# In2Core: Leveraging Influence Functions for Coreset Selection in Instruction Finetuning of Large Language Models

**Ayrton San Joaquin**[♡] **Bin Wang**[♢] **Zhengyuan Liu**[♢]
**Nicholas Asher**[§,♡] **Brian Lim**[∞,♡] **Philippe Muller**[§,♡] **Nancy F. Chen**[♢,♡,†]
[♡]CNRS@CREATE, Singapore [♢]Institute for Infocomm Research ($I^2R$), A*STAR, Singapore
[§]CNRS, IRIT, France [∞]National University of Singapore, Singapore
[†]Centre for Frontier AI Research (CFAR), A*STAR, Singapore

## Abstract

Despite advancements, fine-tuning Large Language Models (LLMs) remains costly due to the extensive parameter count and substantial data requirements for model generalization. Accessibility to computing resources remains a barrier for the open-source community. To address this challenge, we propose the `In2Core` algorithm, which selects a coreset by analyzing the correlation between training and evaluation samples with a trained model. Notably, we assess the model's internal gradients to estimate this relationship, aiming to rank the contribution of each training point. To enhance efficiency, we propose an optimization to compute influence functions with a reduced number of layers while achieving similar accuracy. By applying our algorithm to instruction fine-tuning data of LLMs, we can achieve similar performance with just 50% of the training data. Meantime, using influence functions to analyze model coverage to certain testing samples could provide a reliable and interpretable signal on the training set's coverage of those test points.

## 1 Introduction

With the advent of Large Language Models (LLMs) exhibiting surprising abilities across a variety of language tasks (Clark et al., 2018; Zellers et al., 2019; Sakaguchi et al., 2019; Hendrycks et al., 2020; Cobbe et al., 2021; Gao et al., 2021; Lin et al., 2022; Beeching et al., 2023), open-source language models have surged in popularity as part of broader efforts to democratize their accessibility (Taori et al., 2023). Furthermore, performance can be improved with custom data that is smaller than their pretraining data, which is known as supervised fine-tuning (Liu et al., 2024b). One type of fine-tuning important to the open-source community is instruction tuning, which allows models to follow a broad or specific-set of instructions and to discuss with users in natural dialogue (Zhang et al., 2023). However, instruction-tuning of open-source LLMs remains limited in multiple areas.

One prominent limitation is the expensive cost of fine-tuning such models, caused by a large number of parameters and data required (Liu et al., 2024a). The popular approach is to scale-up the parameter count and the dataset size, increasing the computation required to train the models. This method to improve LLMs does not lend-well to organizations without massive computing resources, such as the majority of entities who rely on open-source. Another limitation is evaluation, especially for instruction-following models. Evaluation is challenging as a result of LLMs' inherent open-ended generation, where a space of ideal outputs exists, and existing evaluations usually measure ability in a specific area, such as summarization. Furthermore, evaluation sets may be hard to construct for domain-specific abilities. These two issues limit the continued adoption of open-source models.

To alleviate the cost of fine-tuning, many have focused on improving the hardware (Jouppi et al., 2023) or the algorithm of the training process (Hu et al., 2021). To improve evaluation, many have introduced more sophisticated benchmarks (Liang et al., 2022) or rely on automated evaluation using more powerful LLMs (Li et al., 2023a). However, one prominent limitation faced by these approaches is that they do not analyze how the training data can be manipulated to induce a better-performing trained model.

By focusing on the problem of coreset selection, we instead take a data-centric approach using influence functions. Influence functions are a method from robust statistics (Hampel, 1974) and are first adapted to neural-network-based machine learning models by (Koh and Liang, 2017). They approximate how much the model's prediction changes when a particular training point is removed from the training process via the model gradients.

In this work, we aim to show that Influence Func-



*Findings of the Association for Computational Linguistics: EACL 2024*, pages 10324–10335

tions are versatile statistical tools that can address two questions on data suitability: **1) What training points are suitable for the test set? and 2) What test points are suitable given that the model was already trained on a particular training set?** We use influence functions to identify influential data to address the two aforementioned issues by making fine-tuning more efficient by using less data and identifying test points well-covered by the fine-tuned model.

Our method is orthogonal to concurrent efforts to make fine-tuning more efficient for LLMs that focus on modifying the model architecture, such as Simplifying Transformer Blocks (He and Hofmann, 2023), or that focus on reducing the number of trainable parameters, such as LoRA (Hu et al., 2021). It can therefore be combined with these methods. In particular, our method relies on DataInf (Kwon et al., 2023), which works on top of LoRA to calculate influence values.

Our main contributions can be summarized as follows:

- We developed `In2Core`, an algorithm that significantly reduces the size of the training set while creating a model that outperforms the one trained on the full training set.
- With `In2Core`, we can identify whether a testing point is well covered by the training set, thus providing an interpretable explanation of how a given model reacts to a particular test point.
- We further improve the efficiency of the influence function algorithm by limiting the number of model layers in calculating influence values, and we introduce a method to select the optimal number of layers given a memory budget.

## 2 Related Work

### 2.1 Influence Functions

Influence functions, a subset of Data Attribution methods, seek to measure the effect of a given training point/s on a trained model. For a given training point, they seek to capture the change in behavior of a trained statistical model had that single training point not been part of the training dataset (leave-one-out-retraining). It outputs a value, called the influence value, for each training point in question. For further discussion on Data Attribution and Influence Functions in machine learning, we refer to Hammoudeh and Lowd (2024).

The current trend of scaling LLMs to an order of billions of parameters, as well as leveraging huge amounts of training data, pose additional computational challenges for existing influence function methods (Grosse et al., 2023; Koh and Liang, 2017). Recent works seek to adapt data attribution for these kinds of models: TRAK (Park et al., 2023) uses an "influence function-style" estimation that simplifies the Hessian matrix, while DataInf (Kwon et al., 2023) leverages LoRA (Hu et al., 2021) to calculate influence values for fine-tuned LLMs. These methods make data attribution tractable for modern LLMs. In our study, we use DataInf because our focus is on the fine-tuning phase of training.

### 2.2 Coreset Selection for LLMs

Coreset selection aims to select a subset of the full training data, such that a model well-fitted to the coreset also fits to the full data. One of the first to study it for big data viewed it as a compression method for large datasets (Phillips, 2017). In machine learning, Mirzasoleiman et al. (2020) introduced a coreset algorithm to boost iterative gradient-based training. Coreset selection has been recently explored for LLMs, given the obvious problem of expensive training (Li et al., 2023c; Chen et al., 2023) or to understand model properties such as in-context learning (Han et al., 2023). Another line of work such as Han et al. (2023) and Wang et al. (2023) apply coreset selection for selecting the pretraining data of language models. Our work differs by focusing on the fine-tuning data, which is relevant to practitioners and to the open-source community.

### 2.3 Coreset Selection with Influence Functions

Our work is closest to Wang et al. (2023); Yang et al. (2022); Xia et al. (2024) in that they use influence functions to rank data in their selection algorithms. Wang et al. (2023) on the pretraining data, while ours focus on the fine-tuning data. Yang et al. (2022) apply it for vision tasks with clearly-defined evaluation metrics (e.g. image classification) for which they can specify an objective function to achieve that metric, whereas instruction-following does not have a clear evaluation metric and remains an open-problem in the field. We use the perplexity as a coarse proxy to capture what it means to follow instructions effectively. Xia et al. (2024)

also uses a definition of influence to select coresets for instruction-tuning. However, we further show that influence functions can be used to measure how much a trained model's coverage on unseen test samples. Additionally, earlier work (Guo et al., 2020) used influence values on new, but similar training data to further fine-tune a tuned model.

## 3 Influence Functions for Coreset Selection

In this section, we explain what influence functions are, their use in assigning a value for each training datapoint in In2Core, and an algorithmic improvement to use less memory for the influence function algorithm we use, namely DataInf (Kwon et al., 2023). We then explain the In2Core algorithm and perform experiments to showcase the performance of different subset selections of the data. Finally, we discuss the implications of having a smaller and better-quality training dataset, namely faster training and comparable performance.

### 3.1 Preliminaries

Following the formulations of (Koh and Liang, 2017; Cook and Weisberg, 1980), an influence function measures the impact of a given training point on the change in model parameters for a given validation point. It measures this impact by upweighing this training point and measuring the rate of change in the model parameters.

In the context of machine learning, given a model $f$ parameterized by the empirical risk minimizer $\theta*$, we consider how a given training point changes the validation loss. Given a training point $z$ and a validation point $z'$, the influence of $z$ on $f(\theta*)$'s performance on $z'$ is defined as

$$I(z, z') := (\nabla_\theta \ell(z')|_{\theta=\theta*})^T I_{\theta*}(z) \quad (1)$$

where

$$I_{\theta*}(z) := \frac{d\theta^z}{d\epsilon}|_{\epsilon=0} = -H(\theta*)^{-1}\nabla_\theta \ell_k \quad (2)$$

and $H := \nabla^2_\theta(n^{-1}\sum_{i=1}^n \ell(f_\theta(z))$, the Hessian of the empirical loss.

Equation (1) can be extended to an entire validation set $D^{val} := z'^m_{i=1}$ by taking the average gradient of the validation set as

$$I(z, D^{(val)}) := (\frac{1}{m}\sum_{i=1}^{m} \nabla_\theta \ell(z'_i)|_{\theta=\theta*})^T I_{\theta*}(z) \quad (3)$$

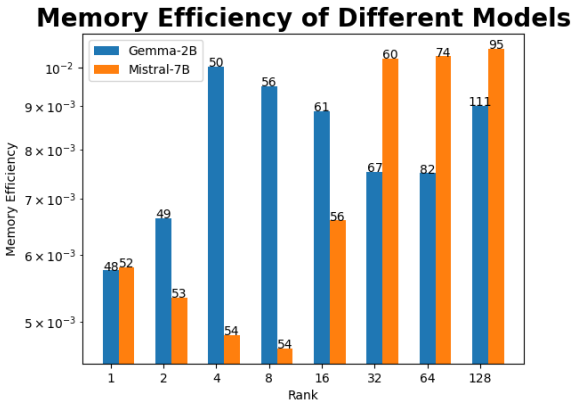



Figure 1: Memory efficiency across various number of LoRA layers to consider when calculating influence values. The numbers above each bar correspond to the virtual CPU memory consumed (in GB). The memory efficiency rate is not linear, and varies across different models. One should sample a subset to calculate the optimal combination of CPU virtual memory and number of layers to use given their hardware constraints.

The influence value and its sign represents the impact of including a particular point in the training set on a given validation point/set, via the validation loss. For clarity, we follow (Pruthi et al., 2020) in using their terminologies. We define

1. **Proponents** - Points with negative influence values. Their addition to the training set reduces the validation loss.

2. **Opponents** - Points with positive influence values. Their addition to the training set increases the validation loss.

For current language models, which rely on transformer-based models with billions of parameters, computing the Hessian term in Equation 2 is extremely expensive. In practice, we rely on the Hessian's estimation. In this paper, we use the estimation of influence from DataInf by (Kwon et al., 2023).

**Efficient DataInf**. While the DataInf algorithm efficiently calculates the gradients of the model parameters for each point, in practice the CPU memory consumption from the gradient collection becomes a bottleneck as the size of the training dataset grows. This is a problem for sizeable fine-tuning datasets such as the Ultrachat subset we use, which contains roughly 50,000 training points. Figure 1 shows the trend for N=250 points.

We found that selecting the number of layers is non-trivial. To illustrate this, let *all-layer* denote

the case where we use every layer with a LoRA adapter. First, using large $k$ can result in marginal gains at approximating influence values from all-layer at the expense of the memory budget. Second, different model architectures give different efficiency, implying $k$ is different for each model (and dataset). These two reasons require a hyperparameter search for $k$.

To enable comparisons across different models given the same dataset, we define the metric Memory Efficiency, $s := \frac{\rho}{\text{number of layers used}}$, where $\rho$ is Spearman's rank correlation coefficient of a particular setup with respect to the case where all layers are used to calculate the gradient (*all-layer*). $\rho$ is a metric of interest because it describes how faithful the ranking of the current setup is to the ranking in the all-layer setup. Note that $k$ layers refer to the first $k$ layers because the first layers capture influence better than the last layers (Yeh et al., 2022). We then simply set $k$ as the number of layers with the highest $s$, subject to our virtual memory budget. For our experiments, we used $k = \{8, 16\}$ for Gemma-2B and Mistral-7B, respectively. Practitioners should perform this preliminary evaluation of memory efficiency on a small subset, in order to avoid out-of-memory (OOM) errors when calculating the influence values.

### 3.2 Algorithm

Given a large dataset $D_{full}$, evaluation set $D_{eval}$, and reference model $f$ fine-tuned on $D_{full}$, to select a coreset $D_p$, where $D_p \subset D_{full}$

1. Calculate the influence of each $z \in D_{full}$using $f$.
2. Rank each $z$ by influence value.
3. Select $p$ and get the top-$p$ proponents as elements of $D_p$.

The first step is the most expensive step as the influence function algorithm visits each point. In particular, it is necessary to get the model gradients for each point, which creates a memory bottleneck. In Section 3.1, we improved the influence function algorithm we use to mitigate this.

The second step provides an ordered list of the training data with respect to each points' influence value. For the third step, note that $p$ is a hyperparameter and depends on the training budget of how many data points the training can accommodate. Given the definition of proponents, the "top" proponents are the points with most negative influence values.

### 3.3 Experimental Setup

For coreset selection, we use Mistral-7B-v0.1 (Jiang et al., 2023) & Gemma-2B (Team et al., 2023) as base models and fine-tune them on subsets of a "full" training dataset, which is a random 50k subset of the first round of dialogues from Ultrachat-200k (Ding et al., 2023). Note that the models we use are allowed for research. We evaluate the fine-tuned models on a disjoint 250-random subset from Ultrachat-200k, following the same format as the training dataset. For both Sections 3.4 and 4.1, we use perplexity (Jelinek et al., 1977) on the evaluation set to measure model performance, given that it is directly related to the loss, whose reduction is the goal of fine-tuning (training loss) and is a component to the calculation of influence values (validation loss).

For simplicity, we refer to the mean perplexity of the model on the validation dataset as perplexity and BERTScore between each validation point and the training dataset as similarity.

### 3.4 Results

**Model Perplexity**. Using Equation (3), we calculate the influence values with respect to the entire evaluation dataset. Figure 3 illustrates our algorithm's performance at coreset selection. The x-axis represents the different selection strategies based on their influence values. Minimum refers to selecting points whose influence values are nearest to zero in absolute terms. Random refers to uniformly-sampled points. In all figures, selecting Proponents consistently leads to the lowest perplexity among the other subset selection strategies. This applies when, with respect to the base models being fine-tuned, the reference model is smaller (as in the case with Gemma-2B or it is of a different architecture with a different pretraining data (as with Phi-2 pretrained on synthetic Textbook-quality data (Li et al., 2023b)). Importantly, selecting the best $h = 25$k proponents, which is half of the full dataset, leads to a model that has lower perplexity than a model trained on the full dataset. This shows that it is possible to achieve better model performance with less training data.

Interestingly, the behavior of Minimum and Opponents are similar. In some cases selecting Minimum leads to a model with higher perplexity than

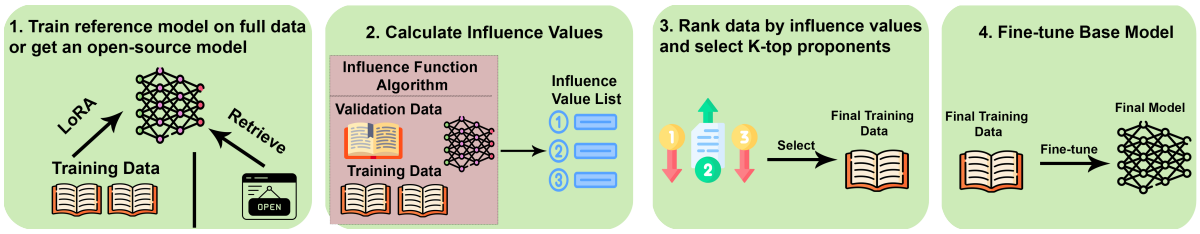
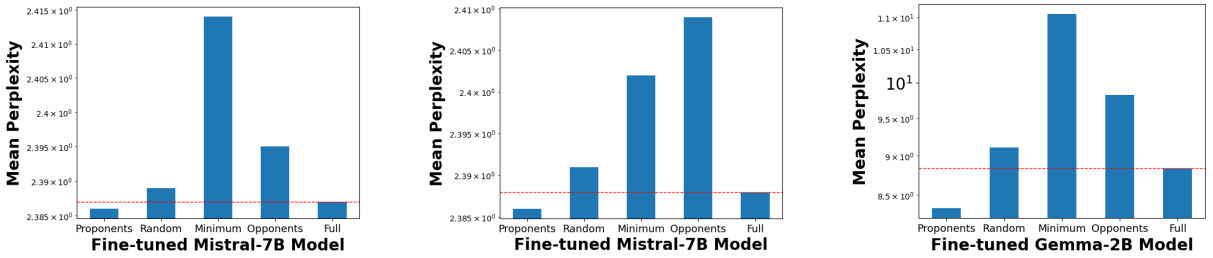



Figure 2: Overview of In2Core for coreset selection. From left to right, we first calculate the influence values of each training point using the validation dataset and a reference model fine-tuned on the full dataset with LoRA. Then, we rank the training points by influence values and select the $h$ highest-scoring proponents as the final training data, where $h$ is a hyperparameter. Finally, we train a base model on this final training data. Both the reference and base model may have distinct architectures from each other.




(a) Performance of Models (Ref: Mistral-7B)

(b) Performance of Models (Ref: Gemma-2B

(c) Performance of Models (Ref: Gemma-2B

Figure 3: Mean Perplexity of models fine-tuned on different coreset selection strategies on the 250-point Ultrachat Evaluation Set. These strategies differ by selecting based on the influence values. 'Full' denotes a model trained on the full training data. Proponents, which is the default strategy in practice, outperform all groups except Full in all cases. Interestingly, some strategies result in a worse model than Random. In Section 3.5, we argue that some points inherently degrade model training when included (e.g. Minimum and Opponents).

| Model | Accuracy |
|---|---|
| Full | 0.32 |
| Proponents-25k | 0.30 |

Table 1: MMLU Average Accuracy (Zero-shot learning). All numbers are rounded-off to 2 decimal places.

Opponents. The similarity in behavior is explained by the distribution of influence values, which is left-skewed as shown in Figure 4. Thus, there is a significant overlap between the points selected for Minimum and those for Opponents. For how Minimum can lead to higher perplexity than Opponents, it may indicate that points with the least absolute influence changes the model's behavior the least compared to other strategies. In other words, it best retains the model's behavior before fine-tuning, which in our experiments is a base model that is trained to produce verbose output. A high perplexity here is then explained by the Minimum model producing longer text than the other models.

**MMLU Accuracy**. We further evaluate the two models of interest: Full (trained on the full dataset) and Proponents-25k (trained on the best 25k proponents) on the Massive Multitask Language Understanding (MMLU). MMLU tests language models for extensive world knowledge and problem solving capabilities in multiple-choice format (Hendrycks et al., 2020). We use the Gemma-2B models fine-tuned on the Ultrachat subset, which are the same models in the previous experiments. Table 1 reports the average accuracy across the different subjects as the fine-tuning was intended to improve general instruction-following.

Proponents-25k surprisingly performs similarly to Full for a wide range of tasks, and attains a similar accuracy as Full. There may be large differences in some subjects (around $10-25\%$), but there is similar performance in capabilities which those subjects are components of. For example, for *logical reasoning*, which is composed of subjects such as {*formal logic, elementary mathematics, logical fallacies*}, Full performs better in the first two subjects, but Proponents-25k performs better for the last subject. We postulate this is because our

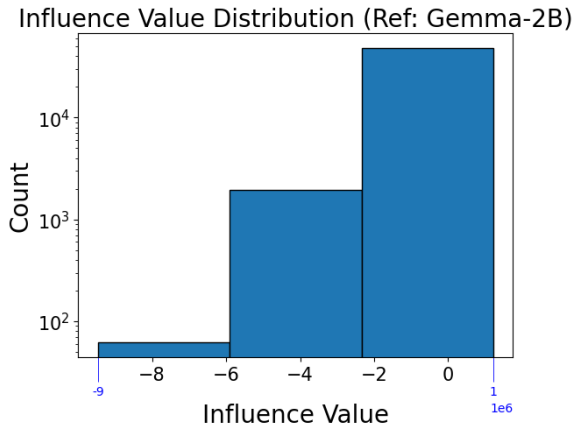



Figure 4: Histogram of influence values of the 50k training set using Gemma-2B as the reference model. The distribution is left-skewed, with the majority of influence values being negative. The Opponent and Minimum groups, (points with positive values and values smallest in influence value magnitude respectively) are located to the right side of the histogram and have high overlap (overlap coefficient = 0.59. Note that the numbers in blue denote the rounded value of the extremes of the distribution.

evaluation set represents samples for general capabilities. We expect In2Core can be used to improve specific capabilities if the evaluation set's distribution matches those capabilities. While it performs slightly worse, Proponents-25k being trained on half the data provides further evidence that additional data used by Full is only marginally useful.

### 3.5 Discussion

*More data can result in worse models.* For a given test set, some points are **inherently harmful** to include in the training data. This goes against the conventional idea that more data is better for transformer-based LLMs, given how their typical training (both pre-training and fine-tuning) is inherently data-intensive. Our work sheds light on the fact that not all points are of equal quality, and we can dramatically reduce the size of the training dataset if we only keep data that works towards our metric of interest. How specifically these Opponents, when added to the training dataset, can undo the influence of Proponents is not well-understood, but this may be related to the phenomenon of catastrophic forgetting (Goodfellow et al., 2013), where the abilities learned by a model from a given proponent may be forgotten when learning from an opponent/s.

We expect such harmful points to become increasingly common as synthetic data is increasingly used to fine-tune LLMs, where there is risk of model collapse from using poor-quality training data (Shumailov et al., 2023), or a training dataset that does not accurately reflect the distribution of the validation dataset. As such coreset selection will become an increasingly important method for most types of LLM training. Our algorithm can be incorporated as a pre-processing step for practitioners before fine-tuning.

*Influence values from smaller models transfer to larger models.* Surprisingly, we find that a smaller model (in this case almost 4x smaller than the model to be fine-tuned) can act as a reliable reference model, specifically for selecting what points to avoid adding in the training dataset. We expect that this transferring ability will remain as long as the reference model sufficiently learns the underlying training distribution after training. Importantly, this allows a further cost reduction of the two most expensive steps in our algorithm – namely the fine-tuning of the reference model on the full training dataset and the subsequent gradient collection.

*Influence values transfer across model architectures.* We show that our method applies to reference models with distinct model families, even those with completely different pre-training regimes. This implies that a practitioner can simply bypass training a reference model if an open-source version is available.

## 4 In2Core for Model Coverage

In this section, we use In2Core at test-time. Specifically, we use influence values to measure how "suitable" individual test points are to a final model trained on a given training set. We first describe the rationale and our method for analyzing model coverage. Then, we perform an experiment comparing our method to analyzing model coverage via semantic similarity. Finally, we discuss the implications of the experiment. Note that we use the Gemma-2B model fine-tuned on the Proponents dataset for our experiments, described in 3.4, and thus omit an Experimental Setup section.

Evaluating a trained instruction-tuned model is difficult considering the space of acceptable outputs given certain questions. Rather than solely looking at discrete outputs to determine if the model generalizes to a test point, we instead augment this analysis by also looking at the training set. We capture the influence of the entire training set by

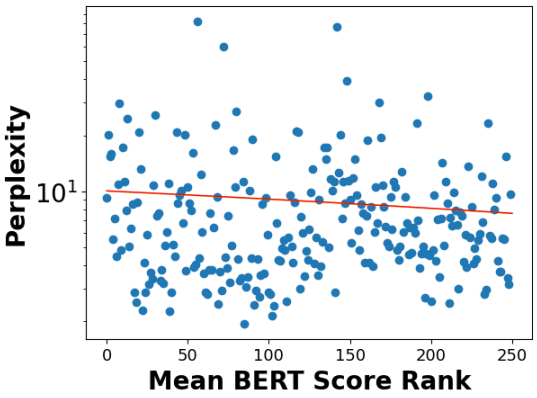
taking its average gradient, thereby giving a single influence value for each test point. This is akin to measuring how much a test point falls within the training distribution. To the best of our knowledge, this is the first time that influence values have been used in the literature to analyze how a given training set as a whole is appropriate for a given test point via a trained model. Importantly, this method is compatible with different evaluation metrics. In Section 4.1, we specifically apply this on the perplexity metric.

For instruction-tuned LLMs, having an indication on whether a given test input fits within its training distribution is essential because there is either an absence of a ground-truth to validate with or a ground-truth that is hard to articulate (e.g. a writing task where there is a space of acceptable outputs). We would like to emphasize that influence functions should not be used by themselves to evaluate a model's capability. It may be the case that the entire training set is beneficial to the model at learning a specific capability, but the training set size is insufficient for the model to learn that capability. Rather, influence functions provide further guidance at understanding model capability at evaluation by tying that capability to the training set, especially when one wishes to determine how to improve the model (as in coreset selection). Our method can be integrated in pipelines for model debugging (*which test points does the model need more data for?*) and handling inference requests (*which requests is the model not suited to perform well given its training?*) in production-grade LLMs.

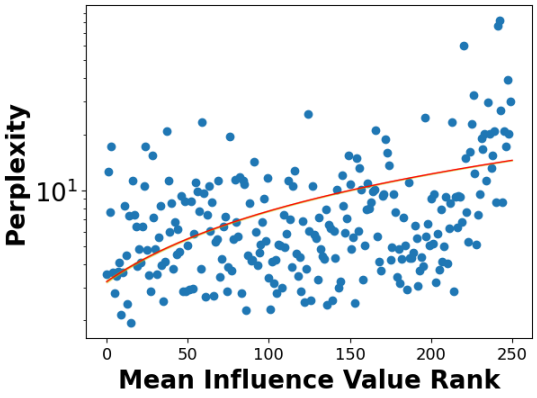
Furthermore, using influence functions in this manner can be computationally-efficient. We can exploit a feature of influence functions to our advantage: the most expensive computational step is calculating the gradients, but once the gradient computation is done, we can store them. In particular, we store the average gradient of the training dataset, and we can arbitrarily use it for future inference on unseen test points. While we still need to compute the model gradients with respect to new test points, the largest cost, coming from the training set, is only performed once.

### 4.1 Results

In Figure 5, we compare two different notions of measuring the importance of the entire training set to each evaluation point: Semantic similarity via




(a) Similarity of the Training Set on Evaluation Set




(b) Influence of the Training Set on Evaluation Set

Figure 5: Relationship between Perplexity and Measures of Importance of the Training Set on the Test Set (N=250). Influence values provide a better signal (correlation coefficient = 0.56) to indicate how well a model generalizes to a particular test point compared to semantic similarity (correlation coefficient = -0.087).

BERTScore (Zhang et al., 2019) and Influence via influence functions from DataInf. The BERTScore of two points is their cosine similarity in the contextual embedding space of BERT (Devlin et al., 2018). BERTScore aims to capture the semantic similarity between two different sequences of texts. In our experiments, we use DistilBERT (Sanh et al., 2019) for efficiency. For this case, we calculate the BERTScore of each training point to each test point. Then for each test point, we take the mean of the scores of all training points on that test point. We define this as the similarity score of the entire training set for that particular test point. Therefore for each test point, we obtain a scalar value as the similarity score. Meanwhile, to calculate the influence of the entire training set on each test point, we take the mean of the training point gradients to reduce it into a single point. We then calculate influence values with respect to individual test points as per

normal.

For both types of importance, we sort each test point into ascending order and plot the rank of each point. We compare these ranks to the perplexity of the model on the test point, where the model is the Gemma-2B fine-tuned on the entire training set. The red line for each graph denotes the linear regression line. Coefficient values are within a 95% confidence interval, but the intervals are too narrow to visualize. For semantic similarity as a metric, we hypothesize perplexity and semantic similarity to have a negative linear correlation (i.e. a downward trend) because test points with lower semantic similarity to the training set signifies that the test points are from a different distribution than the training set. However, while we observe this downward trend, it is a very weak correlation (coefficient = -0.087), suggesting that semantic similarity as measured by BERTScore is not a strong signal to indicate the fit of the training set to the test points.

In contrast, perplexity and influence values exhibit a stronger correlation. For Influence as a metric, we hypothesise perplexity and Influence to have a positive linear correlation (i.e. an upward trend) because test points assigned negative influence values imply that the training set is a proponent of those test points, and the training set is an opponent for the reverse case. We see this relationship between perplexity and influence, and observe that the correlation is stronger (coefficient = 0.56). This suggests that using influence values provide a stronger signal to indicate whether the model can generalize to a particular test point. Furthermore, using influence values is cheaper compared to using BERTScore, because we can reduce the training set to a single point when representing it as a gradient, compared to calculating point-wise similarities with BERTScore.

### 4.2 Discussion

*Influence functions and evaluation metrics are complementary and co-dependent at improving model capability.* We claim that these two measures are complementary because they provide an actionable feedback loop at improving the model. Evaluation metrics measure how close the model is behaving towards a desired behavior, and influence functions verifies if the training data does improve the model towards that desired behavior. Our coreset selection aglorithm is an example of how the two measures synergize instruction-tuning, which in our experiments focus on perplexity and influence. We claim that they are co-dependent because using these measurements on their own provide practitioners inadequate information on (a) whether the training data is sufficient at improving a capability, or (b) whether the additional training data will even contribute towards improving the capability, respectively.

*Measuring semantic similarity of training points is insufficient to capture a model's ability to generalize to a test point.* A model's ability to generalize to a test point is dependent on the model's loss on that test point. Semantic similarity, while providing an intuitive explanation for model generalization (a model learning from similar points will perform well on an unseen but similar point), does not completely capture different ways how a model learns from different training points to generalize to a test point. In particular, some points may not be semantically related to the test point, but they still serve to reduce the loss on that point. Since influence values are calculated with respect to the loss value (e.g. how much a given training point reduces the loss on an evaluation point/set), they are a closer metric to capture model generalization, and Section 4.1 demonstrates this empirically via the correlation coefficients.

## 5 Conclusion

In this work, we have shown that blindly adding data into the training set can *degrade* the model's performance. We developed In2Core, an algorithm for coreset selection based on influence functions. Our algorithm is practical and can be combined with other pre-processing steps to make supervised fine-tuning more efficient. We show that training on just half of the original training data can be comparable with the performance of a model trained on the full data for a variety of reference and target models. Importantly, (1) we observe a transfer effect where a reference model of a different architecture can be used to select the coreset data, and (2) this effect holds even with a reference model that is smaller than the model we are fine-tuning. Finally, we explore the use of influence to measure a trained model's coverage. We show that measuring data influence and an evaluation metric can aid in efficient model training than merely using an evaluation metric alone. We also show that this "influence" computed via influence functions captures a model's coverage better than by semantic

similarity.

## Limitations

**LLM Evaluation** There is ongoing work to evaluate instruction-following models on a variety of tasks (Hendrycks et al., 2020; Gao et al., 2021; Liang et al., 2022; Beeching et al., 2023; Wang et al., 2024a,b), but reaching a consensus on how to best evaluate such general-purpose models is still an open issue. Furthermore, there is an existing line of work that uses a larger model for automatic evaluation (Li et al., 2023a; Chiang et al., 2023; Zhao et al., 2023; Chen et al., 2023; Wang et al., 2024c), most popularly GPT-4 (Achiam et al., 2023). However, the decision of the larger model can be biased to models from the same family. Li et al. (2023a) Furthermore, the output is opaque and uninterpretable because there is no guarantee that the accompanying explanation to its answer is faithful (Jacovi and Goldberg, 2020).

**Group influence**. Our formulation of influence looks at the contribution of each points individually, but not their contribution as a group of points. We assumed that removing / adding groups of points based on their *individual* influences would create the expected effect (e.g. a sufficient learned model). Although we demonstrated this empirically in Section 3.4, we leave to future work the theoretical analysis if there are better methods to select points based on their individual or group influences.

As for selecting a group of points based on their group influences, we would like to stress that our algorithm could be applied for group influence by replacing DataInf with the group influence algorithm of choice. Calculating group influence with influence functions has first been studied by (Koh et al., 2019) and algorithms to calculate it are becoming more efficient (Basu et al., 2020).

**Average gradient of the tokens as the gradient of the entire sequence**. We take the average gradient of the tokens when calculating the gradient of each point to make influence value calculations tractable. Xia et al. (2024) showed that this method penalizes the influence of points with long sequences of text. This is a fundamental limitation for definitions of influence that involve calculating the gradient of sequences of varying lengths. See Table 7 for training examples selected by In2Core.

## 6 Ethics Statement

Training LLMs is compute-intensive and thus carbon emissions-heavy (Rillig et al., 2023). Our paper introduces a method to make LLM fine-tuning more efficient, which can lead to positive environmental impact. Using less data to fine-tune translates to less electricity and water consumption, all else being equal. We intend our method to be adopted by practitioners to reduce the cost of training. Our results are also transparent and derived from open-source LLMs & datasets.

## Acknowledgements

This research is supported by the National Research Foundation, Prime Minister's Office, Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) programme. The computational work for this article was partially performed on resources of the National Supercomputing Centre, Singapore (https://www.nscc.sg).

## References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Samyadeep Basu, Xuchen You, and Soheil Feizi. 2020. On second-order group influence functions for black-box predictions. In *International Conference on Machine Learning*, pages 715–724. PMLR.

Edward Beeching, Clémentine Fourrier, Nathan Habib, Sheon Han, Nathan Lambert, Nazneen Rajani, Omar Sanseviero, Lewis Tunstall, and Thomas Wolf. 2023. Open llm leaderboard. https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard.

Lichang Chen, Shiyang Li, Jun Yan, Hai Wang, Kalpa Gunaratna, Vikas Yadav, Zheng Tang, Vijay Srinivasan, Tianyi Zhou, Heng Huang, et al. 2023. Alpagasus: Training a better alpaca with fewer data. *arXiv preprint arXiv:2307.08701*.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, et al. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. *See https://vicuna. lmsys. org (accessed 14 April 2023)*, 2(3):6.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems.

R. Dennis Cook and Sanford Weisberg. 1980. Characterizations of an empirical influence function for detecting influential cases in regression. *Technometrics*, 22:495–508.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Zhi Zheng, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. 2023. Enhancing chat language models by scaling high-quality instructional conversations. *arXiv preprint arXiv:2305.14233*.

Leo Gao, Jonathan Tow, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Kyle McDonell, Niklas Muennighoff, et al. 2021. A framework for few-shot language model evaluation. *Version v0. 0.1. Sept*, page 8.

Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. 2013. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*.

Roger Grosse, Juhan Bae, Cem Anil, Nelson Elhage, Alex Tamkin, Amirhossein Tajdini, Benoit Steiner, Dustin Li, Esin Durmus, Ethan Perez, et al. 2023. Studying large language model generalization with influence functions. *arXiv preprint arXiv:2308.03296*.

Han Guo, Nazneen Fatema Rajani, Peter Hase, Mohit Bansal, and Caiming Xiong. 2020. Fastif: Scalable influence functions for efficient model interpretation and debugging. *arXiv preprint arXiv:2012.15781*.

Zayd Hammoudeh and Daniel Lowd. 2024. Training data influence analysis and estimation: A survey. *Machine Learning*, pages 1–53.

Frank R. Hampel. 1974. The influence curve and its role in robust estimation. *Journal of the American Statistical Association*, 69(346):383–393.

Xiaochuang Han, Daniel Simig, Todor Mihaylov, Yulia Tsvetkov, Asli Celikyilmaz, and Tianlu Wang. 2023. Understanding in-context learning via supportive pretraining data. *arXiv preprint arXiv:2306.15091*.

Bobby He and Thomas Hofmann. 2023. Simplifying transformer blocks. *arXiv preprint arXiv:2311.01906*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Alon Jacovi and Yoav Goldberg. 2020. Towards faithfully interpretable nlp systems: How should we define and evaluate faithfulness? *arXiv preprint arXiv:2004.03685*.

Fred Jelinek, Robert L Mercer, Lalit R Bahl, and James K Baker. 1977. Perplexity—a measure of the difficulty of speech recognition tasks. *The Journal of the Acoustical Society of America*, 62(S1):S63–S63.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Norm Jouppi, George Kurian, Sheng Li, Peter Ma, Rahul Nagarajan, Lifeng Nai, Nishant Patil, Suvinay Subramanian, Andy Swing, Brian Towles, et al. 2023. Tpu v4: An optically reconfigurable supercomputer for machine learning with hardware support for embeddings. In *Proceedings of the 50th Annual International Symposium on Computer Architecture*, pages 1–14.

Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pages 1885–1894. PMLR.

Pang Wei W Koh, Kai-Siang Ang, Hubert Teo, and Percy S Liang. 2019. On the accuracy of influence functions for measuring group effects. *Advances in neural information processing systems*, 32.

Yongchan Kwon, Eric Wu, Kevin Wu, and James Zou. 2023. Datainf: Efficiently estimating data influence in lora-tuned llms and diffusion models. *arXiv preprint arXiv:2310.00902*.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023a. Alpacaeval: An automatic evaluator of instruction-following models. `https://github.com/tatsu-lab/alpaca_eval`.

Yuanzhi Li, Sébastien Bubeck, Ronen Eldan, Allie Del Giorno, Suriya Gunasekar, and Yin Tat Lee. 2023b. Textbooks are all you need ii: phi-1.5 technical report. *arXiv preprint arXiv:2309.05463*.

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*.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, et al. 2022. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110*.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. Truthfulqa: Measuring how models mimic human falsehoods.

Yang Liu, Jiahuan Cao, Chongyu Liu, Kai Ding, and Lianwen Jin. 2024a. Datasets for large language models: A comprehensive survey. *arXiv preprint arXiv:2402.18041*.

Yiheng Liu, Hao He, Tianle Han, Xu Zhang, Mengyuan Liu, Jiaming Tian, Yutong Zhang, Jiaqi Wang, Xiaohui Gao, Tianyang Zhong, et al. 2024b. Understanding llms: A comprehensive overview from training to inference. *arXiv preprint arXiv:2401.02038*.

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.

Sung Min Park, Kristian Georgiev, Andrew Ilyas, Guillaume Leclerc, and Aleksander Madry. 2023. Trak: Attributing model behavior at scale. *arXiv preprint arXiv:2303.14186*.

Jeff M Phillips. 2017. Coresets and sketches. In *Handbook of discrete and computational geometry*, pages 1269–1288. Chapman and Hall/CRC.

Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. 2020. Estimating training data influence by tracing gradient descent. *Advances in Neural Information Processing Systems*, 33:19920–19930.

Matthias C Rillig, Marlene Ågerstrand, Mohan Bi, Kenneth A Gould, and Uli Sauerland. 2023. Risks and benefits of large language models for the environment. *Environmental Science & Technology*, 57(9):3464–3466.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. WINOGRANDE: an adversarial winograd schema challenge at scale.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Yarin Gal, Nicolas Papernot, and Ross Anderson. 2023. The curse of recursion: Training on generated data makes models forget. *arXiv preprint arXiv:2305.17493*.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

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*.

Bin Wang, Zhengyuan Liu, Xin Huang, Fangkai Jiao, Yang Ding, AiTi Aw, and Nancy Chen. 2024a. SeaEval for multilingual foundation models: From cross-lingual alignment to cultural reasoning. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 370–390, Mexico City, Mexico. Association for Computational Linguistics.

Bin Wang, Chengwei Wei, Zhengyuan Liu, Geyu Lin, and Nancy F Chen. 2024b. Resilience of large language models for noisy instructions. *arXiv preprint arXiv:2404.09754*.

Bin Wang, Xunlong Zou, Geyu Lin, Shuo Sun, Zhuohan Liu, Wenyu Zhang, Zhengyuan Liu, AiTi Aw, and Nancy F Chen. 2024c. Audiobench: A universal benchmark for audio large language models. *arXiv preprint arXiv:2406.16020*.

Xiao Wang, Weikang Zhou, Qi Zhang, Jie Zhou, Songyang Gao, Junzhe Wang, Menghan Zhang, Xiang Gao, Yunwen Chen, and Tao Gui. 2023. Farewell to aimless large-scale pretraining: Influential subset selection for language model. *arXiv preprint arXiv:2305.12816*.

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

Shuo Yang, Zeke Xie, Hanyu Peng, Min Xu, Mingming Sun, and Ping Li. 2022. Dataset pruning: Reducing training data by examining generalization influence. *arXiv preprint arXiv:2205.09329*.

Chih-Kuan Yeh, Ankur Taly, Mukund Sundararajan, Frederick Liu, and Pradeep Ravikumar. 2022. First is better than last for language data influence. *Advances in Neural Information Processing Systems*, 35:32285–32298.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence?

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

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

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*.

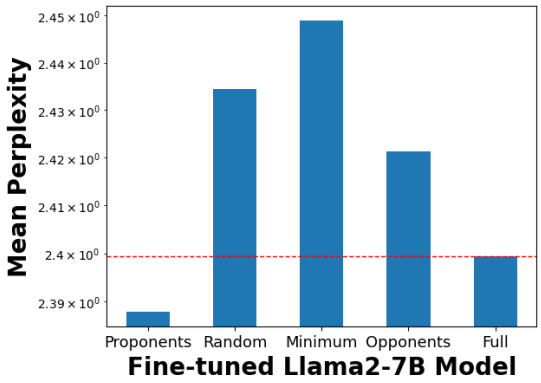
# A Appendix

## A.1 Experimental Hardware Details

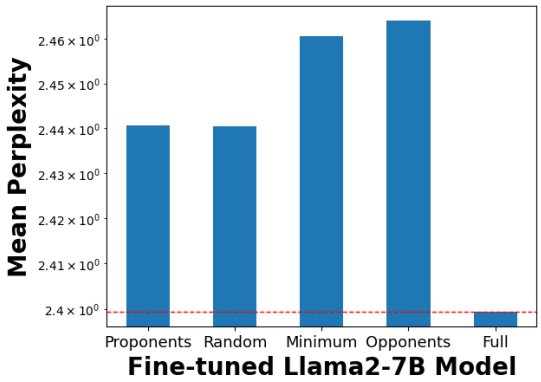
We used NVIDIA A100 Tensor Core GPUs. For influence values computation with the reference models, we used 1 GPU. For fine-tuning of models trained on the full dataset, we use 4 GPUs. For the rest of the fine-tuning setups, we used at most 2 GPUs. We estimate that we used 100 - 200 GPU hours on this project.

## A.2 Coreset selection for Llama2-7B

We conduct further experiments by using Llama2-7B (Touvron et al., 2023) as the target models to be fine-tuned. While the proponents version where Gemma-2B is the reference model performs better than the full version, the proponents version where a Llama2-7B is the reference model performs worse than the full version. The latter is likely limited by the number of layers used to calculate the influence values, because the highest spearman correlation we can attain given our hardware constraints was less than 50% for Llama2-7B.




(a) Performance of Models (Ref: Gemma-2B)




(b) Performance of Models (Ref: Mistral-7B)

Figure 6: Mean Perplexity of models fine-tuned on different coreset selection strategies on the 250-point Ultrachat Evaluation Set for Llama2-7B models. 'Full' denotes a model trained on the full training data.

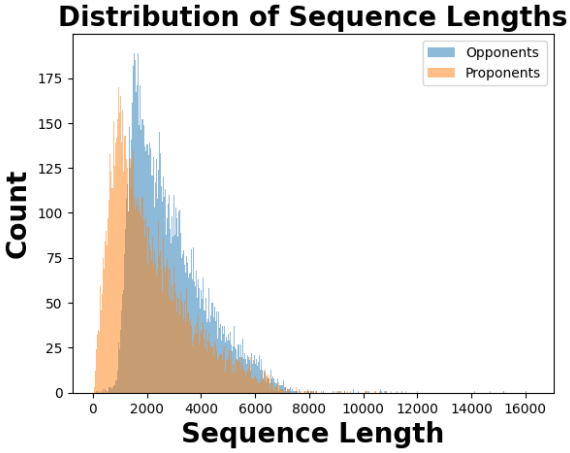



Figure 7: Influece functions are biased towards assigning higher scores to longer sequences. This stems from the fact that the value calculations are based on taking the average gradient of the token as the sequence gradient.