Efficient Data Selection at Scale via Influence Distillation

Mahdi Nikdan [1 2 3] **Vincent Cohen-Addad** [2] **Dan Alistarh** [1 4] **Vahab Mirrokni** [2]

Abstract

Effective data selection is critical for efficient training of modern Large Language Models (LLMs). This paper introduces Influence Distillation, a new mathematically-justified framework for data selection that leverages second-order data weighting to enhance LLM fine-tuning. By distilling the influence of training data on a target distribution, our method assigns optimal, model-specific weights to training samples, ensuring that fine-tuning yields strong performance in the target domain. We derive optimal weights for both Gradient Descent and Adam optimizers. To reduce computational cost, we propose a clustering-based approach that assigns weights at the cluster level rather than to individual samples. We validate Influence Distillation by applying it to instruction tuning on the Tulu V2 dataset, while targeting various tasks such as GSM8k, SQuAD and MMLU. Experiments with the Llama2-7B model demonstrate an average of 0.9% improvement in accuracy on 6 tasks compared to random selection, while being comparable with computationally heavier methods such as RDS+.

1. Introduction

The rise of Large Language Models (LLMs) has driven significant advances in natural language processing; yet, training and fine-tuning these models requires massive computational resources and carefully-curated datasets. One key direction towards improved training efficiency has been via *data selection and data weighting methods* (Xia et al., 2024; Yin & Rush, 2024; Antonello et al., 2020; Marion et al., 2023; Ankner et al., 2024; Li et al., 2023; Ivison et al., 2025; Axiotis et al., 2024; Xie et al., 2023a; Engstrom et al.; Huang et al., 2024). However, existing approaches often rely on heuristics—such as perplexity-based filtering—or require expensive proxy model training or expensive embedding functions to generate data representations.

[1]Institute of Science and Technology Austria (ISTA) [2]Google Research [3]Work done while at Google [4]Neural Magic. Correspondence to: Mahdi Nikdan <mahdi.nikdan@ista.ac.at>.

More precisely, existing methods face several limitations. First, they typically use fixed features or representations for samples (e.g., embeddings) that may not capture the full relationship between training samples and the target distribution (Yin & Rush, 2024; Antonello et al., 2020; Marion et al., 2023; Ankner et al., 2024). Second, methods that update weights during training lack theoretical justification and can be unstable (Xie et al., 2023a; Huang et al., 2024). Finally, approaches that rely on reference model training or costly embeddings are computationally intensive and often challenging to scale (Li et al., 2023; Xia et al., 2024; Ivison et al., 2025). Thus, there is still a clear need for a mathematically-grounded, efficient framework for data selection that directly optimizes for performance on the target data distribution and task.

Contribution. We present Influence Distillation, a new approach for data selection that, given a pre-trained model and target task, formulates the training samples' influence on a target distribution by a second-order approximation. Unlike previous methods, Influence Distillation directly optimizes sample weights by analyzing how they affect model performance through the lens of second-order information. To make this efficient for large-scale settings, we introduce a clustering-based approximation that assigns weights at the cluster level rather than to individual samples.

One key technical insight behind Influence Distillation is that we can estimate a sample's influence on the loss by examining how it affects the loss on the target distribution after one step of gradient-based optimization. By refining this relationship, we obtain a quadratic optimization objective that we show can be solved efficiently.

Specifically, we derive versions of this framework for both standard gradient descent and adaptive (Adam) optimizers, with theoretical justification for the resulting weight derivations. To handle computational constraints with large datasets, we cluster samples and assign weights per cluster, rather than per example.

We validate our approach through instruction tuning experiments on standard open LLMs, considering the Tulu V2 (Ivison et al., 2023) training dataset, while targeting advanced reasoning tasks like MMLU, mathematics and code. The results demonstrate that Influence Distillation outperforms random selection by a large margin and performs

comparable to substantially more expensive methods.

2. Related Work

Data selection ('pruning') and weighting methods have become increasingly important in the context of efficient LLM training. In a celebrated paper, Sorscher et al. (2022) et al. showed that (model-agnostic) data pruning, and in particular deduplication, helps go beyond scaling laws for LLMs. This was later further improved by Abbas et al. (2023).

Early work on model-dependent data pruning focused on heuristics like perplexity-based filtering and confidence-based selection: Marion et al. (2023) found that selecting examples with moderate perplexity scores often outperforms training on the full dataset or examples selected by other metrics. Do & Gaspers (2019) introduced DSIR, which uses importance resampling based on n-gram features to select relevant training examples, with promising results on mathematical reasoning and clinical text summarization. Similarly, Xie et al. (2023b) proposed clustering loss trajectories to identify representative training examples, though their approach focused more on general domain adaptation rather than specific target distributions. Another approach, so-called Classifier, was introduced by Brown et al. (2020) and has been employed in subsequent work (Gao et al. (2020); Chowdhery et al. (2023); Du et al. (2022). Other strategies include selecting examples that maximize the loss difference between LMs trained on candidate and reference datasets (Moore & Lewis (2010); Axelrod (2017); Feng et al. (2022)). Simpler, yet common, techniques involve filtering documents based on length or the presence of excessive special characters (Raffel et al. (2020); Xie et al. (2023b)). A related, though distinct, task in the LM domain is optimizing the weights for sampling from mixed data sources (Chen et al. (2024); Albalak et al. (2023)).

Recent work has also highlighted the importance of considering the training dynamics when selecting data. Zhou et al. (2023) proposed measuring "learnability" based on loss changes during training, while Swayamdipta et al. (2020) introduced "dataset cartography" to analyze training dynamics across examples. These methods provide useful signals about which examples are most valuable for training; at the same time, they require training reference models which can be computationally expensive. For large-scale applications, Bhatt et al. (2024) evaluated various data selection approaches for LLM fine-tuning, and found that facility-location selection based on hidden representations was particularly effective. However, Tirumala et al. (2023) observed that generating these representations for large datasets remains computationally challenging. More recently, Engstrom et al. (2024) framed the data selection problem as an optimization problem: Given the learning algorithm, find the subset of the data that maximizes the performance of the trained model. To obtain an efficient solution, they design a model that given a subset of the training data S and a target example t, predicts the loss of the model trained on S on t. Axiotis et al. (2024) recently use coreset-related ideas to propose a computationally efficient way of sampling an unbiased estimator of the model loss from the training data so as to train on a smaller input.

While previous methods like DSIR and facility location selection rely on fixed features or representations, our method directly optimizes sample weights based on their influence on the target distribution through a second-order approximation. Importantly, this does not require training proxy model to predict the value of the elements and is computed directly from the input, model and learning algorithm. Unlike curriculum learning or confidence-based approaches that update weights during training, we derive optimal weights analytically for both SGD and Adam optimizers. In contrast to methods that require training reference models, our clustering-based approximation allows efficient weight computation without extensive pre-training.

There is a large body of work on data selection methods for other learning tasks and mode, and it is beyond the scope of this paper to provide a detailed overview. We refer the reader to (Kaushal et al., 2019; Killamsetty et al., 2021; Wei et al., 2015; Chen et al., 2023; Cao et al., 2023; Sener & Savarese, 2017) and references therein.

3. Method

3.1. Problem and Notation

Let $\boldsymbol{\theta} \in \mathbb{R}^d$ be the model parameters. For any dataset D of size n and any vector of sample weights $\boldsymbol{w} = [w_1, w_2, ..., w_n]^T$, denote $\mathcal{L}(\boldsymbol{\theta}; D, \boldsymbol{w}) = \frac{1}{n}\sum_{i=1}^{n} w_i\, \ell(\boldsymbol{\theta}; D_i)$ as the weighted average of the model loss ℓ on the samples of dataset D at point $\boldsymbol{\theta}$. Additionally, define $\mathcal{M}(\boldsymbol{\theta}; D, \boldsymbol{w})$ as a training mechanism that returns the parameters after being trained on a dataset D weighted by $\boldsymbol{w}$. Unless otherwise stated, we will assume $\mathcal{M}$ is simply one step of (full) gradient descent.

Let $\mathcal{S}$ and $\mathcal{T}$ represent the training (source) and downstream (target) distributions, respectively. Assume we have access to a dataset S sampled from $\mathcal{S}$ and a small representative dataset T from $\mathcal{T}$. Our high-level goal will be to determine sample weights $\boldsymbol{w}^*$ such that:

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \; \mathcal{L}(\mathcal{M}(\boldsymbol{\theta}; S, \boldsymbol{w}); T, \mathbb{1}) \tag{1}$$

where $\mathbb{1} \in \mathbb{R}^{|T|}$ represents the all-ones vector. In words, we wish to find sample weights $\boldsymbol{w}$ for instances within the source dataset S, such that training on S using these weights results in minimal loss on the target dataset T. Notably, this notation also allows for the special case of $\mathcal{S} = \mathcal{T}$, where our

method would find weights that maximize in-distribution loss improvement.

3.2. A Running Example

Throughout this section, we utilize a toy training setting to illustrate variants of our method. Specifically, we consider a linear regression model parameterized by $\boldsymbol{\theta}$ with the loss function $\ell(\boldsymbol{\theta}; \mathbf{x}, y) = (\boldsymbol{\theta}^T \mathbf{x} - y)^2$ for any $\boldsymbol{\theta}, \mathbf{x} \in \mathbb{R}^d, y \in \{0, 1\}$. For the source dataset, we sample 256 random instances from the first two classes of the CIFAR-10 dataset (Krizhevsky, 2009) and combine them with 256 synthetic samples generated from a Gaussian distribution with the same mean and standard deviation as the real samples. The target dataset consists of another set of 256 samples from CIFAR-10. We use gradient descent with a learning rate of 10^{-3} as the optimizer. Finally, the loss values are reported on a validation dataset of size 256, also sampled from CIFAR-10.

3.3. Influence Distillation

Case 1: Unconstrained Weights. Let $\mathbf{g}_T(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}; T, \mathbb{1})$ and $\mathbf{H}_T(\boldsymbol{\theta}) = \nabla^2_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}; T, \mathbb{1})$ denote the gradient vector and Hessian matrix of the loss with respect to the model parameters on the target dataset. Construct $\mathbf{G}_S(\boldsymbol{\theta}) \in \mathbb{R}^{|S| \times d}$ by stacking the gradients of the loss with respect to $\boldsymbol{\theta}$ across samples of S. As mentioned before, assume $\mathcal{M}$ is one step of gradient descent, i.e., $\mathcal{M}(\boldsymbol{\theta}; \mathcal{D}, \boldsymbol{w}) = \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}; D, \boldsymbol{w}) = \boldsymbol{\theta} - \frac{\eta}{|S|} \mathbf{G}_S^T(\boldsymbol{\theta}) \boldsymbol{w}$, where η denotes the learning rate. We estimate Objective 1 by:

$$
\begin{aligned}
\boldsymbol{w}^* &= \arg\min_{\boldsymbol{w}} \ \mathcal{L}(\mathcal{M}(\boldsymbol{\theta}; S, \boldsymbol{w}); T, \mathbb{1}) \\
&= \arg\min_{\boldsymbol{w}} \ \mathcal{L}(\boldsymbol{\theta} - \frac{\eta}{|S|} \mathbf{G}_S^T(\boldsymbol{\theta}) \boldsymbol{w}; T, \mathbb{1}) \\
&\approx \arg\min_{\boldsymbol{w}} \ [\mathcal{L}(\boldsymbol{\theta}; T, \mathbb{1}) - \frac{\eta}{|S|} \mathbf{g}_T^T(\boldsymbol{\theta}) \mathbf{G}_S^T(\boldsymbol{\theta}) \boldsymbol{w} \\
&\qquad + \frac{\eta^2}{2\,|S|^2} \boldsymbol{w}^T \mathbf{G}_S(\boldsymbol{\theta}) \mathbf{H}_T(\boldsymbol{\theta}) \mathbf{G}_S^T(\boldsymbol{\theta}) \boldsymbol{w}] \\
&= \arg\min_{\boldsymbol{w}} \ [-\mathbf{g}_T^T(\boldsymbol{\theta}) \mathbf{G}_S^T(\boldsymbol{\theta}) \boldsymbol{w} \\
&\qquad + \frac{\eta}{2\,|S|} \boldsymbol{w}^T \mathbf{G}_S(\boldsymbol{\theta}) \mathbf{H}_T(\boldsymbol{\theta}) \mathbf{G}_S^T(\boldsymbol{\theta}) \boldsymbol{w}] \quad (2)
\end{aligned}
$$

where the approximation comes from a second-order Taylor expansion, i.e., $\mathcal{L}(\boldsymbol{\theta} + \boldsymbol{\delta}; T, \mathbb{1}) \approx \mathcal{L}(\boldsymbol{\theta}; T, \mathbb{1}) + \mathbf{g}_T^T(\boldsymbol{\theta}) \boldsymbol{\delta} + \frac{1}{2} \boldsymbol{\delta}^T \mathbf{H}_T(\boldsymbol{\theta}) \boldsymbol{\delta}$ where $\boldsymbol{\delta}$ is replaced with $-\frac{\eta}{|S|} \mathbf{G}_S^T(\boldsymbol{\theta}) \boldsymbol{w}$.

Next, we define two key objects, $\mathbf{p} \in \mathbb{R}^{|S|}$ and $\mathbf{Q} \in \mathbb{R}^{|S| \times |S|}$, as follows:

$$
\mathbf{p}(\boldsymbol{\theta}) = \mathbf{G}_S(\boldsymbol{\theta}) \mathbf{g}_T(\boldsymbol{\theta}), \quad (3)
$$

$$
\mathbf{Q}(\boldsymbol{\theta}) = \frac{1}{|S|} \mathbf{G}_S(\boldsymbol{\theta}) \mathbf{H}_T(\boldsymbol{\theta}) \mathbf{G}_S^T(\boldsymbol{\theta}), \quad (4)
$$

where, for brevity, we have omitted S and T from the arguments of $\mathbf{p}$ and $\mathbf{Q}$. Additionally, let

$$
f(\boldsymbol{w}; \boldsymbol{\theta}) = -\mathbf{p}(\boldsymbol{\theta})^T \boldsymbol{w} + \frac{\eta}{2} \boldsymbol{w}^T \mathbf{Q}(\boldsymbol{\theta}) \boldsymbol{w}. \quad (5)
$$

Then, the objective in Equation 2 becomes

$$
\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \ f(\boldsymbol{w}; \boldsymbol{\theta}) \quad (6)
$$

In words, f represents a scaled approximation of the change in loss on T when the model at point $\boldsymbol{\theta}$ is trained on S with weights $\boldsymbol{w}$. It is a quadratic function in $\boldsymbol{w}$, as $\mathbf{p}$ and $\mathbf{Q}$ do not depend on $\boldsymbol{w}$. This objective can be minimized in closed form as $\boldsymbol{w}^* = \frac{1}{\eta} \mathbf{Q}(\boldsymbol{\theta})^{-1} \mathbf{p}(\boldsymbol{\theta})$.

Discussion. While simple, the proposed solution has several crucial limitations: (a) it may produce negative or highly irregular sample weights, such as excessively large values, which lack intuitive interpretation, (b) the weights may overfit to the current set of parameters $\boldsymbol{\theta}$, and (c) the weights may also overfit to the target dataset. The first two issues can be easily observed in our running experiment. The irregularity of the weights is illustrated in Figure 1 (left). Furthermore, Figure 1 (right) demonstrates that unconstrained weights become invalidated after just one step of training, suggesting that the weights "overfit" to the current model parameters $\boldsymbol{\theta}$. The third issue (c) is illustrated in Appendix B, particularly for cases where $|S| \gg |T|$.

Case 2: Robust Weights. We modify Objective 6 to address the above limitations. First, we restrict the weights to non-negative values, i.e., $\forall\ 1 \leq i \leq |S|$: $\boldsymbol{w}_i \geq 0$. Second, we require the weights to sum to the size of the source dataset, $\boldsymbol{w}^T \mathbb{1} = |S|$. This prevents weights from becoming excessively large and ensures that rescaling the weights does not change the effective step size: using $\alpha \boldsymbol{w}$ with learning rate η is equivalent to using $\boldsymbol{w}$ with learning rate $\alpha\eta$.

To mitigate "overfitting", a standard approach is to add a regularization term. Indeed, Appendix A derives such a term for linear models. In the general case, we employ a simple L2 regularization term, i.e., $\frac{\lambda}{2} \|\boldsymbol{w}\|_2^2$.

The Robust Influence Objective. Hence, we define the robust Influence Distillation objective as below:

$$
\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \ f(\boldsymbol{w}; \boldsymbol{\theta}) + \frac{\lambda}{2} \|\boldsymbol{w}\|_2^2, \ \ s.t. \ \begin{cases} \boldsymbol{w} \geq 0 \\ \boldsymbol{w}^T \mathbb{1} = |S| \end{cases} \quad (7)
$$

Refer to Section 3.7 for a discussion on how we tune λ in practice.

We compute the robust weights with $\lambda \in \{0.01, 0.02, 0.03\}$ in the context of our running example. Figure 1 (right) highlights the effectiveness of these robust weights, showing that all three configurations outperform the default weights while remaining stable throughout training. Additionally, Figure 1 (middle) depicts the distribution of weights for $\lambda = 0.02$.

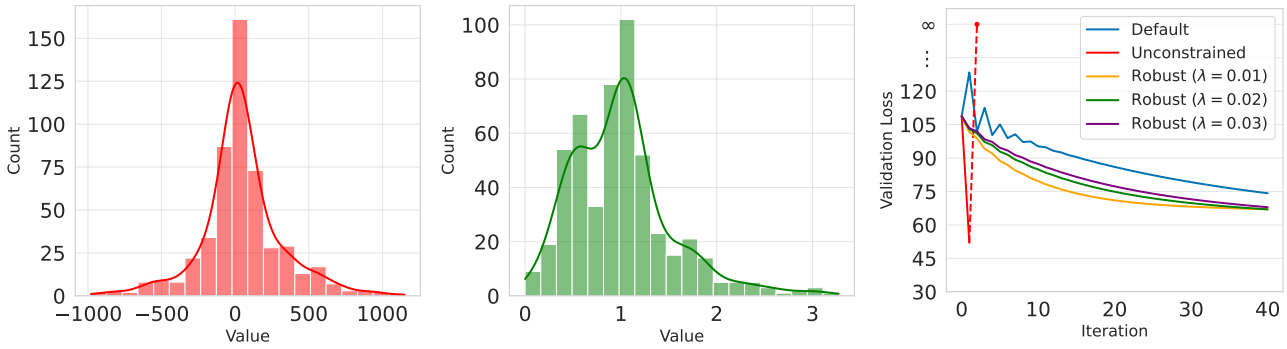

Figure 1: (Left) Distribution of unconstrained weights, (Middle) Distribution of robust weights for $\lambda = 0.02$, and (Right) validation loss during training with different variants in the running experiment setting. Robust weights are found by minimizing Objective 7 using the SLSQP algorithm (Kraft, 1988) implemented in the SciPy library (Virtanen et al., 2020).

Adam Optimizer. The Adam optimizer (Kingma, 2014) is the default choice for fine-tuning LLMs. Therefore, we tailor our method for Adam optimizers. To this end, we employ a greedy approach, where we assume the first- and second-order momentums ($\mathbf{m}$ and $\mathbf{v}$, respectively) are fixed after a warm-up. In this case, the $\mathbf{Q}^{\text{Adam}}$ and $\mathbf{p}^{\text{Adam}}$ objects are calculated as follows:

$$\mathbf{p}^{\text{Adam}}(\boldsymbol{\theta}) = \mathbf{G}_S^{\text{Adam}}(\boldsymbol{\theta})(\mathbf{g}_T(\boldsymbol{\theta}) - \eta * \mathbf{H}_T(\boldsymbol{\theta})\mathbf{b}), \quad (8)$$

$$\mathbf{Q}^{\text{Adam}}(\boldsymbol{\theta}) = \frac{1}{|S|}\mathbf{G}_S^{\text{Adam}}(\boldsymbol{\theta})\mathbf{H}_T(\boldsymbol{\theta})\mathbf{G}_S^{\text{Adam}}(\boldsymbol{\theta})^T, \quad (9)$$

where $\mathbf{b} = \frac{\beta_1 \mathbf{m}}{(1-\beta_1^s)(\sqrt{\frac{\mathbf{v}}{1-\beta_2^s}}+\epsilon)}$, and $\mathbf{G}_S^{\text{Adam}}(\boldsymbol{\theta})$ is constructed by element-wise multiplying every row of $\mathbf{G}_S(\boldsymbol{\theta})$ by $\mathbf{a} = \frac{1-\beta_1}{(1-\beta_1^s)(\sqrt{\frac{\mathbf{v}}{1-\beta_2^s}}+\epsilon)}$. Additionally, s is the number of warmup steps, and $(\beta_1, \beta_2, \epsilon)$ are Adam hyperparameters. See Appendix E for more details.

3.4. Overhead of Calculating Q

Assuming memory constraints limit us to storing a constant number of gradients, computing $\mathbf{Q}$ requires $\Theta(|S|^2)$ gradient evaluations, which quickly becomes computationally infeasible as the number of samples increases.

Recalling the definition of $f(\boldsymbol{\theta}; \boldsymbol{w})$ in Equation 5, we note that the second-order term is scaled by η, the training learning rate. In practice, η typically ranges from 10^{-3} to 10^{-6} when using the Adam optimizer. This observation motivates us to investigate the regime in which the second-order term becomes negligible relative to the first-order term, allowing us to omit the computation of $\mathbf{Q}$ altogether.

To this end, we conduct an empirical study using the LLaMA 3.2-1B model (Grattafiori et al., 2024). We select S as a random subset of 256 samples from the Tulu V2 instruction tuning dataset (Ivison et al., 2023), and consider two cases for T: (1) 8 random samples from the GSM8K training set (Cobbe et al., 2021), (2) 3 random samples from each class of the BBH dataset (Suzgun et al., 2022). Additional dataset details are provided in Section 4.1 and Appendix G.

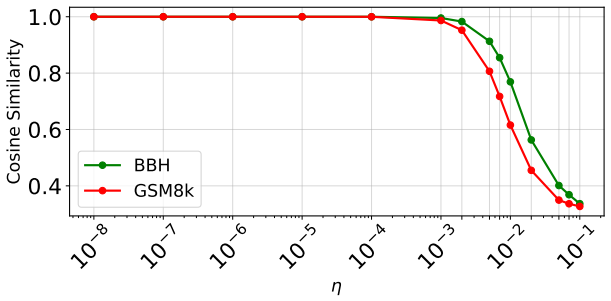

Figure 2: Cosine similarity of first-order Influence Distillation weights and that of second-order with different learning rates.

We fix the regularization coefficient to $\lambda = 0.01$, and compute the optimal SGD weights for learning rates η ranging from 10^{-8} to 10^{0}. Figure 2 plots the cosine similarity between the resulting weights and those obtained with $\eta = 0$, which corresponds to the first-order Influence Distillation. Notably, the similarity remains high for practical learning rates and only begins to decay below $\eta = 10^{-3}$. Even at $\eta = 10^{-2}$, we observe similarities of over 0.6. These findings suggest that for typical LLM training regimes, the first-order approximation of Influence Distillation is sufficient. This aligns with existing gradient-based influence methods, such as those introduced by Xia et al. (2024).

3.5. Clustered Influence Distillation

In this section, we highlight and address two challenges associated with the naive Influence Distillation.

1. **Scalability with Large Training Sets.** In practical

scenarios, the target dataset size, $|T|$, is typically manageable; however, the size of the training dataset, $|S|$, can become significantly large. The naive Influence Distillation requires calculating gradients individually for each sample in S at least once, resulting in substantial computational costs even with access to infinite memory.

2. **Importance Division Among Similar Samples.** Unlike previous methods such as Xia et al. (2024), our objective formulation considers interactions among data samples. Specifically, if a sample is assigned an influence score of x, duplicating this sample r times will distribute its influence equally among the duplicates, assigning each one an influence score of $\frac{x}{r}$. Therefore, a simple top-k selection could fail as important samples may receive lower scores merely because many similar samples exist in the dataset.

To address these issues, we make the heuristic assumption that *similar training samples should receive similar influence scores*. Motivated by this assumption, we propose Clustered Influence Distillation.

Let $\mathbf{h}(x) : S \to \mathbb{R}^m$ denote a function projecting training samples onto an m-dimensional embedding space. Consider a clustering of S within this embedding space represented by $(C, \mathbf{n}, c)$, where:

- $C \subseteq S$, with $|C| = k$, is the set of cluster centers;
- $\mathbf{n} \in \mathbb{N}^k$ is the vector of cluster sizes;
- $c(x) : S \to C$ maps each sample in S to its corresponding cluster center.

Additionally, define $\mathbf{N} = \text{diag}(\mathbf{n})$ as the diagonal matrix containing $\mathbf{n}$ along its diagonal.

Given this clustering, we impose the following constraint on the Influence Distillation objective in Equation 7:

$$\forall x, y \in S : c(x) = c(y) \Rightarrow \boldsymbol{w}_x = \boldsymbol{w}_y,$$

where $\boldsymbol{w}_x$ and $\boldsymbol{w}_y$ denote the elements of $\boldsymbol{w}$ corresponding to samples x and y, respectively. This constraint enforces identical influence values for samples within the same cluster.

Let $\boldsymbol{w}^*$ denote the optimal solution to the original robust objective (Equation 7) under this clustering constraint. We propose an efficient approximation of $\boldsymbol{w}^*$ by first solving for a vector of cluster-level weights $\boldsymbol{w}_C^* \in \mathbb{R}^{|C|}$ and subsequently assigning these cluster-level weights to individual samples within each cluster, forming the approximation $\hat{\boldsymbol{w}}^* \in \mathbb{R}^{|S|}$.

Clustered Objective. Define $\mathbf{G}_C(\boldsymbol{\theta}) \in \mathbb{R}^{k \times d}$ as the stacked gradients corresponding to the cluster centers in C. Additionally, define clustered versions of the functions $\mathbf{p}(.)$, $\mathbf{Q}(.)$, and $f(.)$ from Equations 3, 4, and 5 as follows:

$$\mathbf{p}_C(\boldsymbol{\theta}) = \mathbf{N}\mathbf{G}_C(\boldsymbol{\theta})\mathbf{g}_T(\boldsymbol{\theta}), \tag{10}$$

$$\mathbf{Q}_C(\boldsymbol{\theta}) = \frac{1}{|S|}\mathbf{N}\mathbf{G}_C(\boldsymbol{\theta})\mathbf{H}_T(\boldsymbol{\theta})\mathbf{G}_C^T(\boldsymbol{\theta})\mathbf{N}, \tag{11}$$

$$f_C(\boldsymbol{w}_C; \boldsymbol{\theta}) = -\mathbf{p}_C(\boldsymbol{\theta})^T \boldsymbol{w}_C + \frac{\eta}{2}\boldsymbol{w}_C^T \mathbf{Q}_C(\boldsymbol{\theta})\boldsymbol{w}_C. \tag{12}$$

Finally, we compute the cluster-level optimal weights $\boldsymbol{w}_C^*$ (and consequently $\hat{\boldsymbol{w}}^*$) by optimizing the following *Clustered Robust Objective*:

$$\boldsymbol{w}_C^* = \arg\min_{\boldsymbol{w}_C} \; f_C(\boldsymbol{w}_C; \boldsymbol{\theta}) + \frac{\lambda}{2}\boldsymbol{w}_C^T \mathbf{N} \boldsymbol{w}_C, \text{s.t.} \begin{cases} \boldsymbol{w}_C \geq 0 \\ \boldsymbol{w}_C^T \mathbf{n} = |S| \end{cases} \tag{13}$$

Justification. First, we justify the validity of this approximation. The regularization term and constraints in the clustered objectives are equivalent to those in the original Objective 7. Moreover, Appendix F demonstrates that $f(\hat{\boldsymbol{w}}^*; \boldsymbol{\theta})$ and $f(\boldsymbol{w}^*; \boldsymbol{\theta})$ remain close as long as samples with similar gradients remain close in the embedding space $\mathbf{h}(.)$. See Section 3.6 for our choice of embedding function.

Complexity. The computational complexity of the Clustered Influence Distillation now depends on $|C|$ rather than $|S|$. Since $|C|$ can be directly controlled and is typically significantly smaller than $|S|$, this drastically reduces computational overhead.

Clustered Sampling. We now describe our sampling strategy. Suppose we are asked to select k samples. Let $\mathbf{m} = \mathbf{n} * \boldsymbol{w}_{\mathcal{C}}$ denote the *importance masses* assigned clusters, where $\mathbf{n}$ represents the cluster sizes and $\boldsymbol{w}_{\mathcal{C}}$ the associated weights and $*$ is element-wise multiplication. For a sampling hyperparameter α, we allocate

$$k_i = \frac{m_i^\alpha}{\|\mathbf{m}\|_\alpha^\alpha} \cdot k$$

samples to cluster i, and assign each selected sample a weight of $\frac{m_i}{k_i}$.

The parameter α governs the tradeoff between sampling and weighting. Larger values of α emphasize sampling from more important clusters. Notably, when $\alpha = 0$, the method reduces to pure weighting (i.e., uniform sampling with importance weighting), and when $\alpha = 1$, it reduces to importance sampling with uniform weights. By default, we set $\alpha = 0.5$ to balance these two extremes.

3.6. Embedding Function

As described in Section 3.5, we perform clustering in an embedding space that is efficient to compute for all samples in the dataset and effectively captures gradient similarities. To this end, we introduce the *Jacobian-vector Product (JVP) Embeddings*, detailed below. In Appendix H, we evaluate

the efficiency of JVP and several alternative embeddings, as well as their ability to capture gradient similarity.

JVP Embeddings. For a sample $x \in S$, we define the JVP embedding as:

$$h_{JVP}(x; \mathcal{N}, \ell, V) = \frac{1}{|V|} \sum_{\mathbf{v} \in V} \frac{\partial \mathcal{N}_\ell(x)}{\partial \boldsymbol{\theta}_\ell} \cdot \mathbf{v} \qquad (14)$$

where $\mathcal{N}$ is the network being trained, $\mathcal{N}_\ell(\cdot)$ denotes the logits of the next predicted token after running the first ℓ layers (or transformer blocks, in the case of LLMs), $\boldsymbol{\theta}_\ell$ is the vector of parameters of the first ℓ layers, V is a set of random vectors with the same shape as $\boldsymbol{\theta}_\ell$, and $\frac{\partial \mathcal{N}_\ell(x)}{\partial \boldsymbol{\theta}_\ell}$ is the Jacobian matrix of $\mathcal{N}_\ell(x)$ with respect to $\boldsymbol{\theta}_\ell$. In words, JVP embeddings compute projections of the Jacobian of an intermediate activation with respect to the parameters preceding that activation.

In practice, computing the JVP for each vector $\mathbf{v} \in V$ costs approximately the same as running the partial forward pass $\mathcal{N}_\ell(x)$, using forward-mode automatic differentiation, which is supported by many deep learning frameworks.

3.7. Tuning the Regularization Coefficient.

Finally, we describe how the regularization constant λ in Equation 7 is selected. Appendix I shows that when $\lambda = 0$, the entire weight mass is assigned to a single element, and that increasing λ gradually decreases the sparsity of the solution—that is, more samples receive non-zero weight. Notably, in the limit as $\lambda \to \infty$, the solution becomes fully dense (i.e., a uniform vector of all ones).

In practice, we tune λ via binary search to achieve a target sparsity level s, defined as the proportion of samples whose weights are exactly zero. Unless otherwise specified, we set the default sparsity level to $s = 0.5$. This choice reflects a balance between sample diversity and importance.

4. Experiments

In this section presents evaluate Influence Distillation across several challenging tasks. We start by detailing the datasets, the model, and hyperparameters used in our experiments. Then we present our main results.

4.1. Experimental Setting

We largely follow the challenging setup introduced by Ivison et al. (2025), where many existing data selection methods have been shown to fail.

Training Dataset. We use Tulu V2 (Ivison et al., 2023), a collection of 9 instruction-tuning datasets containing approximately 5.8 million samples. Detailed descriptions of each component dataset are provided in Appendix G. In each experiment, we randomly sample 200k examples from Tulu V2, and then use sampling methods to pick a subset of 10k samples from this pool.

Target Datasets. We evaluate on six target datasets: MMLU (Hendrycks et al., 2021b;a), GSM8k (Cobbe et al., 2021), BBH (Suzgun et al., 2022), TyDIQA (Clark et al., 2020), Codex (Chen et al., 2021), and SQuAD (Rajpurkar et al., 2016). For each, we sample between 8 and 500 examples from their train, dev, or eval splits, following the setup of Ivison et al. (2025). More details are available in Appendix G.

Model. We fine-tune the LLaMA 2–7B model (Touvron et al., 2023), consistent with the model used in the Tulu V2 paper (Ivison et al., 2023), as well as in the experiments of Ivison et al. (2025).

Baselines. We consider two baselines: (1) Random selection, which picks samples uniformly at random, and (2) The state-of-the-art RDS+ (Ivison et al., 2025) embedding-based method, where the embeddings are computed by a position-weighted mean pool of the last hidden layer states. Notably, RDS+ requires running full forward passes on all the samples in the pool, which is significantly more expensive that our JVP embeddings.

Embedding and Clustering. We experiment with two types of embeddings: (1) *Semantic embeddings*: we employ the GTR sentence-transformer (Ni et al., 2021) as a fast semantic embedder. Our results, along with those in Appendix H, indicate that semantic embeddings are not well-suited for our purpose. (2) *JVP embeddings*: as described in Section 3.6, we use JVP-based embeddings with $\ell = 4$ and $|V| = 1$.

For clustering, we apply the K-means algorithm with 8192 clusters using the cosine distance metric. We utilize the CuML GPU (Raschka et al., 2020) implementation to perform K-means clustering 10 times with different random initializations and select the run with the best objective value. To ensure that cluster centers correspond to actual samples, we apply a K-medoids algorithm on the K-means output.

Hyperparameters. We use the AdamW optimizer with a learning rate of 2×10^{-5} for 3 epochs. The sequence length is fixed at 2048, and we use a micro batch size of 1 with gradient accumulation over 128 steps. All experiments are performed on a single H100 GPU. We report each experiment with 3 seeds (including selecting $200k$ samples from the full Tulu V2 dataset). The JVP embeddings are calculated after a warm-up on $10k$ randomly selected samples, then the model is restarted to the original weights and trained on the selected samples.

Method	MMLU	GSM8k	BBH	TyDIQA	CODEX	SQuAD	Avg.	Sampling FLOPS
Random	45.6 ± 0.43	17.5 ± 1.08	41.8 ± 0.20	51.6 ± 0.38	27.0 ± 0.60	80.8 ± 1.05	**44.4**	0
RDS+	46.3 ± 0.33	$\mathbf{20.2 \pm 2.77}$	$\mathbf{42.7 \pm 0.61}$	50.5 ± 0.84	$\mathbf{30.4 \pm 0.96}$	$\mathbf{85.3 \pm 0.22}$	**45.9**	$2\|S\|d \approx 2800\ TF$
Ours (GTR)	46.0 ± 0.57	18.4 ± 1.26	42.5 ± 0.27	52.1 ± 0.10	27.5 ± 1.15	82.1 ± 0.96	**44.8**	$2\|S\|d_{\text{GTR}} + 6(\|C\|+\|T\|)d \approx 400\ TF$
Ours (JVP)	$\mathbf{46.7 \pm 0.81}$	18.8 ± 1.61	42.0 ± 0.27	$\mathbf{53.8 \pm 0.69}$	27.5 ± 2.09	82.9 ± 0.82	**45.3**	$\frac{1}{4}\|S\|d + 6(\|C\|+\|T\|)d \approx 700\ TF$

Table 1: Accuracy ($\pm$ standard deviation) and sampling runtime estimation of methods on various tasks. Notably, d denotes the size of the model being trained (7B), d_{GTR} refers to the size of the GTR embedding model (110M), and TF stands for Trillion FLOPS. Additioanlly, $S = 200k$ and T denote the size of training and target datasets, while $|C|= 8192$ denotes the number of clusters. In the final approximation, we assume a target dataset size is fixed at 100 for simplicity.

4.2. Main Experiments.

Table 1 demonstrates that Influence Distillation with JVP embeddings consistently outperforms both Random selection and Influence Distillation with GTR embeddings, while achieving accuracy that is, on average, competitive with RDS+, which requires full forward passes over the entire training pool. The table also reports the approximate FLOPS required to perform sampling for each method. We adopt the estimation from Kaplan et al. (2020), where each forward pass incurs a cost of $2d$ FLOPS and each backward pass incurs $4d$ FLOPS, with d denoting the number of model parameters. These estimates indicate that Influence Distillation with JVP embeddings is approximately $4\times$ more efficient than RDS+, while maintaining comparable accuracy.

4.3. Effect of α

While the sampling hyperparameter α is set to 0.5 in the main experiments, we additionally evaluate two extreme settings of α to explore its effect. For the low-α case, we set $\alpha = 0.1$, ensuring that clusters with zero mass receive no samples. For the high-α case, we use $\alpha = 1.0$. These two configurations represent the extremes of the sampling spectrum: the former emphasizes pure weighting with close to uniform sampling, while the latter corresponds to pure sampling based on cluster masses without weighting.

Table 2 demonstrates these that $\alpha = 0.5$ performs slightly better than the two extreme cases, striking a balance between importance sampling and diversity.

5. Conclusion

We introduced Influence Distillation, an efficient gradient-based data selection method. Influence Distillation optimizes per-sample weights to maximize model performance on a target distribution, leveraging second-order information. To improve scalability, we adopt a clustering-based strategy that assigns importance weights at the cluster level rather than for individual samples. Our experiments demonstrate the effectiveness of Influence Distillation in an instruction tuning setting, where it outperforms random selection with an average accuracy gain of 0.9%, while remaining competitive with the more computationally intensive RDS+ method.

References

Abbas, A., Tirumala, K., Simig, D., Ganguli, S., and Morcos, A. S. Semdedup: Data-efficient learning at web-scale through semantic deduplication. *arXiv preprint arXiv:2303.09540*, 2023.

Albalak, A., Pan, L., Raffel, C., and Wang, W. Y. Efficient online data mixing for language model pre-training. In *R0-FoMo: Robustness of Few-shot and Zero-shot Learning in Large Foundation Models*, 2023.

Ankner, Z., Blakeney, C., Sreenivasan, K., Marion, M., Leavitt, M. L., and Paul, M. Perplexed by perplexity: Perplexity-based data pruning with small reference models. *arXiv preprint arXiv:2405.20541*, 2024.

Antonello, R., Beckage, N., Turek, J., and Huth, A. Selecting informative contexts improves language model finetuning. *arXiv preprint arXiv:2005.00175*, 2020.

Axelrod, A. Cynical selection of language model training data. *arXiv preprint arXiv:1709.02279*, 2017.

Axiotis, K., Cohen-Addad, V., Henzinger, M., Jerome, S., Mirrokni, V., Saulpic, D., Woodruff, D. P., and Wunder, M. Data-efficient learning via clustering-based sensitivity sampling: Foundation models and beyond. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=WUQ4YzIQt2.

Bhatt, G. et al. An experimental design framework for label-efficient supervised finetuning of large language models. *arXiv preprint arXiv:2401.06692*, 2024.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Voss, G., and Amodei, D. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Cao, Y., Kang, Y., Wang, C., and Sun, L. Instruction mining: Instruction data selection for tuning large language models. *arXiv preprint arXiv:2307.06290*, 2023.

Method	MMLU	GSM8k	BBH	TyDIQA	CODEX	SQuAD	Avg.
GTR ($\alpha = 0.1$)	46.1 ± 0.53	17.4 ± 1.27	42.3 ± 0.40	51.3 ± 0.44	29.7 ± 1.10	81.5 ± 0.68	44.7
GTR ($\alpha = 0.5$)	46.0 ± 0.57	18.4 ± 1.26	42.5 ± 0.27	52.1 ± 0.10	27.5 ± 1.15	82.1 ± 0.96	**44.8**
GTR ($\alpha = 1.0$)	46.2 ± 0.32	17.1 ± 0.38	42.5 ± 0.50	51.8 ± 0.07	28.4 ± 0.55	82.1 ± 0.15	44.7
JVP ($\alpha = 0.1$)	46.4 ± 0.35	18.8 ± 0.37	42.3 ± 0.48	52.8 ± 0.34	28.2 ± 1.15	81.7 ± 1.59	45.0
JVP ($\alpha = 0.5$)	46.7 ± 0.81	18.8 ± 1.61	42.0 ± 0.27	53.8 ± 0.69	27.5 ± 2.09	82.9 ± 0.82	**45.3**
JVP ($\alpha = 1.0$)	46.8 ± 0.29	18.5 ± 0.86	42.7 ± 0.20	53.8 ± 0.49	27.7 ± 0.55	81.9 ± 1.50	45.2

Table 2: Accuracy ($\pm$ standard deviation) of different values of α on various tasks.

Chen, L., Li, S., Yan, J., Wang, H., Gunaratna, K., Yadav, V., Tang, Z., Srinivasan, V., Zhou, T., Huang, H., et al. Alpagasus: Training a better alpaca with fewer data. *arXiv preprint arXiv:2307.08701*, 2023.

Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. D. O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Chen, M., Roberts, N., Bhatia, K., Wang, J., Zhang, C., Sala, F., and Ré, C. Skill-it! a data-driven skills framework for understanding and training language models. *Advances in Neural Information Processing Systems*, 36, 2024.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.

Clark, J. H., Choi, E., Collins, M., Garrette, D., Kwiatkowski, T., Nikolaev, V., and Palomaki, J. Tydi qa: A benchmark for information-seeking question answering in typologically diverse languages. *Transactions of the Association for Computational Linguistics*, 2020.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Do, Q. and Gaspers, J. Cross-lingual transfer learning with data selection for large-scale spoken language understanding. *EMNLP*, 2019.

Du, N., Huang, Y., Dai, A. M., Tong, S., Lepikhin, D., Xu, Y., Krikun, M., Zhou, Y., Yu, A. W., Firat, O., et al. Glam: Efficient scaling of language models with mixture-of-experts. In *International Conference on Machine Learning*, pp. 5547–5569. PMLR, 2022.

Engstrom, L., Feldmann, A., and Madry, A. Dsdm: Model-aware dataset selection with datamodels, 2024. *URL https://arxiv. org/abs/2401.12926*.

Engstrom, L., Feldmann, A., and Madry, A. Dsdm: Model-aware dataset selection with datamodels. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=GC8HkKeH8s.

Feng, Y., Xia, P., Van Durme, B., and Sedoc, J. Automatic document selection for efficient encoder pretraining. *arXiv preprint arXiv:2210.10951*, 2022.

Gao, L., Biderman, S., Black, S., Golding, L., Hoppe, T., Foster, C., Phang, J., He, H., Thite, A., Nabeshima, N., et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Hendrycks, D., Burns, C., Basart, S., Critch, A., Li, J., Song, D., and Steinhardt, J. Aligning ai with shared human values. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021a.

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021b.

Huang, W., Zhang, Y., Guo, S., Shang, Y., and Fu, X. Dynimpt: A dynamic data selection method for improving model training efficiency. *IEEE Transactions on Knowledge and Data Engineering*, 2024.

Ivison, H., Wang, Y., Pyatkin, V., Lambert, N., Peters, M., Dasigi, P., Jang, J., Wadden, D., Smith, N. A., Beltagy, I., et al. Camels in a changing climate: Enhancing lm adaptation with tulu 2. *arXiv preprint arXiv:2311.10702*, 2023.

Ivison, H., Zhang, M., Brahman, F., Koh, P. W., and Dasigi, P. Large-scale data selection for instruction tuning. *arXiv preprint arXiv:2503.01807*, 2025.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Kaushal, V., Iyer, R., Kothawade, S., Mahadev, R., Doctor, K., and Ramakrishnan, G. Learning from less data: A unified data subset selection and active learning framework for computer vision. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1289–1299. IEEE, 2019.

Killamsetty, K., Zhao, X., Chen, F., and Iyer, R. Retrieve: Coreset selection for efficient and robust semi-supervised learning. *Advances in neural information processing systems*, 34:14488–14501, 2021.

Kingma, D. P. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kraft, D. A software package for sequential quadratic programming. *Forschungsbericht- Deutsche Forschungs- und Versuchsanstalt fur Luft- und Raumfahrt*, 1988.

Krizhevsky, A. Learning multiple layers of features from tiny images. Technical report, 2009.

Li, M., Zhang, Y., Li, Z., Chen, J., Chen, L., Cheng, N., Wang, J., Zhou, T., and Xiao, J. From quantity to quality: Boosting llm performance with self-guided data selection for instruction tuning. *arXiv preprint arXiv:2308.12032*, 2023.

Marion, M., Üstün, A., Pozzobon, L., Wang, A., Fadaee, M., and Hooker, S. When less is more: Investigating data pruning for pretraining llms at scale. *arXiv preprint arXiv:2309.04564*, 2023.

Moore, R. C. and Lewis, W. Intelligent selection of language model training data. In *Proceedings of the ACL 2010 conference short papers*, pp. 220–224, 2010.

Ni, J., Qu, C., Lu, J., Dai, Z., Ábrego, G. H., Ma, J., Zhao, V. Y., Luan, Y., Hall, K. B., Chang, M.-W., et al. Large dual encoders are generalizable retrievers. *arXiv preprint arXiv:2112.07899*, 2021.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21 (140):1–67, 2020.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. SQuAD: 100,000+ questions for machine comprehension of text. In Su, J., Duh, K., and Carreras, X. (eds.), *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2383–2392, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1264. URL https://aclanthology.org/D16-1264.

Raschka, S., Patterson, J., and Nolet, C. Machine learning in python: Main developments and technology trends in data science, machine learning, and artificial intelligence. *arXiv preprint arXiv:2002.04803*, 2020.

Sener, O. and Savarese, S. Active learning for convolutional neural networks: A core-set approach. *arXiv preprint arXiv:1708.00489*, 2017.

Sorscher, B., Geirhos, R., Shekhar, S., Ganguli, S., and Morcos, A. Beyond neural scaling laws: beating power law scaling via data pruning. *Advances in Neural Information Processing Systems*, 35:19523–19536, 2022.

Suzgun, M., Scales, N., Schärli, N., Gehrmann, S., Tay, Y., Chung, H. W., Chowdhery, A., Le, Q. V., Chi, E. H., Zhou, D., , and Wei, J. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*, 2022.

Swayamdipta, S., Schwartz, R., Lourie, N., Wang, Y., Hajishirzi, H., Smith, N. A., and Choi, Y. Dataset cartography: Mapping and diagnosing datasets with training dynamics. *EMNLP*, 2020.

Tirumala, K., Simig, D., Aghajanyan, A., and Morcos, A. D4: improving LLM pretraining via document deduplication and diversification. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S. (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/a8f8cbd7f7a5fb2c837e578c75e5b615-Abstract-Datasets_and_Benchmarks.html.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Virtanen, P., Gommers, R., Oliphant, T., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., et al. Fundamental algorithms for scientific computing in python and scipy 1.0 contributors. scipy 1.0. *Nat. Methods*, 17:261–272, 2020.

Wei, K., Iyer, R., and Bilmes, J. Submodularity in data subset selection and active learning. In *International conference on machine learning*, pp. 1954–1963. PMLR, 2015.

Xia, M., Malladi, S., Gururangan, S., Arora, S., and Chen, D. Less: Selecting influential data for targeted instruction tuning. *arXiv preprint arXiv:2402.04333*, 2024.

Xie, S. M., Santurkar, S., Ma, T., and Liang, P. S. Data selection for language models via importance resampling. *Advances in Neural Information Processing Systems*, 36: 34201–34227, 2023a.

Xie, S. M. et al. Smalltolarge (s2l): Scalable data selection for fine-tuning large language models by summarizing training loss trajectories of small models. *arXiv preprint*, 2023b.

Yin, J. O. and Rush, A. M. Compute-constrained data selection. *arXiv preprint arXiv:2410.16208*, 2024.

Zhou, H. et al. Lobass: Gauging learnability in supervised fine-tuning data. *arXiv preprint arXiv:2310.13008*, 2023.