. SHED eliminates the need for human intervention or the use of commercial LLMs. Moreover, the datasets curated through SHED exhibit transferability, indicating they can be reused across different LLMs with consistently high performance. We conduct extensive experiments to evaluate the datasets curated by SHED . The results demonstrate SHED 's superiority over state-of-the-art methods across various tasks and LLMs; notably, datasets comprising only 10% of the original data selected by SHED achieve performance comparable to or surpassing that of the full datasets.

1 Introduction

The development of LLMs marks a major leap in machine learning, transforming how we approach natural language processing (NLP) and Artificial Intelligence (AI) research (Radford et al., 2019). Models such as GPT-3 (Brown et al., 2020) and LLaMA (Touvron et al., 2023) highlight the benefits of pre-training on large and diverse datasets, empowering these LLMs with a wealth of knowledge. Moreover, one of the pivotal strengths of LLMs lies in their adaptability to specific tasks through fine-tuning. Fine-tuning, a process that involves updating models on a task-specific dataset, enables the pre-trained model to acquire task-specific information. Furthermore, it facilitates the alignment of models to more accurately follow human instructions through fine-tuning on a dataset comprised of instructions paired with appropriate responses, which is known as instruction tuning.

However, fine-tuning LLMs also raises challenges. A primary concern is that the noisy data or harmful instances in the fine-tuning dataset can significantly degrade the performance of pre-trained LLMs (Srivastava et al., 2020). Many works have developed large and diverse datasets for fine-tuning purposes. While the datasets for fine-tuning have grown in size, recent research suggests that meticulously curated datasets of high quality, even if smaller in size, can be more effective in harnessing the full potential of LLMs (Gunasekar et al., 2023; Zhou et al., 2023a; Chen et al., 2023a). Indiscriminately increasing the volume of data can lead to ineffective performance improvements and might even deteriorate model performance as a result of introducing noisy and harmful instances. In addition, for instruction tuning, the model has already learned the necessary knowledge in the pre-training stage. The dataset used in the fine-tuning stage merely aims to better align the models to follow human instructions, indicating this process does not necessitate extensive data (Li et al., 2024). Furthermore, fine-tuning LLMs on extensive datasets incurs significant computational costs. The necessity for considerable GPU resources presents a critical challenge (Strubell et al., 2019). Only those researchers and institutions equipped with sufficient computing resources are able to perform such tasks, limiting the broader applications and progress within the LLM community. Consequently, there is a pressing need to design a novel method for curating small and high-quality datasets that enable efficient fine-tuning.

Previous efforts have employed various methods such as curation or generation through manual efforts or commercial LLMs (Gunasekar et al., 2023; Cao et al., 2023), identifying subsets from

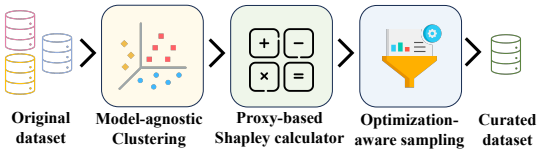


Figure 1: Overview of SHED.

larger datasets via training dynamics or estimating marginal contributions (Swayamdipta et al., 2020; Schoch et al., 2023). Most current methods for data selection neglect the potential influence that different combinations of samples can have on model performance. The Shapley value (Roth, 1988), introduced in cooperative game theory, provides a method for fairly evaluating the contribution of each participant by examining all possible combinations and their effects on the overall result. This principle has also been utilized in machine learning to assess the impact of individual data within a given dataset (Rozemberczki et al., 2022). The Shapley value can serve as criteria to refine one or more large datasets to extract high-quality data points, enabling the curation of a smaller yet high-quality dataset. This method not only facilitates the selection of impactful data, but also considers the effectiveness of selected data combinations. Shapley value seems to be a promising toolkit for data selection. However, calculating the Shapley value for all the data samples in a dataset is computationally expensive, especially for large-scale fine-tuning datasets.

Motivated by the aforementioned challenges, we present SHED , a Shapley-based automated dataset refinement framework for fine-tuning LLMs. The key intuition behind SHED is to perform Shapley value evaluations on a small portion of representative samples only, such that the computation complexity of the Shapley-based data refinement can be dramatically decreased. Specifically, as Figure 1 illustrates, SHED consists of three key components: model-agnostic clustering, proxy-based Shapley calculator, and optimization-aware sampling. Initially, the model-agnostic clustering groups embeddings of the original dataset, and then selects representative data samples as a proxy for each cluster based on the distance of embeddings to the cluster centroid. These proxy data instances are then evaluated by the proxy-based Shapley calculator, which employs an approximation method to efficiently calculate their Shapley values, focusing on task-specific objectives (*e.g.,* accuracy and fairness). This method involves iteratively removing groups of instances from the proxy dataset and assessing the performance variation of the model to estimate the collective contribution of these instances, thereby streamlining the computation of Shapley values. The derived Shapley values of these proxy data instances are used as the quality score for their respective clusters. Finally, optimization-aware sampling selects data from clusters to compile a compact yet high-quality dataset, employing strategies that may favor clusters with higher-quality scores. SHED only computes Shapley values for the cluster representatives rather than each data, drastically boosting the efficiency of data refinement. Moreover, SHED offers a unified yet flexible framework, catering to various user needs by providing multiple options within each component. For example, the optimization objective for Shapley value measurement can be tailored to specific tasks (*e.g.,* fairness). Our key contributions can be summarized as follows:

- We present SHED , a generic data refinement framework based on Shapley values, which can curate a small yet high-quality dataset for boosting the efficiency of fine-tuning LLMs.
- We conducted extensive experiments on two benchmark datasets, *i.e.,* MMLU and WizzardLLM, the results demonstrate that fine-tuning LLMs with small datasets curated by SHED yields performance comparable to, or even better than, using the original large datasets. Remarkably, we observed that the datasets curated by SHED exhibit transferability, achieving robust performance across various LLMs models even from different families. This amortizes the computational cost for data selection when applying the curated dataset to fine-tune various LLM models.
- All the code associated with the collection of high-quality datasets curated by SHED will be open-sourced upon the acceptance of this paper, enabling the community to fine-tune LLM models more efficiently.

2 Related Work

2.1 Coreset Selection

Coreset selection plays a critical role in machine learning by targeting the selection of a representative subset from a larger dataset. Various coreset selection methods use unique criteria for choosing samples. Geometry-based approaches focus on the geometric properties of the data points, striving to retain geometrically significant samples that

represent the overall data distribution (Chen et al., 2012; Sener and Savarese, 2018; Sinha et al., 2020; Agarwal et al., 2020). Uncertainty-based methods choose samples based on the uncertainty they present to the model, typically engaging samples that the model finds challenging to classify (Coleman et al., 2019; Toneva et al., 2018; Paul et al., 2021). Decision-boundary-based methods select samples that are close to the decision boundary of the classifier, ensuring that the nuances of the classification boundary are well-represented in the selected subset (Margatina et al., 2021; Ducoffe and Precioso, 2018). Gradient-matching approaches involve selecting a subset that yields similar gradient distributions as the entire dataset when used in training (Killamsetty et al., 2021a; Mirzasoleiman et al., 2020). Bilevel Optimization optimizes the coreset selection in a way that the selected subset maximizes certain performance metrics (Killamsetty et al., 2021b). Submodularity-based approaches consider both diversity and information richness, striving for a balanced representation of the dataset (Iyer et al., 2021).

2.2 Data Selection for Instruction Fine-tuning

Due to the superiority of instruction tuning in enhancing the performance of LLMs, many recent studies focus on selecting high-quality instruction tuning data. Based on methods, it can be divided into the following categories. Indicators-based methods define multiple metrics, such as instruction length and perplexity, to compute quality scores for each instruction instance (Cao et al., 2023; Wei et al., 2023; Zhou et al., 2023b; Du et al., 2023). Training-based methods leverage the performance improvement through fine-tuning to score and select instruction data suited for fine-tuning (Li et al., 2023a,b,c; Wu et al., 2023; Chen et al., 2023b; Kung et al., 2023). Some other methods employ commercial LLMs like ChatGPT to assess quality, complexity, and diversity of instructions for selection (Chen et al., 2023a; Lu et al., 2023; Xu et al., 2023b; Liu et al., 2023; Zhao et al., 2023).

2.3 Limitations of Previous Work

Most existing methods for data selection overlook the impact of various data combinations on model performance. As Table 1 illustrates, datasets formed by combining high-quality data, which are merely based on the independent quality score of each individual data sample, do not necessarily enhance model performance effectively. The combination of different data can impact the final performance of fine-tuning. Furthermore, many works are task-specific, limiting their broader applicability.

Table 1: We apply DSIR (Xie et al., 2023) to compile a high-quality dataset (10k instances), a random dataset (10k instances) from MMLU, and a mixed dataset samples 5k instances from each of the high-quality and random datasets. We fine-tune the LLaMA-7B model (Touvron et al., 2023) on the curated dataset and evaluate them using the MMLU test set.

Dataset	High-quality	Random	Mixed
MMLU	40.04	39.13	40.92

3 Proposed Method

Motivated by the aforementioned challenges, we present SHED, a generic framework that exploits Shapley value to identify and select high-quality data to improve the performance and efficiency of fine-tuning LLMs.

3.1 Preliminary

The motivation behind this work is underscored by the observation, as illustrated in Table 1, that naively aggregating high-quality data merely based on the independent importance of individual samples does not guarantee a performance improvement of fine-tuning. We believe this phenomenon is attributed to the complex interactions between different instances within the fine-tuning process. Thus, there is a pressing need to design a novel data selection method, which accounts for the individual and collective contributions of instances to model performance.

The Shapley value offers a compelling solution to this challenge. It quantifies the marginal contribution of each instance to the overall performance of the model, considering all possible combinations of instances. The formulation of the Shapley value for a data sample i in dataset D can be expressed as:

$$S_i = \sum_{P \in D \setminus \{i\}} \frac{|P|!(|D| - |P| - 1)!}{|D|!} (v(P \cup i) - v(P)), \tag{1}$$

where S_i is the Shapley value of i, P is the subset of dataset D, $|D|$ and $|P|$ are the total number of instances in D and P, $v(P)$ is the value function of P, which represents the performance of the LLM model fine-tuned on the subset P. As Eq. 1 indicates, the Shapley value of an instance i captures

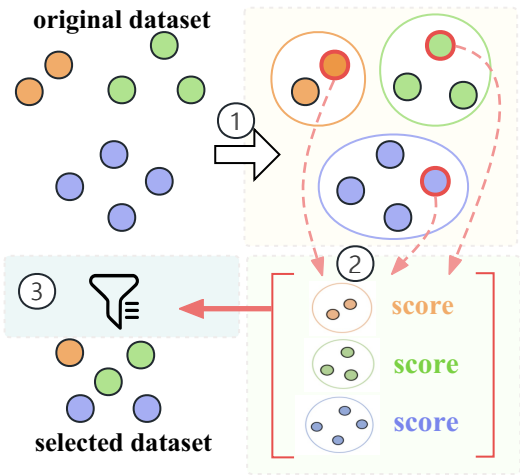


Figure 2: Workflow of SHED: ① Clustering and determining proxy data; ② Calculating Shapley values as scores; and ③ Sampling based on scores.

its average impact on model performance across all subsets it might be part of. This ensures a fair evaluation of the contribution of each instance in the original dataset, enabling the selected data is genuinely beneficial for enhancing model performance when integrated with other data samples.

Additionally, the value function $v(P)$ in Eq. 1 serves to calculate contributions from corresponding data. This value function can be tailored for various optimization objectives, such as accuracy and fairness, facilitating the selection of data that aligns with the task-specific requirements.

However, computing the Shapley value, as depicted in Eq. 1, demands extensive computational efforts, because it requires evaluating the contribution of each instance across all possible combinations. For a dataset with $|D|$ instances, there are a total of $2^{|D|} - 1$ possible combinations. For each combination, two evaluations are needed, *i.e.,* one includes a certain instance and the other one holds out that instance, doubling the computational workload to determine the contribution of that particular instance. Thus, the time complexity for measuring the Shapley value of each instance is $O(2^{|D|})$. Given the need to perform this calculation for all $|D|$ instances to determine their individual Shapley values, the overall time complexity for the dataset increases to $O(|D| \cdot 2^{|D|})$. This exponential complexity makes direct computation of Shapley values impractical for large datasets.

3.2 Design of SHED

To address the above challenges, we design SHED, comprising of three key components: model-agnostic clustering, proxy-based Shapley calculator, and optimization-aware sampling. We introduce each component in detail.

Model-agnostic Clustering. Given the time complexity of computing the Shapley value, calculating the Shapley value for all instances in a large fine-tuning dataset is impractical. The model-agnostic clustering employs models from Sentence Transformers (Reimers and Gurevych, 2019) to generate semantically meaningful embeddings for each sample in the original dataset. These embeddings facilitate the efficient and effective computation of semantic similarities between textual inputs, enabling the grouping of data with similar contexts. Moreover, those model-agnostic embeddings enhance the transferability of the curated dataset, as demonstrated in Table 7. Then, the model-agnostic clustering applies algorithms, such as K-means (Ahmed et al., 2020) and Agglomerative Clustering (Murtagh and Legendre, 2014), to group the embeddings. It then selects the representative data, which is closest to the cluster centroids in the embedding space, for each cluster. In doing so, we use these representative samples as the proxy of the respective clusters. Subsequently, SHED only calculates the Shapley values of those proxy data, using their Shapley values as the quality scores for their respective clusters. Employing proxy data effectively captures the essence of the diversity and complexity in the dataset. This strategy significantly reduces the computational burden associated with calculating Shapley values across vast datasets.

Proxy-based Shapley Calculator. To further improve efficiency for Shapley value calculations, the proxy-based Shapley calculator employs an approximation method to estimate the Shapley values of the proxy data. This method iteratively removes groups of n instances from the proxy data D_p, followed by an evaluation of the model's performance to assess the impact of these instances. The performance variations before and after the removal of a specific group of instances quantify their collective contribution. Specifically, the contribution of the initial group of n instances, denoted as $c_{(1..n)\in D_p}$, is computed by $c_{(1..n)\in D_p} = v(D_p) - v(D_p \setminus \{1..n\})$. Similarly, the contribution for the subsequent group of n instances is determined by $c_{(n+1..2n)\in D_p} = v(D_p \setminus \{1..n\}) - v(D_p \setminus \{1..2n\})$. This procedure is repeated, progressively removing groups of n instances until the entire proxy data

has been visited, which marks the completion of a single iteration. This entire iteration process is then repeated k times to enhance the accuracy of the approximation. After completing k iterations, the Shapley value for a certain instance i of the proxy dataset is approximated using the average of its contributions across all iterations, defined as $S_i \approx \frac{1}{k}\sum_k \frac{c_i(k)}{n}$, where $c_i(k)$ denotes the contribution associated with instance i in the kth iteration.

Optimization-aware Sampling. The Shapley value of each proxy data is assigned as the quality score of the corresponding cluster. Optimization-aware sampling utilizes these quality scores to sample data from these clusters, aiming to curate a small yet high-quality dataset. Optimization-aware Sampling offers two sampling methods: Quality-Ordered Cluster Sampling (QOCS) and Quality-Weighted Cluster Sampling (QWCS). QOCS prioritizes sampling from clusters with the highest quality scores. It selects instances starting from the most high-quality clusters until a predefined target sampling number is reached. QWCS adopts a probabilistic approach to sample instances across all clusters, with the probability of selection from a given cluster weighted by its quality score. This method aims to balance quality with diversity by allowing for the inclusion of instances from a broader array of clusters, thus potentially enriching the dataset with a wider variety of high-quality data points. The probability $\mathrm{Pr}(i)$ of selecting an instance from cluster i is defined in Eq. 2:

$$\mathrm{Pr}(i) = \frac{e^{fS_i}}{\sum_i e^{fS_i}}, \tag{2}$$

where S_i represents the quality score of cluster i, and f is a scaling factor that modulates the emphasis on quality versus diversity within the sampled dataset. By adjusting f, users can tailor the sampling process to prioritize either quality or diversity to suit specific task goals. A higher f value tends towards selecting higher-quality instances, offering a versatile toolkit for dataset optimization.

4 Experiments

4.1 Experimental Setup

Datasets. We conduct experiments on two famous benchmark datasets, MMLU (99.8k instances) (Hendrycks et al., 2021) and WizardLM-evol-instruct-70k (70k instances) (Xu et al., 2023a).

SHED Implementation. We use the K-means algorithm for the model-agnostic clustering and set the number of clusters to 3000. For the proxy-based Shapley calculator, the value function is set as the accuracy of the foundation model fine-tuned on the proxy data. We use LLaMA-7B (Touvron et al., 2023) as the pre-trained foundation model and 10% instances in the MMLU test set calculating the Shapley values of proxy data. The number of iterations k is set to 10, and the number of instances n removed from the proxy data each step is set to 60. To conserve time and resources, instruction fine-tuning within the proxy-based Shapley calculator is conducted for one epoch. For optimization-aware sampling, we employ the QOCS and QWCS strategies with setting the scaling factor to 1, investigating their efficacy with a variety of target sampling sizes. These implementations are denoted as **SHED-QOCS** and **SHED-QWCS**. The target sampling size varies from $1,000$ to $20,000$ with increments of $1,000$, to thoroughly assess the impact of each sampling approach on fine-tuning performance.

Baseline Methods. We compare SHED with three baseline methods. Specifically, we implement a random-sampling method, denoted as **RS**, which randomly selects a subset from a large dataset. We also use the Dataset Quantization method (Zhou et al., 2023b), denoted by **DQ**, and the Data Selection with Importance Resampling (Xie et al., 2023), denoted by **DSIR**, for comparisons. In addition, we also consider fine-tuning models on the entire dataset, denoted as **FULL**, as a baseline.

Evaluation Settings. After obtaining the curated datasets using SHED and baseline methods, we fine-tune the pre-trained models using each curated subset, respectively. We apply the Low-Rank Adaptation (LoRA) for fine-tuning and set the default LoRA rank to 128. For all curated datasets, the instruction fine-tuning was conducted for 3 epochs. We evaluate the performance of fine-tuned models on MMLU and ARC-challenge tasks using the `lm-evaluation-harness` testing framework (Gao et al., 2023). To better evaluate the human preferences of fine-tuned models, we adopt `MT-Bench` (Zheng et al., 2023) in our experiments. All the experiments are conducted on two A100 GPUs.

4.2 Experiment Results

We summarize the experimental results for SHED and other baseline methods. For consistency, the **bold** numbers indicate the corresponding method outperforms the **FULL** method. Additionally, we

Table 2: Performance comparison of curated datasets of the same size by SHED and baseline methods.

Original dataset	MMLU					WizardLM				
Method	RS	DQ	DSIR	QOCS	QWCS	RS	DQ	DSIR	QOCS	QWCS
MMLU	38.94	39.88	40.24	44.80	43.87	33.12	**33.20**	**33.86**	**35.43**	**34.91**
ARC-challenge	45.10	**46.35**	45.67	**47.10**	**47.23**	46.01	**48.71**	47.66	**49.47**	**49.92**

Table 3: Performance of the best-selected datasets of SHED and baseline methods on the MMLU task.

	MMLU	WizardLM
QOCS	44.80 (10k)	**35.92 (4k)**
QWCS	44.24 (13k)	**35.76 (9k)**
RS	40.87 (15k)	**34.33 (7k)**
DQ	43.50 (7k)	**33.97 (7k)**
DSIR	40.23 (13k)	**34.72 (10k)**
Full	45.56 (99.8k)	33.16 (70k)

Table 4: Performance of the best-selected datasets of SHED and baselines on the ARC-challenge task.

	MMLU	WizardLM
QOCS	**47.10 (10k)**	**51.36 (1k)**
QWCS	**49.21 (9k)**	**50.26 (7k)**
RS	**47.07 (13k)**	**49.33 (16k)**
DQ	**46.50 (3k)**	**50.24 (5k)**
DSIR	**46.90 (3k)**	**48.78 (12k)**
Full	45.99 (99.8k)	47.95 (70k)

underline the best result achieved among all the methods that curate subsets.

For each method, the dataset from the curated collections that yields the optimal result across various sample sizes is referred to as the **best-selected dataset**.

Effectiveness of SHED. Given the datasets generated from SHED and the baseline methods, we fine-tune the LLaMA-7B model, respectively, and evaluate the fine-tuned models on the MMLU and ARC-challenge tasks. We compare the results of the datasets of 10k instances curated by SHED and the baseline methods. As depicted in Table 2, when the number of total sampling instances is fixed (10k), the datasets curated by SHED consistently outperform those chosen by baseline methods. We also compare the performance of fine-tuned models using the best-selected dataset by each method. Table 3 shows the evaluation results for the MMLU task. Our method, **SHED-QOCS**, demonstrated superior performance on the MMLU dataset compared to baseline methods, achieving the highest results among the curated datasets. Furthermore, **SHED-QOCS** also led in performance when utilizing the WizardLM dataset. It is notable that **SHED-QOCS** outperforms the full dataset, achieving a 2.76% higher accuracy. In Table 4, we report the results of the ARC-challenge task. Similarly, among the datasets curated from the MMLU dataset, the selected dataset of our method **SHED-QWCS** achieves the best result compared with the baseline methods. It also surpasses the full dataset by 3.22%. Within the datasets derived from WizardLM, **SHED-QOCS** once again curated the dataset of best performance, which surpasses the full dataset by 3.41%. The results demonstrate the effectiveness of SHED. Although SHED demands more computational effort, its strength lies in creating high-performance datasets.

Evaluations on MT-Bench. We also use MT-Bench to evaluate the performance of datasets curated by SHED in terms of human preferences. Table 5 demonstrates that the dataset curated by SHED aligns well with human preferences, not only enhancing accuracy but also enabling the model to better understand and follow human instructions, generating answers that are more favorable to humans. The dataset constructed through the SHED-QWCS method, sampled from WizardLM, achieved a remarkable score of 5.24 on the MT-Bench.

Transferability Evaluation of Curated Datasets across Various Models. To evaluate the transferability of datasets curated by SHED , we first apply SHED to select data from the MMLU and WizardLM datasets based on LLaMA-7B. Then, we fine-tune LLaMA-13B, Vicuna-7B, and GPT-2 using the best-selected dataset curated by SHED and the baseline methods. As summarized in Table 6 and Table 7, datasets curated by SHED exhibit robust performance across various models, demonstrating their transferability and applicability across various tasks and even different model families. The strong transferability of the curated datasets indicates that SHED identifies generally high-quality data. In addition, the computational cost for data selection can be significantly amor-

Table 5: MT-Bench evaluation of the best-selected datasets of SHED and baselines.

Original dataset	MMLU					WizardLM				
Method	Full	RS	QOCS	RS	QWCS	Full	RS	QOCS	RS	QWCS
Size	99.8k	10k	10k	13k	13k	70k	4k	4k	9k	9k
LLaMA-7B	3.02	2.23	2.53	2.44	2.83	5.21	4.77	4.89	4.81	**5.24**

Table 6: Transferability evaluation using the best-selected datasets across different models on MMLU task.

Original dataset	MMLU					WizardLM				
Method	Full	RS	QOCS	RS	QWCS	Full	RS	QOCS	RS	QWCS
Size	99.8k	10k	10k	13k	13k	70k	4k	4k	9k	9k
LLaMA-13B	53.22	50.04	52.95	50.12	51.54	45.63	45.77	**45.93**	45.81	**46.36**
VICUNA-7B	49.70	48.43	**50.01**	47.21	48.93	45.56	45.71	**47.19**	45.44	**48.16**
GPT-2	24.22	23.74	**26.89**	**24.33**	**25.83**	26.19	25.07	**26.76**	24.85	25.77

tized across various models.

Impact of Number of Clusters. The number of clusters in K-means affects the computational cost needed for Shapley value calculations and the relevance of proxy data to its cluster. An increase in the number of clusters leads to smaller and more homogeneous groups, thereby improving the proxy data's representativeness for its respective clusters. However, this comes at the cost of increased computational overhead, highlighting a balance that must be struck to optimize both efficiency and representativeness. In this experiment, we evaluate the best-selected dataset by SHED across varying numbers of clusters using LLaMA-7B on the MMLU test set. Guided by the findings in (Zhou et al., 2010), our investigation begins with a baseline cluster count of $C = \sqrt{|D|}$. We present the computation time for Shapley value computations across different settings, maintaining consistency with the experimental setup outlined in Section 4.1.

As Figure 3(a) and Figure 3(b) show, the results reveal that performance improvements of curated dataset reach a plateau when the number of clusters exceeds $3\sqrt{|D|}$. Meanwhile, Figure 3(c) demonstrates a proportional increase in computation time for Shapley value calculations as the number of clusters rises. Notably, at very low cluster counts (e.g., below 1000), Shapley value computation times are largely dictated by the evaluation, with the time spent remaining relatively constant across varying datasets. In such cases, the computation time is more significantly affected by the size of pre-trained models rather than the number of clusters itself. Given the transferability of datasets curated using the SHED, it is feasible to employ a smaller foundational model than the target model within the proxy-based Shapley value calculator. In doing so, the computation overhead for evaluation can be significantly reduced, making SHED a practical approach in real-world settings.

Impact of Number of Iterations on Proxy-based Shapley Calculator. The precision of Shapley value estimates increases with the number of iterations k, providing a more accurate measurement of each data sample's contribution to the model performance. However, this increment also leads to a proportional rise in computational cost, leading to a contrasting relationship between computational efficiency and the accuracy of Shapley value estimations.

To seek the optimal number of iterations for Shapley value calculations, we analyzed the performance of datasets curated by SHED under varying iteration settings. The experiments are conducted with the LLaMA-7B model on the MMLU test set, following the experimental settings detailed in Section 4.1.

Figures 4(a) and 4(b) illustrate that the performance of the curated datasets by **QOCS** and **QWCS** are stable once the iteration number surpasses 10. This result highlights the stability of our methods beyond 10 iterations, showing that further iterations beyond this threshold do not significantly improve dataset quality. Given the balance between computational cost and performance, setting the number of iterations to 10 is recommended for optimal efficiency and robustness.

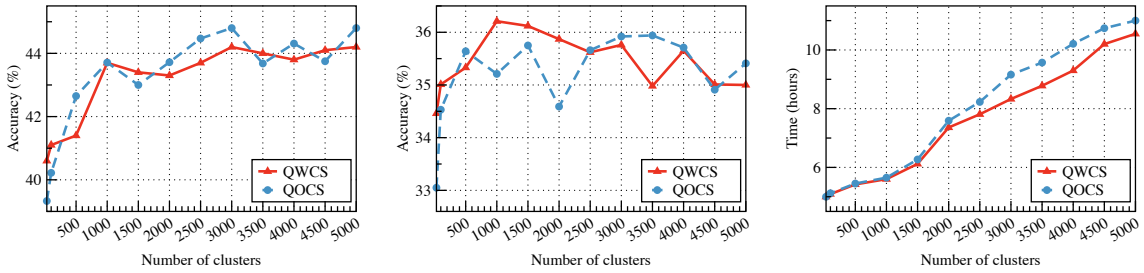
5 Discussion

5.1 Data Selection for Multiple Tasks.

In our experiments, we thoroughly evaluate methods regarding accuracy. It is notable that our frame-

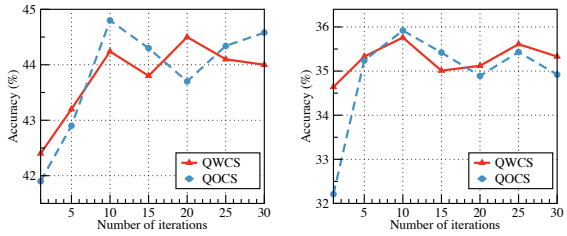
Table 7: Transferability evaluation using the best-selected datasets across different models on ARC-challenge task.

Original dataset	MMLU					WizardLM				
Method	Full	RS	QOCS	RS	QWCS	Full	RS	QOCS	RS	QWCS
Size	99.8k	10k	10k	13k	13k	70k	4k	4k	9k	9k
LLaMA-13B	49.31	47.31	**50.43**	48.83	**50.68**	54.09	53.17	**55.20**	54.11	**55.63**
VICUNA-7B	44.88	44.86	**45.23**	43.24	**44.91**	49.91	47.72	**50.26**	47.98	48.72
GPT-2	19.45	18.77	**19.81**	19.02	**20.05**	19.19	17.98	**19.28**	18.72	**19.54**



(a) Subsets selected from MMLU. (b) Subsets selected from WizardLM. (c) Computational time for one iteration of Shapley value calculation.

Figure 3: Performance of subsets with varying numbers of clusters in SHED.



(a) Subsets selected from MMLU. (b) Subsets selected from WizardLM.

Figure 4: Performance of subsets with varying iterations in SHED.

work is readily adaptable. By setting different value functions $v(P)$, SHED can select any subset using arbitrary criteria. This adaptability allows SHED to customize its data selection process to produce a small dataset while improving specific objectives, such as model fairness (Huang et al., 2019).

In particular, if we aim to curate a dataset using the common fairness notion, *i.e.,* demographic parity, we can define $v(P)$ the disparity in positive prediction rates between groups with protected attributes (e.g., males vs. females), calculated as the negative absolute difference $-|X_{\text{Male}} - X_{\text{Female}}|$, where X_{Male} and X_{Female} are the positive prediction rates for male and female groups, respectively.

5.2 Complexity Analysis

We assume that the running time required to fine-tune the model using a single instance is denoted by t, and the time needed to evaluate the model on a test set consisting of m instances is represented by T_m. Let C denote the number of clusters, n denote the number of instances within a group and k signifies the number of iterations utilized in the proxy-based Shapley calculator as illustrated in Section 3.2. The total number of evaluations and fine-tuning per iteration would be proportional to $\frac{C}{n}$. For simplicity, we assume that C is evenly divisible by n for simplicity. Given k iterations, the overall time complexity of this approximation method can be expressed as $\mathcal{O}\left(\frac{Ck}{n}\left[\frac{(C+n)t}{2} + T_m\right]\right)$.

6 Conclusion

In this work, we introduced SHED, an innovative Shapley value-based framework designed to refine datasets for the efficient fine-tuning of LLMs, addressing the computational hurdles commonly associated with Shapley value calculations through a novel clustering and proxy-based approach. Through extensive experiments conducted on benchmark datasets such as MMLU and WizardLLM, we have shown that LLMs fine-tuned with datasets curated by SHEDnot only match but, in some cases, surpass the performance of those trained with the original, larger datasets. Significantly, SHED-curated datasets have demonstrated a high degree of transferability, maintaining robust performance across various models. Furthermore, SHED's flexibility and efficiency underscore its potential to revolutionize LLM fine-tuning by allowing for the creation of compact, high-quality datasets.

7 Limitations

This research, while presenting significant advancements, encounters certain limitations that merit attention for future work. Firstly, the proposed method's reliance on clustering and proxy data may inadvertently diminish data diversity, potentially overlooking rare but critical samples. To address this, future research will investigate the adoption of novel clustering algorithms specifically designed to enhance the visibility and inclusion of rare samples. Secondly, the framework's current objective focuses predominantly on model performance, which may inadvertently lead to model bias. This singular focus overlooks the equally important aspect of model fairness, which is crucial for ensuring that the models perform equitably across diverse groups. Recognizing this, our framework is designed to be extensible and objective-agnostic, laying the groundwork for incorporating additional criteria. In subsequent research, we plan to integrate considerations of model fairness alongside performance.

8 Ethics Statement

In this work, we present SHED, a generic data refinement framework for data refinement utilizing Shapley values, aimed at assembling a compact yet effective dataset to boost the efficiency of the fine-tuning process of LLMs. This study carefully avoids ethical issues beyond standard AI concerns, leveraging properly cited publicly available Internet text data. This approach ensures adherence to ethical data use standards, reflecting our commitment to responsible research practices in the AI field.

References

Sharat Agarwal, Himanshu Arora, Saket Anand, and Chetan Arora. 2020. Contextual diversity for active learning. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVI 16*, pages 137–153. Springer.

Mohiuddin Ahmed, Raihan Seraj, and Syed Mohammed Shamsul Islam. 2020. The k-means algorithm: A comprehensive survey and performance evaluation. *Electronics*, 9(8):1295.

Paul T Boggs and Jon W Tolle. 1995. Sequential quadratic programming. *Acta numerica*, 4:1–51.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Yihan Cao, Yanbin Kang, Chi Wang, and Lichao Sun. 2023. Instruction mining: When data mining meets large language model finetuning.

Lichang Chen, Shiyang Li, Jun Yan, Hai Wang, Kalpa Gunaratna, Vikas Yadav, Zheng Tang, Vijay Srinivasan, Tianyi Zhou, Heng Huang, and Hongxia Jin. 2023a. Alpagasus: Training a better alpaca with fewer data.

Yongrui Chen, Haiyun Jiang, Xinting Huang, Shuming Shi, and Guilin Qi. 2023b. Tegit: Generating high-quality instruction-tuning data with text-grounded task design. *arXiv preprint arXiv:2309.05447*.

Yutian Chen, Max Welling, and Alex Smola. 2012. Super-samples from kernel herding.

Cody Coleman, Christopher Yeh, Stephen Mussmann, Baharan Mirzasoleiman, Peter Bailis, Percy Liang, Jure Leskovec, and Matei Zaharia. 2019. Selection via proxy: Efficient data selection for deep learning. *arXiv preprint arXiv:1906.11829*.

Qianlong Du, Chengqing Zong, and Jiajun Zhang. 2023. Mods: Model-oriented data selection for instruction tuning. *arXiv preprint arXiv:2311.15653*.

Melanie Ducoffe and Frederic Precioso. 2018. Adversarial active learning for deep networks: a margin based approach. *arXiv preprint arXiv:1802.09841*.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2023. A framework for few-shot language model evaluation.

Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, Adil Salim, Shital Shah, Harkirat Singh Behl, Xin Wang, Sébastien Bubeck, Ronen Eldan, Adam Tauman Kalai, Yin Tat Lee, and Yuanzhi Li. 2023. Textbooks are all you need.

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)*.

Po-Sen Huang, Huan Zhang, Ray Jiang, Robert Stanforth, Johannes Welbl, Jack Rae, Vishal Maini, Dani Yogatama, and Pushmeet Kohli. 2019. Reducing sentiment bias in language models via counterfactual evaluation. *arXiv preprint arXiv:1911.03064*.

Rishabh Iyer, Ninad Khargoankar, Jeff Bilmes, and Himanshu Asanani. 2021. Submodular combinatorial information measures with applications in machine learning. In *Algorithmic Learning Theory*, pages 722–754. PMLR.

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

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

Po-Nien Kung, Fan Yin, Di Wu, Kai-Wei Chang, and Nanyun Peng. 2023. Active instruction tuning: Improving cross-task generalization by training on prompt sensitive tasks. *arXiv preprint arXiv:2311.00288*.

Ming Li, Yong Zhang, Zhitao Li, Jiuhai Chen, Lichang Chen, Ning Cheng, Jianzong Wang, Tianyi Zhou, and Jing Xiao. 2023a. From quantity to quality: Boosting llm performance with self-guided data selection for instruction tuning. *arXiv preprint arXiv:2308.12032*.

Xian Li, Ping Yu, Chunting Zhou, Timo Schick, Luke Zettlemoyer, Omer Levy, Jason Weston, and Mike Lewis. 2023b. Self-alignment with instruction backtranslation. *arXiv preprint arXiv:2308.06259*.

Yunshui Li, Binyuan Hui, Xiaobo Xia, Jiaxi Yang, Min Yang, Lei Zhang, Shuzheng Si, Junhao Liu, Tongliang Liu, Fei Huang, and Yongbin Li. 2024. One shot learning as instruction data prospector for large language models.

Yunshui Li, Binyuan Hui, Xiaobo Xia, Jiaxi Yang, Min Yang, Lei Zhang, Shuzheng Si, Junhao Liu, Tongliang Liu, Fei Huang, et al. 2023c. One shot learning as instruction data prospector for large language models. *arXiv preprint arXiv:2312.10302*.

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*.

Keming Lu, Hongyi Yuan, Zheng Yuan, Runji Lin, Junyang Lin, Chuanqi Tan, Chang Zhou, and Jingren Zhou. 2023. # instag: Instruction tagging for analyzing supervised fine-tuning of large language models. *arXiv e-prints*, pages arXiv–2308.

Katerina Margatina, Giorgos Vernikos, Loïc Barrault, and Nikolaos Aletras. 2021. Active learning by acquiring contrastive examples. *arXiv preprint arXiv:2109.03764*.

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

Fionn Murtagh and Pierre Legendre. 2014. Ward's hierarchical agglomerative clustering method: which algorithms implement ward's criterion? *Journal of classification*, 31:274–295.

Mansheej Paul, Surya Ganguli, and Gintare Karolina Dziugaite. 2021. Deep learning on a data diet: Finding important examples early in training. *Advances in Neural Information Processing Systems*, 34:20596–20607.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks.

Alvin E Roth. 1988. *The Shapley value: essays in honor of Lloyd S. Shapley*. Cambridge University Press.

Benedek Rozemberczki, Lauren Watson, Péter Bayer, Hao-Tsung Yang, Olivér Kiss, Sebastian Nilsson, and Rik Sarkar. 2022. The shapley value in machine learning.

Stephanie Schoch, Ritwick Mishra, and Yangfeng Ji. 2023. Data selection for fine-tuning large language models using transferred shapley values.

Ozan Sener and Silvio Savarese. 2018. Active learning for convolutional neural networks: A core-set approach.

Samarth Sinha, Han Zhang, Anirudh Goyal, Yoshua Bengio, Hugo Larochelle, and Augustus Odena. 2020. Small-gan: Speeding up gan training using core-sets. In *International Conference on Machine Learning*, pages 9005–9015. PMLR.

Ankit Srivastava, Piyush Makhija, and Anuj Gupta. 2020. Noisy text data: Achilles' heel of bert. In *Proceedings of the Sixth Workshop on Noisy User-generated Text (W-NUT 2020)*, pages 16–21.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in nlp.

Swabha Swayamdipta, Roy Schwartz, Nicholas Lourie, Yizhong Wang, Hannaneh Hajishirzi, Noah A. Smith, and Yejin Choi. 2020. Dataset cartography: Mapping and diagnosing datasets with training dynamics.

Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J Gordon. 2018. An empirical study of example forgetting during deep neural network learning. *arXiv preprint arXiv:1812.05159*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Lai Wei, Zihao Jiang, Weiran Huang, and Lichao Sun. 2023. Instructiongpt-4: A 200-instruction paradigm for fine-tuning minigpt-4. *arXiv preprint arXiv:2308.12067*.

Shengguang Wu, Keming Lu, Benfeng Xu, Junyang Lin, Qi Su, and Chang Zhou. 2023. Self-evolved diverse data sampling for efficient instruction tuning. *arXiv preprint arXiv:2311.08182*.

Sang Michael Xie, Shibani Santurkar, Tengyu Ma, and Percy Liang. 2023. Data selection for language models via importance resampling.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. 2023a. Wizardlm: Empowering large language models to follow complex instructions.

Yang Xu, Yongqiang Yao, Yufan Huang, Mengnan Qi, Maoquan Wang, Bin Gu, and Neel Sundaresan. 2023b. Rethinking the instruction quality: Lift is what you need.

Yingxiu Zhao, Bowen Yu, Binyuan Hui, Haiyang Yu, Fei Huang, Yongbin Li, and Nevin L Zhang. 2023. A preliminary study of the intrinsic relationship between complexity and alignment. *arXiv preprint arXiv:2308.05696*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. 2023a. Lima: Less is more for alignment.

Daquan Zhou, Kai Wang, Jianyang Gu, Xiangyu Peng, Dongze Lian, Yifan Zhang, Yang You, and Jiashi Feng. 2023b. Dataset quantization.

Shi-Bing Zhou, Zhen-Yuan Xu, and Xu-Qing Tang. 2010. Method for determining optimal number of clusters in k-means clustering algorithm. *Journal of computer applications*, 30(8):1995.

A Automatic Hyperparameter Setting

To enhance user convenience, SHARE introduces an automatic mechanism for setting hyperparameters.

Based on Figure 3(c), the time per iteration when calculating Shapley values exhibits an approximate linear relationship with the number of clusters, and similarly, the total time for computing Shapley values closely aligns linearly with the number of iterations. Thus, the computation time t for Shapley values can be modeled as $t = \theta kC$. SHARE randomly samples 2000 instances from the dataset to calculate the Shapley value for one iteration, recording this to determine θ. To optimize k to be close to 10 and the number of clusters near $3\sqrt{|D|}$, an optimization problem is formulated in Eq. 3.

$$\begin{aligned} \min_{k,C} \quad & \lambda_1(k-10)^2 + \lambda_2(C - 3\sqrt{|D|})^2 \\ \text{s.t.} \quad & \theta kC = t_0 \end{aligned} \tag{3}$$

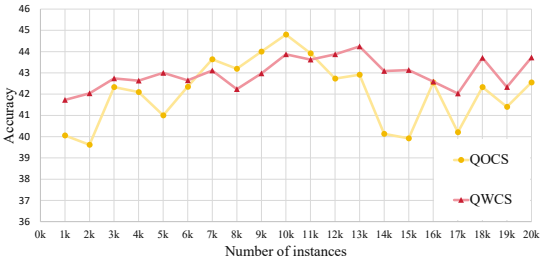
where λ_1 and λ_2 serve as weights, both defaulting to 1, while t_0 represents the maximum runtime set by the user. This optimization problem can be solved using the SQP (Sequential quadratic programming) method (Boggs and Tolle, 1995) to determine the optimal number of clusters and iterations.

B Comparison of the SHED-QOCS and SHED-QWCS

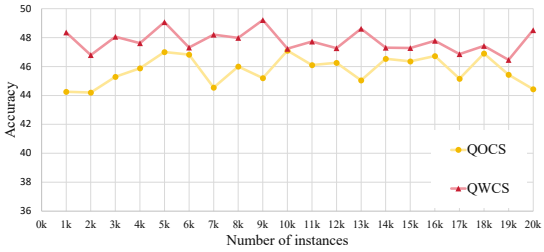
We compared the performance of datasets of varying sample sizes using SHED-QOCS and SHED-QWCS methods. For fine-tuning, we utilized the LLaMA-7B model. Other experimental configurations were aligned with the parameters detailed in Section 4.1.

As figure 5 shows, datasets sampled using the SHED-QWCS approach generally outperform those obtained through SHED-QOCS, particularly with smaller sample sizes. This discrepancy is likely attributable to the limitation of SHED-QOCS in scenarios where the sample size is minimal. In such cases, SHED-QOCS tends to sample data from a limited number of clusters, leading to significant redundancy in the curated dataset.

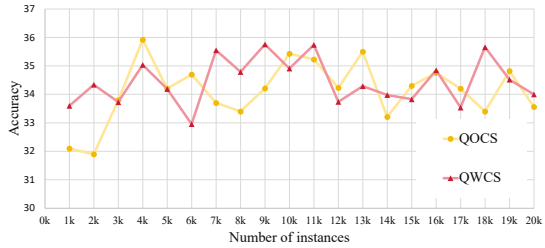
Conversely, it is observed that datasets that achieve the best performance are often those sampled via SHED-QOCS. This improved performance is observed when the sample size is sufficiently large, allowing SHED-QOCS to sample data from a wider range of clusters. The inherent strength of SHED-QOCS lies in its strategic focus on harvesting high-quality data. As the sample size increases to a point where SHED-QOCS can effectively draw from multiple clusters, its advantage in prioritizing data quality becomes significantly beneficial.



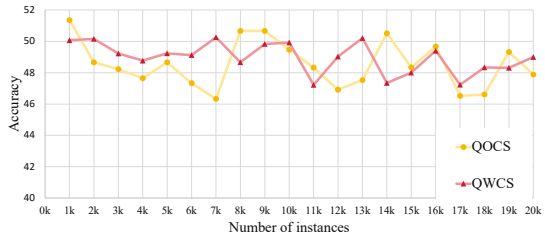
(a) Original dataset: MMLU. Test set: MMLU



(b) Original dataset: MMLU. Test set: ARC-challenge



(c) Original dataset: WizardLM. Test set: MMLU



(d) Original dataset: WizardLM. Test set: ARC-challenge

Figure 5: Results of curated datasets of different samples.

Therefore, we suggest that users select between these two sampling methods based on the sample size they desire.