GREATS: Online Selection of High-Quality Data for LLM Training in *Every* Iteration

Jiachen T. Wang[⋆]
Princeton University
tianhaowang@princeton.edu

Tong Wu
Princeton University
tongwu@princeton.edu

Dawn Song
UC Berkeley
dawnsong@cs.berkeley.edu

Prateek Mittal
Princeton University
pmittal@princeton.edu

Ruoxi Jia[⋆]
Virginia Tech
ruoxijia@vt.edu

Abstract

Online batch selection methods offer an adaptive alternative to static training data selection by dynamically selecting data batches during training. However, existing methods either rely on impractical reference models or simple heuristics that may not capture true data informativeness. To address these limitations, we propose *GREedy Approximation Taylor Selection* (GREATS), a principled and efficient online batch selection method that applies greedy algorithm to optimize the data batch quality approximated by Taylor expansion. We develop a series of techniques to scale GREATS to large-scale model training. Extensive experiments with large language models (LLMs) demonstrate that GREATS significantly improves training convergence speed and generalization performance. Our codebase will be publically available at https://github.com/Jiachen-T-Wang/GREATS.

1 Introduction

Large language models (LLMs) are of paramount importance in today's technological landscape. However, the extensive training times, often spanning weeks or even months, pose challenges such as prolonged development cycles and increased resource consumption. Moreover, these models are trained on massive data collected from the open world, which can include low-quality, redundant, and biased information. This underscores the need for effective selection of high-value training data.

Online batch selection: an adaptive variant of data selection at the batch level. Online batch selection methods aim to improve data selection by dynamically choosing data during the training process. At each training iteration, these methods leverage the partially trained model to determine which data to select for the current training iteration from a sampled batch, thereby adapting to the model's learning progress and focusing on the most informative examples *for the model's current state*. In contrast to static data selection methods (e.g., [11, 43, 42, 45]), which select training data only once prior to the training process, online batch selection allows for a more adaptive and dynamic approach to data selection. By continuously updating the selection criteria based on the model's progress, online batch selection can identify the most relevant and informative examples at each stage of training, potentially leading to faster convergence and better generalization performance. Moreover, online batch selection operates on smaller batches of data, reducing the need for cumbersome data preprocessing and enabling more efficient use of computational resources compared to static data selection methods that process the entire dataset upfront.

[⋆]Correspondence to **Jiachen T. Wang** and **Ruoxi Jia**.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

However, existing online batch selection algorithms exhibit significant limitations and have found minimal success with LLMs. Reference-model-based batch selection methods [30, 9] rely on additional reference models. In [30], the reference models are trained on a substantial amount of hold-out data. This leads to considerable computational costs and reduces the amount of data available for training the main model. [9] utilize publicly available large-scale pre-trained models that already achieve very high performance on the targeted downstream tasks as reference models. Even if this assumption holds true in practice, the algorithm requires a Bayesian treatment and querying the reference models for every sample in the candidate batch at each iteration. These operations are computationally expensive, making the approach impractical for large-scale LLM training. Reference-model-free methods prioritize challenging samples based on metrics like high loss [28, 20] or large gradient norm [23]. While some of these methods are computationally efficient and practical to implement, they often rely on simple heuristics that may not capture the true informativeness or relevance of the examples. As a result, these methods often fall short in terms of performance and may even underperform compared to simple uniform selection in some cases. This highlights the need for more efficient and principled online batch selection techniques that can identify the most informative examples based on a deeper understanding of the model's learning dynamics and the relationships between the examples.

In this work, we propose *GREedy Approximation Taylor Selection* (GREATS), which addresses the limitations of existing methods and significantly improves the convergence speed and generalization performance of language model training. We summarize our contributions as follows.

I. Principled Formulation for Optimal Batch Selection. We introduce a principled formulation of the online batch selection problem as a set utility function optimization task. Given a small set of validation data from the target domain, the utility function measures the reduction in loss achieved by updating the model with a selected subset of the training batch. Unlike previous methods that rely on heuristics, this framework aims to directly optimize the model's performance on the validation set, ensuring the selection of informative and diverse examples.

II. Efficient Approximations for Scalable Batch Selection. The set function optimization formulation naturally leads to a greedy algorithm that iteratively selects the most informative examples based on their marginal contribution to the model's performance. However, directly applying the greedy algorithm to optimize the validation performance involves updating model with each potential candidate training point and checking the validation performance, which is computationally inefficient. To tackle this challenge, we propose to leverage Taylor expansions to approximate the variation of validation loss in one-step gradient descent. The key insight is that the impact of a training example on the model's validation loss can be efficiently approximated using gradient inner-products between the training examples and the validation data. This approximation eliminates the need for expensive model updates and validation loss evaluations for each candidate subset.

III. Online Batch Selection at the Speed of Regular Training. A direct implementation of GREATS would require computing per-sample gradients, which is computationally expensive. To address this challenge, we develop a novel technique called *"ghost inner-product"* that allows for the efficient computation of pairwise gradient inner-products without the need to instantiate any model-sized vectors. As gradient inner-products arise in various machine learning algorithms and applications beyond data selection, this technique may be of independent interest.

IV. Comprehensive Evaluations. We conduct extensive experiments on various language modeling tasks to thoroughly assess the performance of GREATS. We show that GREATS consistently speeds up training convergence and improves generalization performance across different models, training datasets, and evaluation datasets, even with a limited number of validation points. Furthermore, we show that GREATS can provide benefits even in the pretraining setting, where the validation data comes from the same domain as the training dataset. This highlights the robustness and versatility of our approach in various learning scenarios. In addition to its performance benefits, we empirically confirm that GREATS, equipped with the "ghost inner-product" technique, achieves a runtime comparable to regular training. This underscores the practical feasibility of our approach.

2 Related Works

Online Batch Selection. Few studies have investigated the use of online batch selection to enhance the training of models before the era of large language models. [28, 23, 20] examined selecting the

"hard examples" based on their gradient norm or maximum sample loss. While some of these methods are computationally efficient, they often depend on simple heuristics that cannot represent the true informativeness. [30, 9] suggested using additional reference models to more accurately estimate the importance of samples. However, recently [22] demonstrated these methods are computationally expensive and cannot directly apply to large language models.

Static Data Selection for Large-scale Models. Recently, there has been a growing interest in design methods to select data *before* training foundation models. We point the readers to [2] for a comprehensive literature review. These works select training data only *once*, prior to the training process. This is primarily motivated by efficiency concerns, as the time spent on data selection can be amortized over a large number of training steps. However, the non-adaptive nature of this single-step selection often results in suboptimal performance, as the selected data may not be the most informative or diverse throughout the entire training process [40]. Moreover, these algorithms often require extensive and complex data preprocessing steps. Some of the data selection algorithms even require training an additional model solely for the purpose of data selection [43, 44], which introduces additional computational costs and implementation complexity to the training pipeline. These drawbacks emphasize the need for more efficient and *adaptive* data selection techniques that can dynamically identify the most informative and relevant examples throughout the training process.

Online Domain Reweighting. Recent work has explored online methods for dynamically reweighting domains during language model pre-training. Compared with online batch selection, this approach operates at a coarser granularity by focusing on data source-level selection rather than individual examples, and typically updates domain weights less frequently. Similar to this work, [12] uses a gradient-based influence estimation to update domain weights. [3] uses training loss as a reward signal to adapt domain sampling probabilities. This has been recently refined by [21] through a scaling law-based method.

3 Background

In this section, we introduce the setup of online batch selection and the concept of a utility function. We then discuss the limitations of existing scoring and top-k paradigm in the literature.

Set-up of online batch selection. Given a training dataset $\mathcal{D}_{\text{tr}} = \{z_i\}_{i=1}^N$, a deep learning model is usually being trained to minimize the training loss $\sum_{i=1}^N \ell(w, z_i)$ via an iterative optimization procedure such as stochastic gradient descent (SGD). Starting with an initial model w_0, during an iteration t, a batch S of the training points is being used, and update the model parameters from w_t to w_{t+1} via $w_{t+1} := w_t - \eta_t \sum_{z \in S} \nabla \ell(w_t, z)$ where η_t is the learning rate at iteration t.[1] A complete run of neural network training thus consists of model checkpoints $\{w_0, w_1, \ldots, w_T\}$. In the setting of online batch selection, a large batch $\mathcal{B}_t = \{z_1, \ldots, z_B\}$ is being sampled from the training set $\mathcal{D}_{\text{tr}}$ at training iteration t. An online batch selection algorithm aims to select the most valuable subset $\widehat{\mathcal{B}}_t$ from $\mathcal{B}_t$. It can be naturally formulated as an optimization problem, where the objective is to maximize the utility of the selected $\widehat{\mathcal{B}}_t$ for model update. Here, we describe the existing online batch selection algorithms through the concept of a *utility function*.

Utility Function. At training iteration t, a *utility function* $U^{(t)}$ maps an input training data batch S to a score indicating the utility of this batch for the model update at iteration t. Specifically, for a given utility function U, the task of *online batch selection* over a candidate batch $\mathcal{B}_t$ is to identify the subset $\widehat{\mathcal{B}}_t \subseteq \mathcal{B}_t$ that optimizes:

$$\widehat{\mathcal{B}}_t^{(k)} = \underset{S \subseteq \mathcal{B}_t, |S|=k}{\operatorname{argmax}} \; U^{(t)}(S) \tag{1}$$

where k is a fixed budget of sample number $k < |\mathcal{B}_t|$ used to update the model. Since $U^{(t)}$ is a set function, solving Equation (1) presents significant challenges, as it may require evaluating the utility $U^{(t)}(S)$ for a large number of subsets $S \subseteq \mathcal{B}_t$. Existing online batch selection methods circumvent this issue through *"Scoring and Top-k Paradigm"*, which compute an importance score ϕ_z for each data point $z \in \mathcal{B}_t$ and then selecting the subset of data points with the highest importance scores. For example, [28, 20] use the individual loss on the training data point $\phi_z = \ell(w_t, z)$ as the importance

[1] Here we incorporate the normalization term $|S|$ into the learning rate η_t for notational simplicity.

score. [23] use the individual gradient norm $\phi_z = \|\nabla \ell(w_t, z)\|$ as the importance score. [30, 9] leverage a reference model and use the "reducible loss" as the importance score. The use of importance scores for online batch selection essentially defines the utility function $U^{(t)}(S) = \sum_{i \in S} \phi_i(U)$ and conjectures that the sum of individual data points' importance scores is a reliable indicator of a dataset S's utility, hoping for a positive correlation with the model w_t's performance at the $(t+1)$-th step after updating on S. Consequently, existing online batch selection strategies aim to maximize $\sum_{i \in S} \phi_i(U)$ by selecting the top-k data points with the highest importance scores.

Limitations of Scoring and Top-k Paradigm. Most scoring mechanisms for estimating the value of an individual data point $z \in \mathcal{B}_t$ result in similar data receiving similar scores. However, in the context of online batch selection, diversity is crucial. Consequently, a subset $\widehat{\mathcal{B}}_t$ consisting of the top-K valued data points may lack diversity. In particular, duplicate points might be scored equally high and be incorrectly assumed to doubly improve the model, though this is likely not the case. The primary issue with this top-K methodology is that it ignores the interactions among the selected data points. **When a data point is selected, the importance scores of the remaining data points will usually change.** For instance, the values of data points similar to the selected ones will typically decrease, while the values of data points that are very different from the selected ones will increase.

4 Optimizing Utility in Online Batch Selection via Greedy Algorithms

4.1 A Principled Utility Function for Online Batch Selection.

The performance of a model is typically measured through a set of validation points $\{z^{(\mathrm{val})}\}$. For a given validation data point $z^{(\mathrm{val})}$, an ideal utility function at a single iteration t is the reduction in validation loss:

$$U^{(t)}(S; z^{(\mathrm{val})}) := \ell(w_t, z^{(\mathrm{val})}) - \ell(\widetilde{w}_{t+1}(S), z^{(\mathrm{val})}) \tag{2}$$

where $\widetilde{w}_{t+1}(S) := w_t - \eta_t \sum_{z \in S} \nabla \ell(w_t, z)$ and $S \subseteq \mathcal{B}_t$ is the subset of the batch selected for model update. While this is a principled choice for an optimization objective in online batch selection, optimizing $U^{(t)}$ is computationally expensive, as it involves evaluating model updates with respect to combinatorially many subsets $S \subseteq \mathcal{B}_t$ (a total of $\binom{|\mathcal{B}_t|}{k}$ subsets).

Vanilla Greedy Algorithm. To address the challenge of evaluating the objective function for numerous subsets, the greedy optimization algorithm is widely used due to its effectiveness in set function optimization. The greedy algorithm iteratively selects the element that provides the largest *marginal gain* to the utility function, given the previously selected elements. Mathematically, when a utility function $U^{(t)}$ is given, the greedy algorithm selects data points $z \in \mathcal{B}_t$ one at a time. At each selection round, the algorithm selects the data point $z^* = \mathrm{argmax}_{z \in \mathcal{B}_t \setminus \widehat{\mathcal{B}}_t} U^{(t)}(\widehat{\mathcal{B}}_t \cup \{z\}) - U^{(t)}(\widehat{\mathcal{B}}_t)$. This process continues until k data points have been added to $\widehat{\mathcal{B}}_t$. The greedy algorithm is known to provide near-optimal solutions for monotone submodular set functions, with a famous $(1 - 1/e)$-approximation guarantee [31]. The greedy algorithm only requires $O(k|\mathcal{B}_t|)$ evaluations of $U^{(t)}$, a significant improvement over the $\binom{|\mathcal{B}_t|}{k}$ evaluations required by the brute-force method.

However, optimizing $U^{(t)}$ using the greedy algorithm is still not practical for online batch selection. Each evaluation of the utility function $U^{(t)}(S)$ in (2) involves computing aggregated gradients, updating the model, and calculating the validation loss, which can significantly increase the per-iteration cost of training. Since online batch selection algorithms run alongside the model training, these costs cannot be amortized across training runs as they would be with static dataset selection. This makes the greedy algorithm infeasible for real-time online batch selection.

4.2 An Efficient Greedy Algorithm for Utility Optimization without Utility Evaluation

Here, we develop an efficient approximation for the marginal gain of a data point $z \in \mathcal{B}_t \setminus \widehat{\mathcal{B}}_t$ to the utility of the already selected subset $\widehat{\mathcal{B}}_t$. For notational simplicity, we denote $\mathbf{g}_{(z_i)} := \nabla \ell(w_t, z_i)$ for all $z_i \in \mathcal{B}_t$. Given that the learning rate η_t in model training is typically small, a lower-order Taylor expansion often provides an accurate approximation for the change in loss during a single gradient

Algorithm 1 GREedy Approximation Taylor Selection (GREATS)

Input:
- `HessianApprox`: approximation method for Hessian matrix.

Initialize model w_0.
for $t = 0, \ldots, T-1$ **do**
  Sampled a random batch $\mathcal{B}_t$ from $\mathcal{D}_{\text{tr}}$.
  Initialize the importance score $\phi_z \leftarrow \eta_t \mathbf{g}_{(z)} \cdot \mathbf{g}_{(z^{(\text{val})})}$ for every $z \in \mathcal{B}_t$.
  Initialize the selected batch $\widehat{\mathcal{B}}_t \leftarrow \{\}$.
  for $r = 1, \ldots, k$ **do**
    $z^* \leftarrow \text{argmax}_{z \in \mathcal{B}_t \setminus \widehat{\mathcal{B}}_t} \phi_z$.
    Add z^* into $\widehat{\mathcal{B}}_t$.
    if `HessianApprox` = "exact" **then**
      Update the importance score $\phi_z \leftarrow \phi_z - \eta_t^2 \mathbf{g}_{(z)} \mathbf{H}_{(z^{(\text{val})})} \mathbf{g}_{(z^*)}$ for $z \in \mathcal{B}_t \setminus \widehat{\mathcal{B}}_t$.
    else if `HessianApprox` = "identity" **then**
      Update the importance score $\phi_z \leftarrow \phi_z - \eta_t^2 \mathbf{g}_{(z)} \cdot \mathbf{g}_{(z^*)}$ for $z \in \mathcal{B}_t \setminus \widehat{\mathcal{B}}_t$.
  Take one step of gradient update on the selected batch $\widehat{\mathcal{B}}_t$ and obtain w_{t+1}.

update, with approximation errors of $O(\eta_t^2)$ for first-order approximations.

$$\begin{aligned} U^{(t)}(z_i|\widehat{\mathcal{B}}_t) &:= U^{(t)}(\widehat{\mathcal{B}}_t \cup \{z_i\}) - U^{(t)}(\widehat{\mathcal{B}}_t) \\ &= \ell(\widetilde{w}_{t+1}(\widehat{\mathcal{B}}_t), z^{(\text{val})}) - \ell(\widetilde{w}_{t+1}(\widehat{\mathcal{B}}_t \cup \{z_i\}), z^{(\text{val})}) \\ &= \ell(\widetilde{w}_{t+1}(\widehat{\mathcal{B}}_t), z^{(\text{val})}) - \ell(\widetilde{w}_{t+1}(\widehat{\mathcal{B}}_t) - \eta_t \mathbf{g}_{(z_i)}, z^{(\text{val})}) \\ &\approx \eta_t \mathbf{g}_{(z_i)} \cdot \nabla \ell(\widetilde{w}_{t+1}(\widehat{\mathcal{B}}_t), z^{(\text{val})}) \end{aligned} \tag{3}$$

Interpretation. The first-order approximation of the marginal gain $U^{(t)}(z_i|\widehat{\mathcal{B}}_t)$ computes the inner-product between **(1)** the gradient of the individual training loss with respect to the original model w_t, and **(2)** the gradient of the validation loss with respect to the "virtual model" $\widetilde{w}_{t+1}(\widehat{\mathcal{B}}_t)$, i.e., w_t updated with the existing selected batch. The inner-product represents the direct influence of z_i on the validation loss at the "virtual model" $\widetilde{w}_{t+1}(\widehat{\mathcal{B}}_t)$. The gradient $\mathbf{g}_{(z_i)}$ is computed with respect to w_t instead of $\widetilde{w}_{t+1}(\widehat{\mathcal{B}}_t)$ because the model update process is performed using the gradients with respect to w_t. The approximation in (3) essentially estimates the improvement in validation loss by including z_i in the model update, assuming that $\widehat{\mathcal{B}}_t$ is already guaranteed to be included.

However, the computation of $\nabla \ell(\widetilde{w}_{t+1}(\widehat{\mathcal{B}}_t), z^{(\text{val})})$ still requires obtaining the updated model parameter $\widetilde{w}_{t+1}(\widehat{\mathcal{B}}_t)$ and performing additional backpropagations to compute the validation gradient, which again incurs significant computational overhead. To efficiently approximate it, we use another Taylor expansion as follows:

$$\nabla \ell(\widetilde{w}_{t+1}(\widehat{\mathcal{B}}_t) = \nabla \ell(w_t - \eta_t \sum_{z \in \widehat{\mathcal{B}}_t} \mathbf{g}_{(z)}, z^{(\text{val})}) \approx \mathbf{g}_{(z^{(\text{val})})} - \eta_t \mathbf{H}_{(z^{(\text{val})})} \sum_{z \in \widehat{\mathcal{B}}_t} \mathbf{g}_{(z)}$$

where the Hessian matrix $\mathbf{H}_{(z^{(\text{val})})} := \nabla^2 \ell(w_t, z^{(\text{val})})$, and $\mathbf{g}_{(z^{(\text{val})})} := \nabla \ell(w_t, z^{(\text{val})})$. Plugging this approximation back into (3), we have

$$U^{(t)}(z_i|\widehat{\mathcal{B}}_t) \approx \underbrace{\eta_t \mathbf{g}_{(z_i)} \cdot \mathbf{g}_{(z^{(\text{val})})}}_{\text{Importance score of } z_i} - \underbrace{\eta_t^2 \mathbf{g}_{(z_i)} \mathbf{H}_{(z^{(\text{val})})} \sum_{z \in \widehat{\mathcal{B}}_t} \mathbf{g}_{(z)}}_{\text{Correction of importance from already selected points}} \tag{4}$$

Interpretation. The first gradient inner-product $\eta_t \mathbf{g}_{(z_i)} \cdot \mathbf{g}_{(z^{(\text{val})})}$ coincides with the TracIN score proposed in [33] as a measure for a data point's importance. It captures the alignment between the gradient of the training loss for z_i and the gradient of the validation loss, indicating how much the update based on z_i would contribute to the reduction of the validation loss with respect to the original model w_t. The second term $-\eta_t^2 \mathbf{g}_{(z_i)} \mathbf{H}_{(z^{(\text{val})})} \sum_{z \in \widehat{\mathcal{B}}_t} \mathbf{g}_{(z)}$ is a correction term for z_i's original importance after picking $\widehat{\mathcal{B}}_t$. It penalizes the similarity between z_i and the data points in $\widehat{\mathcal{B}}_t$, as

measured by the Hessian-weighted inner-product of their gradients. Intuitively, if the gradient of z_i is similar to the gradients of the data points in $\widehat{\mathcal{B}}_t$, the correction term will be large, reducing the overall marginal gain of adding z_i to the selected subset. This encourages the selection of diverse data points that provide complementary information to the model update.

Algorithm. Using the approximation from (4), we develop a new algorithm that approximates the vanilla greedy algorithm. Initially, each data point $z \in \mathcal{B}_t$ is assigned an importance score ϕ_z initialized as $\phi_z = \eta_t \mathbf{g}_{(z)} \cdot \mathbf{g}_{(z^{(\mathrm{val})})}$, which approximates the marginal gain of adding z to an empty set, i.e., $U^{(t)}(z_i \mid \{\}) = U^{(t)}(\{z_i\})$. The algorithm begins by selecting the data point with the highest importance score, $z_1^* = \mathrm{argmax}_{z \in \mathcal{B}_t} \phi_z$. After selecting a data point z^* for model update, the importance scores for the remaining data points are adjusted by $-\eta_t^2 \mathbf{g}_{(z_i)} \mathbf{H}_{(z^{(\mathrm{val})})} \mathbf{g}_{(z^*)}$. This adjustment approximates the marginal gain of adding each remaining data point to the set containing z^*, i.e., $U^{(t)}(z_i \mid \{z^*\})$. The algorithm iteratively selects the data point with the highest adjusted importance score and updates the scores for the remaining points until k data points have been selected. As we can see, this iterative process closely mimics the behavior of the vanilla greedy algorithm while not requiring any actual evaluation of $U^{(t)}$, allowing for a computationally tractable approximation of the greedy algorithm in the context of online batch selection. The pseudocode for the proposed algorithm is detailed in Algorithm 1.

Validity of Taylor Approximation. We evaluate the fidelity of using Taylor expansion to approximate $U^{(t)}$. Following the experimental settings from our GPT2 experiments detailed in Appendix B, we sample different batch subsets S and evaluate $U^{(t)}(S)$ at the 3500th training iteration. Figure 1 illustrates the correlation between actual and predicted validation loss changes in a single gradient update step. Panel (a) shows the correlation when using only the first-order term (the sum of training gradients' dot products with the validation point) for loss change approximation. Panel (b) demonstrates the improved correlation when incorporating both the first-order term and the Hessian interaction term. The enhanced correlation coefficient with the inclusion of the Hessian term indicates that our approximations effectively capture the actual loss dynamics, with the second-order term providing substantial improvement in predictive accuracy.

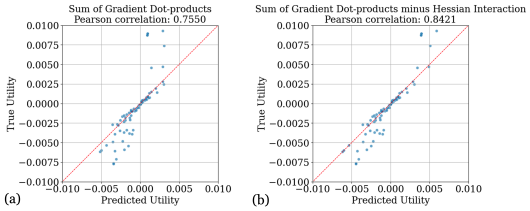

Figure 1: **(a)** We show the correlation between the ground-truth model validation loss change in one gradient update iteration $U^{(t)}(S; z^{(\mathrm{val})}) := \ell(w_t, z^{(\mathrm{val})}) - \ell(\widetilde{w}_{t+1}(S), z^{(\mathrm{val})})$ and the first-order Taylor approximation $\sum_{z \in S} \eta_t \mathbf{g}_{(z)} \cdot \mathbf{g}_{(z^{(\mathrm{val})})}$. **(b)** We show the correlation between $U^{(t)}(S; z^{(\mathrm{val})})$ and the first-order approximation corrected by the Hessian interaction $\sum_{z \in S} \eta_t \mathbf{g}_{(z)} \cdot \mathbf{g}_{(z^{(\mathrm{val})})} - \sum_{z, z' \in S} \eta_t^2 \mathbf{g}_{(z)} \mathbf{H}_{(z^{(\mathrm{val})})} \mathbf{g}_{(z')}$.

4.3 The Ghost Inner-Product Technique

Implementation Challenges of Algorithm 1. Although Algorithm 1 eliminates the need for explicit utility evaluations and relies solely on gradient and Hessian information, its efficient implementation remains a challenge. The initial importance scores $\eta_t \mathbf{g}_{(z)} \cdot \mathbf{g}_{(z^{(\mathrm{val})})}$ require computing the inner-products between the gradients of each $z \in \mathcal{B}_t$ and the validation point $z^{(\mathrm{val})}$. Directly implementing this would involve calculating the individual gradients for every data point in $\mathcal{B}_t$. This cannot leverage the parallel processing capabilities of GPUs and would require running backpropagation $|\mathcal{B}_t|$ times with a mini-batch size of 1, resulting in a significantly higher per-iteration runtime cost compared to regular training. Furthermore, the correction term $-\eta_t^2 \mathbf{g}_{(z)} \mathbf{H}_{(z^{(\mathrm{val})})} \mathbf{g}_{(z^*)}$ requires computing the gradient-Hessian-gradient product for each pair of $z, z' \in \mathcal{B}_t$. Even if we approximate the Hessian as the identity matrix ($\mathbf{H}_{(z^{(\mathrm{val})})} \approx \mathbf{I}$), calculating the pairwise $\mathbf{g}_{(z)} \cdot \mathbf{g}_{(z')}$ still necessitates either storing all individual gradient vectors $\{\mathbf{g}_{(z)}\}$ or recomputing $\mathbf{g}_{(z)}$ during each round of greedy selection. Both the memory and computational demands of these approaches are impractical for training large-scale models.

Efficient Computation of ALL Initial Importance Scores. To address the challenge of computing the gradient inner-products between all $z \in \mathcal{B}_t$ and $z^{(\mathrm{val})}$, we propose a novel technique called "ghost inner-product", which is inspired by the "ghost clipping" technique from the differential

privacy literature [5, 6]. The key idea behind "ghost inner-product" is to avoid explicitly computing individual gradient vectors, thereby improving the efficiency of the algorithm. To illustrate this technique, consider a simple linear layer $\mathbf{s} = \mathbf{a}\mathbf{W}$, where $\mathbf{W} \in \mathbb{R}^{d_1 \times d_2}$ is the weight matrix, $\mathbf{a} = (\mathbf{a}^{(1)}, \ldots, \mathbf{a}^{(B)})^\intercal \in \mathbb{R}^{B \times d_1}$ is the mini-batch input, and $\mathbf{s} = (\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(B)})^\intercal \in \mathbb{R}^{B \times d_2}$ is the output (i.e., the pre-activation tensor). Let $\ell^{(i)}$ denote the individual loss on z_i. By applying the chain rule, we can express the gradient of an individual loss $\ell^{(i)} := \ell(w, z_i)$ with respect to $\mathbf{W}$ as

$$\frac{\partial \ell^{(i)}}{\partial \mathbf{W}} = \frac{\partial \ell^{(i)}}{\partial \mathbf{s}^{(i)}} \frac{\partial \mathbf{s}^{(i)}}{\partial \mathbf{W}} = \frac{\partial \ell}{\partial \mathbf{s}^{(i)}} \mathbf{a}^{(i)} \tag{5}$$

where $\ell := \sum_{j=1}^{B} \ell^{(j)}$ is the aggregated loss. Note that the output gradient $\frac{\partial \ell}{\partial \mathbf{s}^{(i)}}$ is readily available during the backpropagation pass. To efficiently compute the gradient inner-product between a validation point and each training point, we include the validation data $z^{(\mathrm{val})}$ together in the batch for backpropagation. That is, we take the backpropagation on $\sum_{j=1}^{B} \ell^{(j)} + \ell^{(z^{(\mathrm{val})})}$. Hence, for each training-validation pair $(z_i, z^{(\mathrm{val})})$, we have the gradient inner-product

$$\frac{\partial \ell^{(i)}}{\partial \mathbf{W}} \odot \frac{\partial \ell^{(z^{(\mathrm{val})})}}{\partial \mathbf{W}} = \left(\left(\frac{\partial \ell}{\partial \mathbf{s}^{(i)}} \right)^\intercal \left(\frac{\partial \ell}{\partial \mathbf{s}^{(z^{(\mathrm{val})})}} \right) \right) \left(\left(\mathbf{a}^{(i)} \right)^\intercal \mathbf{a}^{(z^{(\mathrm{val})})} \right)$$

By using the "ghost inner-product" technique, we can compute the result without explicitly forming any full gradient vectors. Consequently, computing the gradient inner-product between every pair of training and validation points requires only one backpropagation, which is significantly more efficient than the direct method that would require $> |\mathcal{B}_t|$ backpropagations. We note that the "ghost inner-product" technique can be applied to various types of layers beyond linear layers. Similar decompositions as in Equation (5) have been studied in the differential privacy literature [36, 5, 26], enabling the extension of this technique to other layer types. Extension on LoRA is in Appendix A.

Efficient Approximation of Importance Correction. The importance correction term $\mathbf{g}_{(z)} \mathbf{H}_{(z^{(\mathrm{val})})} \mathbf{g}_{(z^*)}$ poses computational challenges due to the involvement of the Hessian matrix. A straightforward approximation is to assume $\mathbf{H}_{(z^{(\mathrm{val})})} \approx \mathbf{I}$, simplifying the problem to computing the gradient inner-product $\mathbf{g}_{(z)} \cdot \mathbf{g}_{(z^*)}$. This approximation has been widely used in the literature, particularly in the context of second-order optimization methods and meta-learning [29, 13, 32]. The key motivation behind this approximation is that the Hessian matrix is often diagonally dominant, especially when the model is close to a local minimum [4]. By assuming $\mathbf{H}_{(z^{(\mathrm{val})})} \approx \mathbf{I}$, the importance correction term simplifies to $\phi_z - \eta_t^2 \mathbf{g}_{(z)} \cdot \mathbf{g}_{(z^*)}$. We can then apply the ghost inner-product technique previously developed for computing pairwise gradient inner-products. This allows us to efficiently compute the importance correction term without explicitly forming the individual gradient vectors or the Hessian-vector products.

Merging Batch Selection and Gradient Update in One Backpropagation. Utilizing the techniques developed in this section, we can calculate or approximate all importance scores and correction terms in a single backpropagation pass, without the need to materialize any model-sized vectors. Although computing the gradient of the aggregated training loss $\sum_{z_i \in \mathcal{B}_t} \ell^{(i)}$ for the training batch is necessary for parameter updates, an additional backpropagation pass is not required. By retaining the activations and output gradients from the previous backpropagation, we can efficiently compute this gradient without incurring the cost of another pass (see Appendix A.3 for details). Consequently, the process of training with batch selection introduces minimal additional runtime overhead. This approach provides substantial benefits over the direct method of materializing per-sample model-sized gradients, making it more feasible for real-world applications.

5 Experiments

In this section, we first evaluate the performance of GREATS against several baselines on a diverse set of models, training datasets, and validation set configurations. We then empirically examine its computational efficiency when implemented with "ghost inner-product" technique from Section 4.3.

5.1 Experimental Setup

Model-Training-Evaluation Pairs. We examine multiple combinations of models, training datasets, and evaluation datasets to evaluate our proposed GREATS algorithm, as shown in Table 1. Specifically,

Table 1: Combination of models, training datasets, and evaluation datasets

Task	Model	Training Dataset	Evaluation Dataset	Number of validation data
Fine-tuning	LLAMA-2-7B [38]	LESS [43]	MMLU [17]	5
Fine-tuning	MISTRAL-7B [19]	LESS [43]	TYDIQA [7]	10
Fine-tuning	LLAMA-3-8B [1]	ALPACA [37]	SAMSUM [14]	16
Pretraining	GPT-SMALL [34]	OPENWEBTEXT [15]	OPENWEBTEXT [15]	16

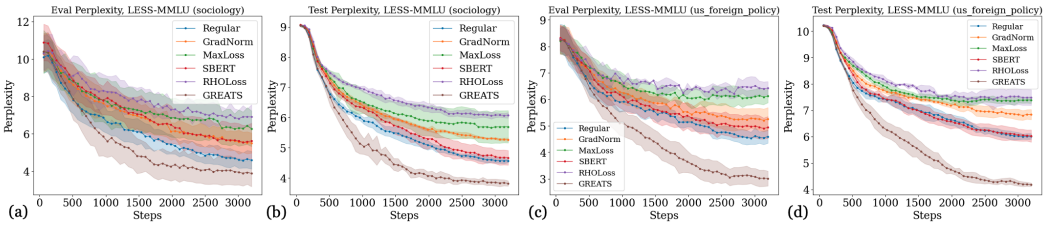

Figure 2: Comparison of the validation and test perplexity dynamics during training for different online batch selection methods on MMLU. We select sociology and US foreign policy subjects.

we fine-tune three large language models (LLMs): LLAMA-2-7B [38], MISTRAL-7B [19], and LLAMA-3-8B [1] using LESS training data [43] and the ALPACA dataset [37]. For evaluation, we employ the MMLU [17], TYDIQA [7], and SAMSUM [14] datasets (deferred to Appendix C). Additionally, we conduct a pretraining experiment using the GPT-SMALL model [34]. For both training and evaluation, we use the OPENWEBTEXT dataset [15]. In all experiments, we limited the validation data to be small (i.e., ≤ 16) to mimic practical scenarios where training directly on them is impossible.

Baselines. We compare our algorithm with regular training and a variety of online batch selection algorithms: **(1)** `MaxLoss` [28], which selects training data points with the highest loss values. **(2)** `GradNorm` [23], which prioritizes training data points with the highest gradient norms. **(3)** Reference model-based method (`RHOLoss`), for which we implement the RHO-Loss algorithm from [30] as a representative baseline. Given computational constraints, we use Llama-3.1-8B-Instruct [10] as the reference model, paired with Llama2-7B as the target model.[2] **(4)** Distance-based method (`SBERT`), which selects training batches based on their semantic similarity to validation data using Sentence-BERT embeddings [35].

Training Details. For all batch selection methods, we select 50% of the batch data for gradient updates during each step. In contrast, the regular training baseline performs updates on the entire batch, utilizing *twice* as much data as the batch selection methods. In the main paper, we show the results of setting the batch size to 4 for MMLU and TYDIQA, 16 for SAMSUM and OPENWEBTEXT. Additional training details and ablation studies are provided in Appendix B.

5.2 Performance Evaluation

In this section, we present and discuss the comparison between GREATS and the baseline algorithms in model training performance across different models, training, and evaluation datasets. We evaluate the performance on both the validation set (being used for batch selection in GREATS) and a test set that is drawn from the same domain of the evaluation datasets.

GREATS significantly speeds up training convergence. In Figure 2 and 4, we show the dynamics of perplexity on the validation and test datasets. As we can see from Figure 2 and Figure 4, across all settings, the GREATS algorithm achieves a significantly faster reduction in test perplexity compared to all of the baselines and often achieves better overall performance.

[2]We note that while larger pretrained models could serve as reference models, their computational costs increase substantially, particularly due to the need for repeated queries at each training iteration. Furthermore, in specialized domains, general-purpose pretrained models may exhibit suboptimal performance, necessitating fine-tuning on substantial holdout datasets. This additional requirement becomes impractical when computational efficiency and data availability are crucial considerations. We thank the anonymous NeurIPS reviewer for their helpful suggestion regarding the reference model selection in our comparisons.

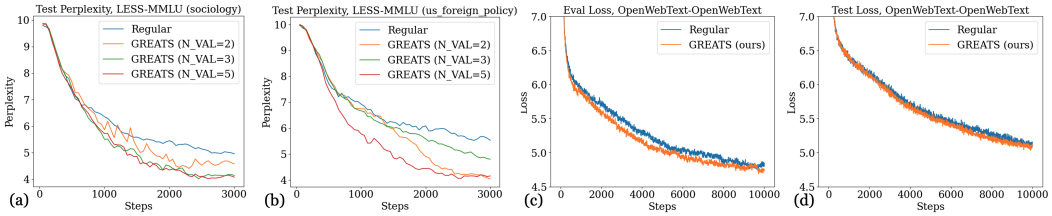

Figure 3: (a)-(b): Impact of the number of validation data points on the performance of GREATS. (c)-(d): Comparison of the validation and test perplexity dynamics during GPT2 pretraining for different online batch selection methods.

Method	MMLU (AVG.)	Soc.	Pol.	Alg.	Anat.	Astr.	Eth.	Clin.	Bio.	Chem.	TYDIQA
Regular	50.4%	62%	60%	40%	52%	48%	52%	**54%**	48%	38%	54.3%
GradNorm	50.4%	62%	62%	38%	50%	48%	54%	52%	**50%**	38%	53.4%
MaxLoss	50.8%	64%	58%	40%	52%	46%	54%	**54%**	**50%**	40%	54.7%
Ours	**54.2%**	**68%**	**64%**	**44%**	**56%**	**52%**	**56%**	**54%**	**50%**	**44%**	**55.0%**

Table 2: Accuracy on MMLU (9 subjects) and TYDIQA test set for online batch selection methods.

These results demonstrate the robust effectiveness of our approach in improving model convergence speed and generalization performance across various settings. We note that the validation-free approaches such as `GradNorm` and `MaxLoss` may lead to the selection of low-quality data. While it is generally considered that data points with high training loss or large gradients are important to learning, there is another possibility that the data points that achieve these properties are corrupted data which is not learnable.

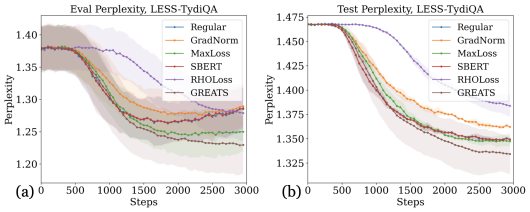

Figure 4: Comparison of the validation and test perplexity dynamics during training for different online batch selection methods on TYDIQA.

GREATS improves performance on downstream tasks. While perplexity is a direct measure of the performance of NLP models, it is not very interpretable, and the performance is often evaluated in terms of downstream tasks. In Table 2, we show the test accuracy on 9 (randomly selected) subjects from MMLU and TYDIQA. As we can see, GREATS consistently outperforms or at least achieves the same accuracy as the baselines. Notably, we observe at least a 3.4% improvement in the average performance of the 9 MMLU subjects compared to all baselines.

GREATS is robust to the number of validation points. In Figure 3 (a)-(b), we conduct an ablation study to evaluate the impact of the number of validation points used for GREATS algorithm. As we can see, even with just 2 validation examples, the test perplexity on the MMLU dataset is consistently lower than that of regular training. This robustness may be attributed to the relatively rare format of the validation corpus, which allows GREATS to effectively select examples from the batch that can help learn the particular format. Even if such a selection may overfit the specific validation examples, the selected batch can still improve the performance on the test examples, demonstrating the effectiveness of the GREATS algorithm in adapting to the characteristics of the validation set.

GREATS also improves LLM pretraining. In Figure 3 (c)-(d), we evaluate the performance of GREATS on pretraining GPT-SMALL on OPENWEBTEXT. Due to computational resource constraints, we omit the baselines of `GradNorm` and `MaxLoss`, as they have been shown to be ineffective in all other experiments. As we can see, even for model pretraining, GREATS provides an improvement in test performance, although the improvement is marginal compared to the gains observed in the fine-tuning experiments. This marginal improvement can be attributed to the fact that the validation data used in this experiment is also drawn from the same distribution as the training set, i.e., OPENWEBTEXT. As a result, the selected batches may not provide as much additional information or diversity as in the case of fine-tuning, where the validation data often comes from a different distribution or focuses on specific tasks. Nevertheless, the consistent improvement in test performance suggests that GREATS can still identify informative examples that contribute to better model generalization, even in the pretraining setting.

5.3 Runtime Comparison

We compare the runtime efficiency of GREATS algorithm with "ghost inner-product" technique against GREATS implemented directly by calculating per-sample gradients. The runtime is measured by training GPT-SMALL on OPENWEBTEXT. Additionally, we compare against `GradNorm`'s direct implementation using per-sample gradients, as it is the most similar algorithm to GREATS and allows for a fair comparison.

GREATS with "ghost inner-product" achieves runtime close to regular training. As shown in Table 3, the runtime of GREATS using our efficient approximation techniques is comparable to that of regular training, with only a slight increase in runtime due to pair-wise inner-product operations (but almost negligible compared with model backpropagation). This demonstrates the effectiveness of our approximation methods in reducing the computational overhead associated with online batch selection. On the other hand, the direct implementation of GREATS, which requires computing per-sample gradients, exhibits a significant runtime increase compared to both regular training and our efficient GREATS implementation. The direct approach is significantly slower than regular training (almost 20 times slower), making it impractical for real-world applications. The runtime of `GradNorm` with direct per-sample gradient computation falls between our efficient GREATS implementation and the direct GREATS implementation. This is because `GradNorm` does not need to compute the per-sample gradients from the validation set, which reduces its computational overhead compared to the direct GREATS implementation. However, `GradNorm` still incurs a significant runtime increase compared to regular training due to the per-sample gradient calculation for the training batch. We remark that the "ghost norm" technique from differential privacy literature [16, 25, 5, 6], which is similar to "ghost inner-product", can be used to improve the runtime of `GradNorm`, potentially bringing it closer to regular training.

	Throughput
Regular Training	76.2
GREATS (ghost)	71.3
GREATS (direct)	4.2
GradNorm (direct)	6.8

Table 3: Efficiency comparison of different implementations of GREATS. We use throughput-# training data points being processed per second-as the efficiency metric.

6 Conclusion and Limitations

In this work, we introduced GREATS, a novel online batch selection algorithm designed to enhance the efficiency and effectiveness of training large language models. Here, we briefly summarize the limitations of this work.

I. Availability of validation data. One potential limitation of GREATS is that it requires the validation data to be available before training. We stress that there are many scenarios where the validation data is naturally available before training such as fine-tuning or domain adaptation. Developing a validation-free variant of GREATS is an interesting future work.

II. Extension to Adam. The ghost inner-product technique developed in this work is specifically tailored for Stochastic Gradient Descent (SGD). It is not directly extendable to other popular optimizers like Adam due to their normalization terms. Nonetheless, using SGD as a proxy for Adam has proved to be effective in our experiment. Extending our ghost inner-product technique to Adam and similar optimizers remains an exciting direction for future research.

III. Memory constraint for large batch sizes. In scenarios where GPU memory constraints prevent adding validation data to the training batch for backpropagation, we can easily extend our "ghost" techniques by using gradient accumulation. However, this may increase runtime due to additional backpropagation steps for validation data, it maintains the feasibility of our techniques under memory constraints. Improving computational efficiency for large batch sizes remains an important direction for future research.

IV. Perplexity may not be an ideal objective. In this work, the utility function is being defined as the validation loss. While GREATS achieves promising results overall in terms of test perplexity, we note that perplexity may not reflect the performance in downstream tasks. While GREATS usually achieves higher performance on the downstream task, the improvement is often minor. Directly optimizing in terms of the downstream performances is another important future work.

Acknowledgment

This work is supported in part by the National Science Foundation under grants IIS-2312794, IIS-2313130, OAC-2239622, CNS-2131938, Amazon-Virginia Tech Initiative in Efficient and Robust Machine Learning, the Commonwealth Cyber Initiative, OpenAI and Google.

We thank Ashwinee Panda, Xinyu Tang and Yiding Jiang for their helpful feedback on the preliminary version of this work. We thank anonymous NeurIPS reviewers for the helpful feedback and discussion of this work.

References

[1] AI@Meta. Llama 3 model card. 2024.

[2] Alon Albalak, Yanai Elazar, Sang Michael Xie, Shayne Longpre, Nathan Lambert, Xinyi Wang, Niklas Muennighoff, Bairu Hou, Liangming Pan, Haewon Jeong, Colin Raffel, Shiyu Chang, Tatsunori Hashimoto, and William Yang Wang. A survey on data selection for language models, 2024.

[3] Alon Albalak, Liangming Pan, Colin Raffel, and William Yang Wang. Efficient online data mixing for language model pre-training. *arXiv preprint arXiv:2312.02406*, 2023.

[4] Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *SIAM review*, 60(2):223–311, 2018.

[5] Zhiqi Bu, Jialin Mao, and Shiyun Xu. Scalable and efficient training of large convolutional neural networks with differential privacy. *Advances in Neural Information Processing Systems*, 35:38305–38318, 2022.

[6] Zhiqi Bu, Yu-Xiang Wang, Sheng Zha, and George Karypis. Differentially private optimization on large model at small cost. In *International Conference on Machine Learning*, pages 3192–3218. PMLR, 2023.

[7] Jonathan H. Clark, Eunsol Choi, Michael Collins, Dan Garrette, Tom Kwiatkowski, Vitaly Nikolaev, and Jennimaria Palomaki. Tydi qa: A benchmark for information-seeking question answering in typologically diverse languages. *Transactions of the Association for Computational Linguistics*, 2020.

[8] Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. Free dolly: Introducing the world's first truly open instruction-tuned llm, 2023.

[9] Zhijie Deng, Peng Cui, and Jun Zhu. Towards accelerated model training via bayesian data selection. *Advances in Neural Information Processing Systems*, 36, 2024.

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

[11] Logan Engstrom, Axel Feldmann, and Aleksander Madry. Dsdm: Model-aware dataset selection with datamodels. *arXiv preprint arXiv:2401.12926*, 2024.

[12] Simin Fan, Matteo Pagliardini, and Martin Jaggi. Doge: Domain reweighting with generalization estimation. In *Forty-first International Conference on Machine Learning*, 2024.

[13] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

[14] Bogdan Gliwa, Iwona Mochol, Maciej Biesek, and Aleksander Wawer. SAMSum corpus: A human-annotated dialogue dataset for abstractive summarization. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pages 70–79, Hong Kong, China, November 2019. Association for Computational Linguistics.

[15] Aaron Gokaslan and Vanya Cohen. Openwebtext corpus. http://Skylion007.github.io/OpenWebTextCorpus, 2019.

[16] Ian Goodfellow. Efficient per-example gradient computations. *arXiv preprint arXiv:1510.01799*, 2015.

[17] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2020.

[18] Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2021.

[19] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023.

[20] Angela H Jiang, Daniel L-K Wong, Giulio Zhou, David G Andersen, Jeffrey Dean, Gregory R Ganger, Gauri Joshi, Michael Kaminksy, Michael Kozuch, Zachary C Lipton, et al. Accelerating deep learning by focusing on the biggest losers. *arXiv preprint arXiv:1910.00762*, 2019.

[21] Yiding Jiang, Allan Zhou, Zhili Feng, Sadhika Malladi, and J Zico Kolter. Adaptive data optimization: Dynamic sample selection with scaling laws. *arXiv preprint arXiv:2410.11820*, 2024.

[22] Jean Kaddour, Oscar Key, Piotr Nawrot, Pasquale Minervini, and Matt Kusner. No train no gain: Revisiting efficient training algorithms for transformer-based language models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[23] Angelos Katharopoulos and François Fleuret. Not all samples are created equal: Deep learning with importance sampling. In *International conference on machine learning*, pages 2525–2534. PMLR, 2018.

[24] Andreas Köpf, Yannic Kilcher, Dimitri von Rütte, Sotiris Anagnostidis, Zhi-Rui Tam, Keith Stevens, Abdullah Barhoum, Nguyen Minh Duc, Oliver Stanley, Richárd Nagyfi, et al. Openassistant conversations–democratizing large language model alignment. *arXiv preprint arXiv:2304.07327*, 2023.

[25] Jaewoo Lee and Daniel Kifer. Scaling up differentially private deep learning with fast per-example gradient clipping. *Proceedings on Privacy Enhancing Technologies*, 2021.

[26] Xuechen Li, Florian Tramer, Percy Liang, and Tatsunori Hashimoto. Large language models can be strong differentially private learners. In *International Conference on Learning Representations*, 2021.

[27] Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V Le, Barret Zoph, Jason Wei, et al. The flan collection: Designing data and methods for effective instruction tuning. *arXiv preprint arXiv:2301.13688*, 2023.

[28] Ilya Loshchilov and Frank Hutter. Online batch selection for faster training of neural networks. *arXiv preprint arXiv:1511.06343*, 2015.

[29] James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International conference on machine learning*, pages 2408–2417. PMLR, 2015.

[30] Sören Mindermann, Jan M Brauner, Muhammed T Razzak, Mrinank Sharma, Andreas Kirsch, Winnie Xu, Benedikt Höltgen, Aidan N Gomez, Adrien Morisot, Sebastian Farquhar, et al. Prioritized training on points that are learnable, worth learning, and not yet learnt. In *International Conference on Machine Learning*, pages 15630–15649. PMLR, 2022.

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

[32] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

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

[34] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[35] N Reimers. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

[36] Gaspar Rochette, Andre Manoel, and Eric W Tramel. Efficient per-example gradient computations in convolutional neural networks. In *Workshop on Theory and Practice of Differential Privacy (TPDP)*, 2020.

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

[38] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[39] Jiachen T Wang, Prateek Mittal, Dawn Song, and Ruoxi Jia. Data shapley in one training run. *arXiv preprint arXiv:2406.11011*, 2024.

[40] Jiachen T Wang, Tianji Yang, James Zou, Yongchan Kwon, and Ruoxi Jia. Rethinking data shapley for data selection tasks: Misleads and merits. *arXiv preprint arXiv:2405.03875*, 2024.

[41] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.

[42] Alexander Wettig, Aatmik Gupta, Saumya Malik, and Danqi Chen. Qurating: Selecting high-quality data for training language models. *arXiv preprint arXiv:2402.09739*, 2024.

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

[44] Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifeng Lu, Percy S Liang, Quoc V Le, Tengyu Ma, and Adams Wei Yu. Doremi: Optimizing data mixtures speeds up language model pretraining. *Advances in Neural Information Processing Systems*, 36, 2024.

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

A Ghost Inner-Product

Notation Review. Consider a linear layer $\mathbf{s} = \mathbf{a}\mathbf{W}$, where $\mathbf{W} \in \mathbb{R}^{d_1 \times d_2}$ is the weight matrix, $\mathbf{a} = (\mathbf{a}^{(1)}, \ldots, \mathbf{a}^{(B)})^\intercal$ is the mini-batch input, and $\mathbf{s} = (\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(B)})^\intercal$ is the output (i.e., the pre-activation tensor). For non-sequential data, $\mathbf{a} \in \mathbb{R}^{B \times d_1}, \mathbf{s} \in \mathbb{R}^{B \times d_2}$. For sequential data with sequence length T, $\mathbf{a} \in \mathbb{R}^{B \times d_1 \times T}, \mathbf{s} \in \mathbb{R}^{B \times d_2 \times T}$. Let $\ell^{(i)} := \ell(w, z_i)$ denote the current model's individual loss on z_i. For notation convenience, we denote $\mathbf{b}^{(i)} := \frac{\partial \ell^{(i)}}{\partial \mathbf{s}}$.

A.1 Ghost Inner-Product for Linear Layers

By applying the chain rule, we can express the gradient of an individual loss $\ell^{(i)} := \ell(w, z_i)$ with respect to $\mathbf{W}$ as

$$\frac{\partial \ell^{(i)}}{\partial \mathbf{W}} = \frac{\partial \ell^{(i)}}{\partial \mathbf{s}^{(i)}} \frac{\partial \mathbf{s}^{(i)}}{\partial \mathbf{W}} = \frac{\partial \ell}{\partial \mathbf{s}^{(i)}} \mathbf{a}^{(i)} = \mathbf{a}^{(i)} \otimes \mathbf{b}^{(i)} \tag{6}$$

where $\ell := \sum_{j=1}^{B} \ell^{(j)}$ is the aggregated loss. Note that the individual's output gradient $\mathbf{b}^{(i)} = \frac{\partial \ell^{(i)}}{\partial \mathbf{s}^{(i)}} = \frac{\partial \ell}{\partial \mathbf{s}^{(i)}}$ is readily available during the backpropagation pass. Suppose we are interested in computing the gradient inner-product $\frac{\partial \ell^{(1)}}{\partial \mathbf{W}} \odot \frac{\partial \ell^{(2)}}{\partial \mathbf{W}}$ between two data points z_1, z_2 in the same batch in the backpropagation.

Non-sequential data. For non-sequential data, we have each $\mathbf{a}^{(i)} \in \mathbb{R}^{d_1 \times 1}$ and $\mathbf{b}^{(i)} \in \mathbb{R}^{1 \times d_2}$. By (6), we have

$$\frac{\partial \ell^{(1)}}{\partial \mathbf{W}} \odot \frac{\partial \ell^{(2)}}{\partial \mathbf{W}} = \left(\mathbf{a}^{(1)} \otimes \mathbf{b}^{(1)}\right) \odot \left(\mathbf{a}^{(2)} \otimes \mathbf{b}^{(2)}\right) = \left(\left(\mathbf{b}^{(1)}\right)^\intercal \left(\mathbf{b}^{(2)}\right)\right) \left(\left(\mathbf{a}^{(1)}\right)^\intercal \mathbf{a}^{(2)}\right)$$

Hence, we can compute the inner-product between $\frac{\partial \ell^{(1)}}{\partial \mathbf{W}}$ and $\frac{\partial \ell^{(2)}}{\partial \mathbf{W}}$ without actually instantiating the gradient vector $\frac{\partial \ell^{(1)}}{\partial \mathbf{W}}$ or $\frac{\partial \ell^{(2)}}{\partial \mathbf{W}}$. We can take the inner products between $\left(\mathbf{b}^{(1)}\right)^\intercal \left(\mathbf{b}^{(2)}\right)$ and $\left(\mathbf{a}^{(1)}\right)^\intercal \mathbf{a}^{(2)}$, and then multiply the results together. Moreover, all of the materials $\mathbf{a}^{(1)}, \mathbf{a}^{(2)}, \mathbf{b}^{(1)}, \mathbf{b}^{(2)}$ that are required for computation are all already available in one backpropagation. Hence, with a single backpropagation, we can efficient compute the gradient inner-product between every pair of the data points within the batch.

Sequential data. For sequential data, we have each $\mathbf{a}^{(i)} \in \mathbb{R}^{d_1 \times T}$ and $\mathbf{b}^{(i)} \in \mathbb{R}^{T \times d_2}$. By (6), we have

$$\begin{aligned}
\frac{\partial \ell^{(1)}}{\partial \mathbf{W}} \odot \frac{\partial \ell^{(2)}}{\partial \mathbf{W}} = \left(\mathbf{a}^{(1)} \otimes \mathbf{b}^{(1)}\right) \odot \left(\mathbf{a}^{(2)} \otimes \mathbf{b}^{(2)}\right) &= \sum_{j,k=1}^{d_1,d_2} \left(\mathbf{a}^{(1)} \otimes \mathbf{b}^{(1)}\right)_{j,k} \left(\mathbf{a}^{(2)} \otimes \mathbf{b}^{(2)}\right)_{j,k} \\
&= \sum_{j,k=1}^{d_1,d_2} \left(\mathbf{a}_j^{(1)} \mathbf{b}_k^{(1)}\right) \left(\mathbf{a}_j^{(2)} \mathbf{b}_k^{(2)}\right) \\
&= \sum_{j,k=1}^{d_1,d_2} \left(\sum_{t=1}^{T} \mathbf{a}_{jt}^{(1)} \mathbf{b}_{tk}^{(1)}\right) \left(\sum_{t=1}^{T} \mathbf{a}_{jt}^{(2)} \mathbf{b}_{tk}^{(2)}\right) \\
&= \sum_{t_1,t_2=1}^{T,T} \left(\sum_{j=1}^{d_1} \mathbf{a}_{jt_1}^{(1)} \mathbf{a}_{jt_2}^{(2)}\right) \left(\sum_{k=1}^{d_2} \mathbf{a}_{kt_1}^{(1)} \mathbf{a}_{kt_2}^{(2)}\right) \\
&= \sum_{t_1,t_2=1}^{T,T} \left(\mathbf{a}_{\cdot,t_1}^{(1)} \mathbf{a}_{\cdot,t_2}^{(2)}\right) \left(\mathbf{b}_{\cdot,t_1}^{(1)} \mathbf{b}_{\cdot,t_2}^{(2)}\right) \\
&= \left(\left(\mathbf{b}^{(1)}\right) \left(\mathbf{b}^{(2)}\right)^\intercal\right) \odot \left(\left(\mathbf{a}^{(1)}\right)^\intercal \mathbf{a}^{(2)}\right)
\end{aligned}$$

Hence, comparing with directly computing per-sample gradients, if $2T^2 < d_1 d_2$, it is more memory-efficient to first multiply the matrices of $\left(\mathbf{b}^{(1)}\right) \left(\mathbf{b}^{(2)}\right)^\intercal$ and $\left(\mathbf{a}^{(1)}\right)^\intercal \mathbf{a}^{(2)}$, then take the inner product

between the two $T \times T$ matrices. If $2T^2 \geq d_1 d_2$, then we can first take the outer products $\mathbf{a}^{(1)} \otimes \mathbf{b}^{(1)}$ and $\mathbf{a}^{(2)} \otimes \mathbf{b}^{(2)}$, then take their inner product. In either case, we only need a single backpropagation to compute the gradient inner-product between every pair of the data points within the batch, similar to the case of non-sequential data.

Concurrent with this work, we also apply the ghost inner-product technique for efficient data attribution [39].

A.2 Ghost Inner-Product for LoRA

LoRA. For a linear layer $\mathbf{W}$, LoRA (Low-Rank Adaptation) [18] introduces additional trainable parameters to adapt the model effectively while maintaining computational efficiency. Specifically, the original weight matrix $\mathbf{W}$ is modified as $\mathbf{W}' = \mathbf{W} + \mathbf{AB}$, where $\mathbf{A} \in \mathbb{R}^{d_1 \times r}$ and $\mathbf{B} \in \mathbb{R}^{r \times d_2}$ are new low-rank matrices with $r \ll \min(d_1, d_2)$. This adaptation allows for significant modifications of the layer's behavior through a low-rank update, which adds a minimal number of parameters to the model compared to the original number in $\mathbf{W}$. Here, we show how to compute ghost inner-product for LoRA.

The gradient of LoRA layer $\mathbf{A}$ can be computed as

$$\frac{\partial \ell}{\partial \mathbf{A}} = \frac{\partial \ell}{\partial \mathbf{W}'} \frac{\partial \mathbf{W}'}{\partial \mathbf{A}} = \left(\frac{\partial \ell}{\partial \mathbf{s}} \frac{\partial \mathbf{s}}{\partial \mathbf{W}'} \right) \mathbf{B}^\intercal = (\mathbf{a} \otimes \mathbf{b}) \, \mathbf{B}^\intercal$$

and the gradient of LoRA layer $\mathbf{B}$ can be computed as

$$\frac{\partial \ell}{\partial \mathbf{B}} = \frac{\partial \ell}{\partial \mathbf{W}'} \frac{\partial \mathbf{W}'}{\partial \mathbf{B}} = \mathbf{A}^\intercal \, (\mathbf{a} \otimes \mathbf{b})$$

Non-sequential data. For non-sequential data, the inner-product between two gradients on $\mathbf{A}$ can be written as

$$\begin{aligned}
\frac{\partial \ell^{(1)}}{\partial \mathbf{A}} \cdot \frac{\partial \ell^{(2)}}{\partial \mathbf{A}} &= \sum_{j=1}^{d_1} \sum_{k=1}^{r} \left(\left(\mathbf{a}^{(1)} \otimes \mathbf{b}^{(1)} \right) \mathbf{B}^\intercal \right)_{jk} \cdot \left(\left(\mathbf{a}^{(2)} \otimes \mathbf{b}^{(2)} \right) \mathbf{B}^\intercal \right)_{jk} \\
&= \sum_{j=1}^{d_1} \sum_{k=1}^{r} \left(\mathbf{a}_j^{(1)} \sum_{i=1}^{d_2} \mathbf{b}_i^{(1)} \mathbf{B}_{ik}^\intercal \right) \left(\mathbf{a}_j^{(2)} \sum_{i=1}^{d_2} \mathbf{b}_i^{(2)} \mathbf{B}_{ik}^\intercal \right) \\
&= \sum_{j=1}^{d_1} \left(\mathbf{a}_j^{(1)} \mathbf{a}_j^{(2)} \right) \sum_{k=1}^{r} \left(\sum_{i=1}^{d_2} \mathbf{b}_i^{(1)} \mathbf{B}_{ik}^\intercal \right) \left(\sum_{i=1}^{d_2} \mathbf{b}_i^{(2)} \mathbf{B}_{ik}^\intercal \right) \\
&= \left(\mathbf{a}^{(1)} * \mathbf{a}^{(2)} \right) \left(\left(\mathbf{b}^{(1)} \mathbf{B}^\intercal \right) * \left(\mathbf{b}^{(2)} \mathbf{B}^\intercal \right) \right)
\end{aligned}$$

Similarly, we have

$$\begin{aligned}
\frac{\partial \ell^{(1)}}{\partial \mathbf{B}} \cdot \frac{\partial \ell^{(2)}}{\partial \mathbf{B}} &= \sum_{j=1}^{d_1} \sum_{k=1}^{r} \left(\mathbf{A}^\intercal \left(\mathbf{a}^{(1)} \otimes \mathbf{b}^{(1)} \right) \right)_{jk} \cdot \left(\mathbf{A}^\intercal \left(\mathbf{a}^{(2)} \otimes \mathbf{b}^{(2)} \right) \right)_{jk} \\
&= \sum_{j=1}^{d_1} \sum_{k=1}^{r} \left(\sum_{i=1}^{d_2} \mathbf{A}_{ji}^\intercal \mathbf{a}_i^{(1)} \mathbf{b}_k^{(1)} \right) \left(\sum_{i=1}^{d_2} \mathbf{A}_{ji}^\intercal \mathbf{a}_i^{(2)} \mathbf{b}_k^{(2)} \right) \\
&= \sum_{k=1}^{r} \left(\mathbf{b}_k^{(1)} \mathbf{b}_k^{(2)} \right) \sum_{j=1}^{d_1} \left(\sum_{i=1}^{d_2} \mathbf{A}_{ji}^\intercal \mathbf{a}_i^{(1)} \right) \left(\sum_{i=1}^{d_2} \mathbf{A}_{ji}^\intercal \mathbf{a}_i^{(2)} \right) \\
&= \left(\mathbf{b}^{(1)} * \mathbf{b}^{(2)} \right) \left(\left(\mathbf{A}^\intercal \mathbf{a}^{(1)} \right) * \left(\mathbf{A}^\intercal \mathbf{a}^{(2)} \right) \right)
\end{aligned}$$

Sequential data. We now consider the setting of sequential data with sequence length T. In this case, we have $\mathbf{a} = (\mathbf{a}^{(1)}, \ldots, \mathbf{a}^{(B)})^\intercal \in \mathbb{R}^{B \times d_1 \times T}$ and $\mathbf{b} = (\mathbf{b}^{(1)}, \ldots, \mathbf{b}^{(B)})^\intercal \in \mathbb{R}^{B \times T \times d_2}$.

$$
\begin{aligned}
\frac{\partial \ell^{(1)}}{\partial \mathbf{A}} \cdot \frac{\partial \ell^{(2)}}{\partial \mathbf{A}} &= \sum_{j=1}^{d_1} \sum_{k=1}^{r} \left(\left(\mathbf{a}^{(1)} \otimes \mathbf{b}^{(1)} \right) \mathbf{B}^{\intercal} \right)_{jk} \cdot \left(\left(\mathbf{a}^{(2)} \otimes \mathbf{b}^{(2)} \right) \mathbf{B}^{\intercal} \right)_{jk} \\
&= \sum_{j=1}^{d_1} \sum_{k=1}^{r} \left(\sum_{i=1}^{d_2} \left(\mathbf{a}_j^{(1)} \otimes \mathbf{b}_i^{(1)} \right) \mathbf{B}_{ik}^{\intercal} \right) \left(\sum_{i=1}^{d_2} \left(\mathbf{a}_j^{(2)} \otimes \mathbf{b}_i^{(2)} \right) \mathbf{B}_{ik}^{\intercal} \right) \\
&= \sum_{j=1}^{d_1} \sum_{k=1}^{r} \left(\sum_{i=1}^{d_2} \left(\sum_{t=1}^{T} \mathbf{a}_{jt}^{(1)} \mathbf{b}_{ti}^{(1)} \right) \mathbf{B}_{ik}^{\intercal} \right) \left(\sum_{i=1}^{d_2} \left(\sum_{t=1}^{T} \mathbf{a}_{jt}^{(2)} \mathbf{b}_{ti}^{(2)} \right) \mathbf{B}_{ik}^{\intercal} \right) \\
&= \sum_{j=1}^{d_1} \sum_{k=1}^{r} \left(\sum_{t=1}^{T} \mathbf{a}_{jt}^{(1)} \sum_{i=1}^{d_2} \mathbf{b}_{ti}^{(1)} \mathbf{B}_{ik}^{\intercal} \right) \left(\sum_{t=1}^{T} \mathbf{a}_{jt}^{(2)} \sum_{i=1}^{d_2} \mathbf{b}_{ti}^{(2)} \mathbf{B}_{ik}^{\intercal} \right) \\
&= \sum_{j=1}^{d_1} \sum_{k=1}^{r} \left(\sum_{t=1}^{T} \mathbf{a}_{jt}^{(1)} \mathbf{b}_{t,\cdot}^{(1)} \mathbf{B}_{,\cdot k}^{\intercal} \right) \left(\sum_{t=1}^{T} \mathbf{a}_{jt}^{(2)} \mathbf{b}_{t,\cdot}^{(2)} \mathbf{B}_{,\cdot k}^{\intercal} \right) \\
&= \sum_{j=1}^{d_1} \sum_{k=1}^{r} \left(\sum_{t=1}^{T} \mathbf{a}_{jt}^{(1)} \mathbf{b}_{t,\cdot}^{(1)} \mathbf{B}_{,\cdot k}^{\intercal} \right) \left(\sum_{t=1}^{T} \mathbf{a}_{jt}^{(2)} \mathbf{b}_{t,\cdot}^{(2)} \mathbf{B}_{,\cdot k}^{\intercal} \right) \\
&= \sum_{j=1}^{d_1} \sum_{k=1}^{r} \left(\sum_{t=1}^{T} \mathbf{a}_{jt}^{(1)} \mathbf{D}_{tk}^{(1)} \right) \left(\sum_{t=1}^{T} \mathbf{a}_{jt}^{(2)} \mathbf{D}_{tk}^{(2)} \right) \\
&= \left(\mathbf{a}^{(1)} \right)^{\intercal} \left(\mathbf{a}^{(2)} \right) \cdot \left(\mathbf{D}^{(1)} \right) \left(\mathbf{D}^{(2)} \right)^{\intercal}
\end{aligned}
$$

where in the second-to-the-last step we denote $\mathbf{D}_{tk}^{(2)} := \mathbf{b}_{t,\cdot}^{(2)} \mathbf{B}_{,\cdot k}^{\intercal}$.

A.3 Merging Batch Selection and Gradient Update in One Backpropagation

By utilizing the ghost inner-product technique developed in this paper, we can calculate or approximate all importance scores and correction terms in a single backpropagation pass, without materializing any model-sized vectors. To compute the gradient inner-product between each training point $z_i \in \mathcal{B}_t$ and the validation data $z^{(\mathrm{val})}$, we propose including $z^{(\mathrm{val})}$ in the backpropagation along with the training batch. Specifically, we can backpropagate with respect to $\sum_{z_i \in \mathcal{B}_t} \ell^{(i)} + \ell^{(z^{(\mathrm{val})})}$.

After performing GREATS and selecting $\widehat{\mathcal{B}}_t$, it may seem necessary to backpropagate with respect to $\sum_{z_i \in \widehat{\mathcal{B}}_t} \ell^{(i)}$ to compute the gradient for the parameter update. However, this is not required. We can simply reuse the output gradient $\frac{\partial \ell^{(i)}}{\partial \mathbf{s}^{(i)}}$ from the original backpropagation and aggregate the gradients for all selected data points. This technique, referred to as the "book-keeping trick," is adapted from [6].

B Details of Experimental Setup

Training Dataset. In our experiments, we use three training datasets: LESS [43], ALPACA [37][CC-BY-NC 4.0], and OPENWEBTEXT [15][CC0]. Specifically, LESS is a combination of four instruction tuning datasets: FLAN V2 [27], COT [41], DOLLY [8], and OPENASSISTANT [24]. The LESS dataset comprises 270k data points, from which we randomly select 5% for training. Alpaca is an instruction-following dataset containing 52k data points. OPENWEBTEXT is a recreation of the WEBTEXT[34] corpus, containing approximately 8 million documents.

Evaluation Dataset. To evaluate our GREATS, we consider four datasets: MMLU [17], TYDIQA [7], SAMSUM [14], and OPENWEBTEXT [15]. Specifically, MMLU consists of multiple-choice questions covering 57 subjects, including math, computer science, US history, and more. In Table 2, we report accuracy for nine selected subjects: Sociology, US Foreign Policy, Abstract Algebra, Anatomy, Astronomy, Business Ethics, Clinical Knowledge, College Biology, and College Chemistry. TYDIQA is a multilingual question-answering dataset including 11 diverse languages. In our evaluation, we randomly select 500 test data to compute the perplexity and F1 score. The task is to extract the answer to a query from a given passage. SAMSUM is a dialog dataset with the task of summarizing a given dialogue between humans.

More Training Details. For the experiment results shown in the main paper, the training hyperparameters are shown below:

1. Finetuning LLAMA-2-7B to MMLU: We finetune LLAMA-2-7B on 5% of the LESS dataset, setting the LoRA rank to 128, LoRA α to 1.0, and dropout to 0.1. The learning rate is set to 2e-5.[3]

2. Finetuning MISTRAL-7B to TYDIQA: We finetune MISTRAL-7B on 40% of the LESS dataset, setting the LoRA rank to 128, LoRA α to 1.0, and dropout to 0.1. The learning rate is set to 1e-5.

3. Finetuning LLAMA-3-8B to SAMSUM: We finetune LLAMA-3-8B on the ALPACA dataset using `Torchtune`[4], setting the LoRA rank to 8, LoRA α to 0.1, and the learning rate to 2e-5.

4. Pretraining GPT-SMALL to OPENWEBTEXT: We pretrain the GPT-SMALL model with a learning rate of 6e-4 and a batch size of 16.[5]

More Evaluation Details. To evaluate the accuracy of MMLU and TYDIQA, we follow the LESS paper's approach, using few-shot in-context learning demonstrations. Specifically, we measure the 5-shot accuracy for the MMLU dataset and the 1-shot macro-averaged F1 score for TYDIQA. We also provide examples of evaluation data in Figure 5,6, 7, and 8.

Example of MMLU

Question: The shift from "civil religion" to "common religion" means that:
(A) the increasing bureaucracy of the state has made religion only a marginal part of our lives,
(B) despite the weakening of traditional authority, our everyday lives and 'common sense' remain shaped by religious beliefs and values
(C) religious participation in collective worship may have declined, but people still practise their faiths in private
(D) people are much more likely to discuss their religious beliefs in public, informal settings
Answer: B

Figure 5: Example of MMLU

[3]Codebase for task 1 and task 2: `https://github.com/princeton-nlp/LESS`
[4]Codebase for task 3: `https://github.com/pytorch/torchtune`
[5]Codebase for task 4: `https://github.com/karpathy/nanoGPT`

Example of TYDIQA

Passage: Home Box Office (HBO) is an American premium cable and satellite television network that is owned by the namesake unit Home Box Office, Inc., a division of AT&T's WarnerMedia. The program which featured on the network consists primarily of theatrically released motion pictures and original television shows, along with made-for-cable movies, documentaries and occasional comedy and concert specials.
Question: Who owns HBO?
Answer: Home Box Office

Figure 6: Example of TYDIQA

Example of SAMSUM

Dialog:
O'Neill: Is everything ok?
O'Neill: I didn't hear back from you O'Neill: <file_gif>
Ted: Hey
Ted: I have been really busy today
Ted: Sorry..
Ted: Yes everything is fine ;)
Ted: I'll send you a photo later on :)
O'Neill: Great!!
Answer: O'Neill is worried about not having heard from Ted. Ted is fine and is going to send a photo later.

Figure 7: Example of SAMSUM

Example of OPENWEBTEXT

Text:
A magazine supplement with an image of Adolf Hitler and the title 'The Unreadable Book' is pictured in Berlin. No law bans "Mei...

Figure 8: Example of OPENWEBTEXT

C Additional Experiment Results

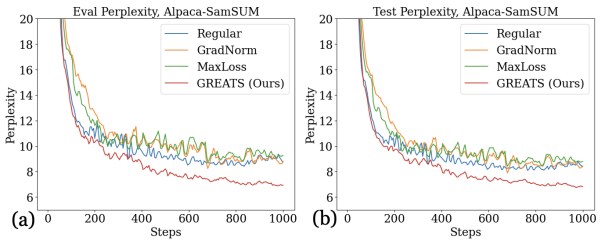

Figure 9: Experimental results on SAMSUM, where we leveraged ALPACA as the training data. Our GREATS method significantly outperforms other approaches.

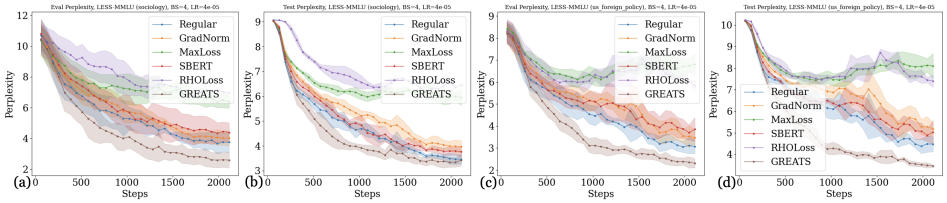

Figure 10: Similar to Figure 2, instead, we use a batch size of **4** and a learning rate of **4e-5**.

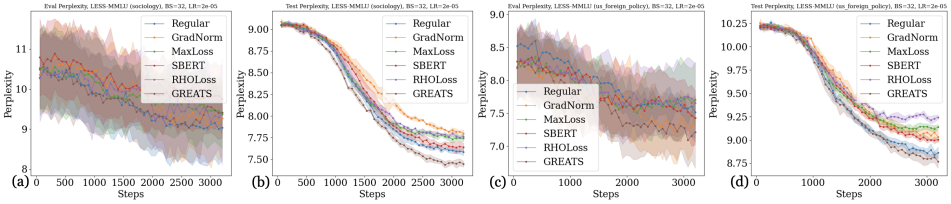

Figure 11: Similar to Figure 2, instead, we use a batch size of **32** and a learning rate of **2e-5**.

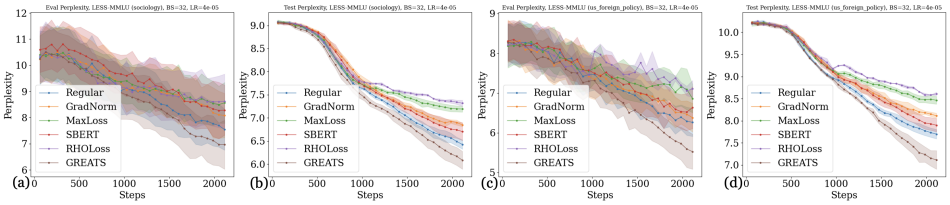

Figure 12: Similar to Figure 2, instead, we use a batch size of **32** and a learning rate of **4e-5**.

D Broader Impacts

We expect our work to have a positive societal impact. We developed a novel method to facilitate the training process of large language models.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Our main contribution claimed in the introduction is developing an efficient online batch selection algorithm which is detailed in Section 4.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: In Section 6, we discussed the limitation of this work.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: The paper does not include theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: In Appendix B, we include all information required for reproducing our experiment results.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: We will release the code repo for this work at the time of publishing. We provided the link to the Github repo that we built upon in Appendix B.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: In Appendix B, we include all information required for reproducing our experiment results.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Due to the computationally expensive nature, for data selection experiments we only take 1 training run, and for most of the other experiments we take the results from the average of 3 runs.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: In Appendix B, we include all information required for reproducing our experiment results.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: we have reviewed the ethics requirement.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discussed the broader impacts in Appendix D.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: all information is mentioned in Appendix B.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: the paper does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [No]

    Justification: the paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.