Towards Scalable Data Diversification for Language Model Pretraining via Leverage Score Sampling

Zailin Ma[1,2*] **Quzhe Huang**[2] **Yujun Li**[2] **Congyuan Rao**[3] **Yaodong Yang**[1†]

[1]Peking University [2]Wizard Intelligence Learning Lab [3]Tsinghua University

Abstract

Data selection for language model pretraining faces a fundamental tension between quality and diversity. While quality filtering is empirically effective, it often induces diversity collapse: by favoring texts similar to high-quality reference corpora (e.g., educational or QA-style data), it systematically excludes valuable data from underrepresented domains. In contrast, diversified selection preserves domain balance and encourages robust downstream performance, yet existing methods either focus on coverage-oriented objectives that indirectly enhance diversity, or directly optimize for diversity via costly covariance matrix recomputation that limits scalability. To address these issues, we introduce **Leverage Score Sampling (Lev)**, which iteratively selects samples that maximally expand the determinantal volume of the embedded data via leverage scores, a computationally efficient criterion that eliminates matrix recomputation and enables scalable selection. Empirically, Lev delivers up to $72\times$ speedup and improves dataset diversity, measured by the Vendi score, by 9.2% over the strong diversification baseline **DiSF**. On CommonCrawl (CC) web data selection, Lev improves accuracy across seven downstream tasks by up to 1.31% over existing baselines. For domains where robust quality criteria are inherently difficult to define (e.g., code), Lev serves as an effective unsupervised curation alternative: on StarCoderData, the selected subset reduces bits-per-byte by 3.08% over DiSF. Notably, we uncover a cross-domain collapse of quality filtering: CC data filtered by DCLM-fastText fail to retain sufficient code-related content, yielding inferior code performance relative to Lev-selected data. These findings advocate for integrating diversity-aware practices into quality filtering for more effective data curation in language model pretraining.

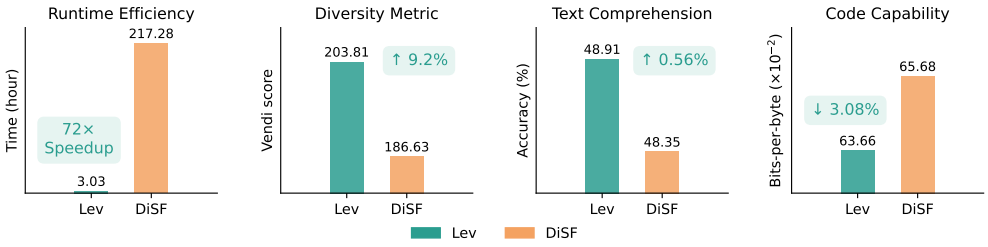


Figure 1: Lev outperforms existing diversification baseline DiSF across all fronts.

1 Introduction

Large language models (LLMs) have achieved unprecedented performance in general text comprehension and problem-solving tasks [12, 47], with their capabilities critically hinging on the composition

*Email: mazailin@stu.pku.edu.cn. Work done during an internship at Wizard Intelligence Learning Lab.
†Corresponding author: yaodong.yang@pku.edu.cn

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

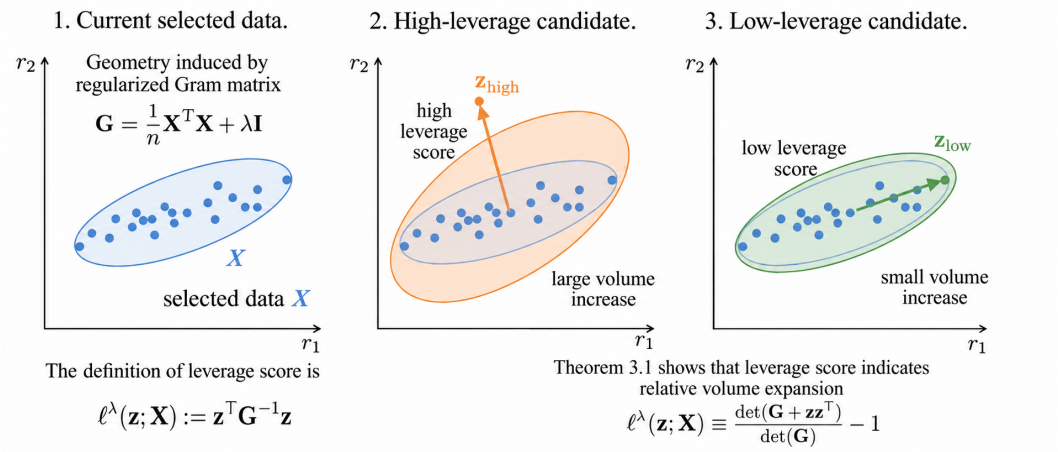


Figure 2: Leverage score is efficiently computed via a quadratic form (left). It quantifies the relative volume expansion of the candidate sample once selected (right), enabling effective diversification.

of pretraining data. Extensive research has sought to curate higher-quality corpora from raw web sources such as CommonCrawl (CC) [43, 35]. Among these efforts, model-based quality filtering [34, 54] has emerged as the dominant mechanism owing to its superior empirical performance. Practitioners train lightweight classifiers such as fastText [23] using curated question-answering or LLM-labeled high-quality texts as positive samples [24, 52]. However, this approach induces **dimensional collapse** [14]: the embedding vectors of the filtered data span a low-dimensional subspace, indicating severe diversity degradation from a geometric perspective, as the classifier favors data similar to the narrow references but neglects out-of-domain instances. Consequently, the resulting models exhibit performance degradation on underrepresented domains of the curated corpora.

To mitigate this degradation, diversified data selection has emerged as a natural solution. Early efforts address this through domain-level organization [55] or attribute-level balancing [65, 26], which only indirectly alleviate dimensional collapse by operating at a coarse granularity. Finer-grained approaches, such as coverage-based sampling [41, 38], pursue uniformity over the data distribution; yet uniform density does not guarantee that selected samples span the full representation space needed for geometric diversity. To directly optimize set-level geometric diversity, DiSF [14] greedily selects samples that minimize the Frobenius norm of the covariance matrix, achieving strong downstream performance. However, this procedure requires per-candidate covariance recomputation at every iteration, incurring prohibitive computational overhead that severely limits scalability.

Motivated by these limitations, we propose **Leverage Score Sampling (Lev)**, a scalable scheme that employs leverage scores [33, 27, 9] as a principled metric for diversity improvement. Grounded in the geometric insight that dataset diversity corresponds to the determinantal volume of the representation subspace spanned by embedded samples [50], we prove that the leverage score of a candidate exactly quantifies its contribution to expanding this volume upon inclusion in the selected subset (Theorem 3.1). Building on this result, Lev iteratively selects samples with the highest leverage scores to directly maximize geometric diversity (Figure 2). Unlike DiSF, which requires costly covariance matrix recomputation for each candidate to evaluate its marginal diversity contribution, Lev evaluates all candidates via quadratic expressions with a single shared Gram matrix, achieving superior computational efficiency and enabling scalable applications.

We validate Lev through extensive experiments. Compared to the strong diversification baseline DiSF, it improves dataset diversity by 9.2% while achieving a 72× runtime speedup. On CommonCrawl (CC) web data subset selection, it improves average accuracy across seven downstream tasks by up to 1.31% over existing baselines. Crucially, Lev requires no curated quality references, making it uniquely effective for domains where defining quality is inherently ambiguous (e.g., code): on StarCoderData, it reduces bits-per-byte by 3.08% over DiSF. Beyond these gains, our analysis uncovers that the prevailing quality filter DCLM-fastText [24] induces a **code-specific dimensional collapse**: it insufficiently retains code-related content from CC compared to Lev, causing persistent code inferiority relative to diversity-based selection. Our contributions are summarized as follows.

- We establish leverage scores as a principled metric for diversity enhancement and propose **Leverage Score Sampling (Lev)**, a scalable diversification algorithm. We show that Lev is provably more efficient than DiSF and demonstrate up to $72\times$ speedup in practice.
- We demonstrate that Lev achieves superior downstream performance over existing baselines on CommonCrawl web data selection. For domains lacking robust quality criteria (e.g., code), we show that Lev serves as an effective **unsupervised** curation strategy: on StarCoderData, Lev selects subset that reduces bits-per-byte by 3.08% over DiSF.
- We uncover that quality filtering induces cross-domain dimensional collapse, disproportionately excluding code-related content that diversity-based methods effectively preserve. This finding underscores the necessity of integrating diversity-aware selection into quality filtering for more robust data curation.

2 Preliminaries

2.1 Problem statement

We study the problem of selecting a diverse subset of pretraining data for language models subject to a fixed selection ratio. Let $\mathcal{D} = \{c_1, \ldots, c_D\}$ denote a corpus where each c_i is a document. Examples include heuristically filtered and deduplicated web text (e.g., CommonCrawl) or source code (e.g., GitHub). Each document c_i is mapped to a d-dimensional *unit vector* via an embedding model ϕ, i.e., $x_i = \phi(c_i) \in \mathbb{R}^d$ and $\|x_i\|_2 = 1$. Given a target selection ratio α, we formulate the data selection problem as:

$$\mathcal{S}^* = \underset{\substack{\mathcal{S}\subset\mathcal{D}\\ |\mathcal{S}|=\alpha D}}{\operatorname{argmax}} \operatorname{Diversity}(\mathcal{S}), \tag{1}$$

where $\operatorname{Diversity}(\mathcal{S})$ denotes a set function that quantifies the diversity of the subset $\mathcal{S}$.

2.2 Recent progress

Existing approaches to diversity-aware data curation can be broadly categorized into heuristic-guided diversity methods and methods that directly optimize set-level diversity. A large body of prior work falls into the former category, encouraging diversity through indirect signals such as domain-level organization [55], attribute-level balancing based on aggregated quality signals [65, 26, 15], or coverage-based sampling that promotes uniform coverage via clustering or local density estimates [41, 38]. Among these approaches, coverage-based methods come closest to capturing sample-level diversity. However, density-based uniformity does not necessarily imply geometric diversity: in high-dimensional spaces, uniformly distributed samples may still lie in a low-dimensional subspace, thereby failing to expand the representational subspace. In contrast, geometry-based methods explicitly optimize $\operatorname{Diversity}(\mathcal{S})$ in Eq. (1). For instance, DiSF [14] greedily selects x_i by maximizing $f(x_i) = e^{-\|C(\mathcal{S}\cup\{x_i\})\|_F}$, where $C(\mathcal{S})$ denotes the normalized feature covariance matrix, encouraging a more uniform eigenspectrum. However, evaluating this score requires expensive iterative covariance updates, which severely limits scalability. This computational bottleneck motivates us to seek an alternative geometric objective that directly spans the representation subspace without repeated covariance tracking. A broader review of related work is provided in Appendix A.

3 Method

3.1 Leverage score as diversity signal

To address this computational bottleneck, we introduce leverage scores as an effective and scalable diversity signal. Leverage scores were originally used for scalable kernel approximation in ridge regression [33, 27, 9, 29, 2], where they identify samples that most effectively span the representational space for efficient Gram matrix approximation. Grounded in this geometric insight, we use leverage scores as a principled measurement for dataset diversity spanning. Specifically, let $X = [x_1, \ldots, x_n]^\top \in \mathbb{R}^{n\times d}$ denote the embedding vectors of the selected data pool. For a candicate sample with embedding $z \in \mathbb{R}^d$, the leverage score of z with respect to X is defined as

$$\ell^\lambda(z; X) := z^\top \left(\frac{1}{n} X^\top X + \lambda I_d\right)^{-1} z, \tag{2}$$

Algorithm 1 Leverage Score Sampling for Diversified Data Selection

Input: Full corpus embeddings $\mathcal{Z} = \{z_1, \ldots, z_D\} \subset \mathbb{R}^d$, selection ratio $\alpha \in (0, 1)$, batch size B, step size b, regularization λ
Output: Selected indices $\mathcal{S}$
$M \leftarrow \lceil D/B \rceil$ ▷ batch number
for each batch $\mathcal{B} \subset \{\mathcal{B}_1, \ldots, \mathcal{B}_M\}$ **do**
  Randomly select b indices from $\mathcal{B}$, add to $\mathcal{S}$
  $K \leftarrow \sum_{i \in \mathcal{S}} z_i z_i^\top$ ▷ initialize Gram matrix
  for $t = 1$ to $\lceil \alpha B/b \rceil - 1$ **do**
    $G \leftarrow \frac{1}{|\mathcal{S}|} K + \lambda I_d$ ▷ regularized Gram matrix
    $L \leftarrow \text{Cholesky}(G)$ ▷ Cholesky decomposition, $G = LL^\top$
    for each candidate $i \in \mathcal{B}$ **do**
      $w \leftarrow L^{-1} z_i$ ▷ solve triangular system
      $\ell_i \leftarrow \|w\|_2^2$ ▷ leverage score
    end for
    Sort candidates in $\mathcal{B}$ by leverage scores in descending order
    Select top-b indices not in $\mathcal{S}$, denote as $\mathcal{I}_{\text{new}}$
    $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{I}_{\text{new}}$
    $K \leftarrow K + \sum_{i \in \mathcal{I}_{\text{new}}} z_i z_i^\top$ ▷ update Gram matrix
  end for
end for
return $\mathcal{S}$

where $\lambda > 0$ is a regularization constant that ensures numerical stability (e.g., $\lambda = 10^{-5}$). Let $G = \frac{1}{n} X^\top X + \lambda I_d$ be the corresponding regularized Gram matrix. We provide Theorem 3.1 that rigorously describes the geometric insight of leverage scores and Figure 2 for a concrete illustration.

Theorem 3.1 *For a candidate sample $z \in \mathbb{R}^d$, the following relationship holds:*

$$\ell^\lambda(z; X) = \frac{\det(G + zz^\top)}{\det(G)} - 1. \tag{3}$$

The determinant of the regularized Gram matrix G can be interpreted as the squared volume of the polytope spanned by X in the representation space [17]. The theorem reveals that the leverage score directly quantifies the relative volume expansion induced by adding the candidate sample z to the current data pool. Samples with higher leverage scores yield larger volume expansion, and therefore contribute more substantially to the geometric diversity of the selected set.

3.2 Algorithm

We present our diversified data selection method *Leverage Score Sampling (Lev)* in Algorithm 1. Let D denote the corpus size, B the batch size, b the step size, d the embedding dimension and α the selection ratio. In practice, B and d are typically of the same order of magnitude. The algorithm processes the full corpus in batches of size B, requiring approximately D/B batch iterations. Within each batch, we perform $m = \alpha B/b$ sub-iterations to select b samples at each step, resulting in αB samples in total. At each iteration, the leverage scores of candidate samples are computed relative to the selected data pool X, and samples with the highest score are added to X. Then we update the Gram matrix with the newly selected data for the next round of selection. The algorithm has two key computational advantages. First, we employ Cholesky decomposition [17] on the Gram matrix to compute leverage scores. Instead of directly inverting Σ, we solve $Lw = z_i$ and compute $\ell_i = \|w\|_2^2$, which is algebraically equivalent to computing $\ell_i = z_i^\top \Sigma^{-1} z_i$, but is numerically more stable and efficient. Second, the computation of leverage scores for all candidate samples rely on a single shared Gram matrix G, which can be efficiently maintained by incrementally updating the feature covariance matrix K through the selection process. This obviates the need to recompute feature covariance matrix for every candidate sample, thus achieving significantly better computational efficiency than DiSF [14]. Codes are available at: `https://github.com/mazailin/LevDataDiversity`.

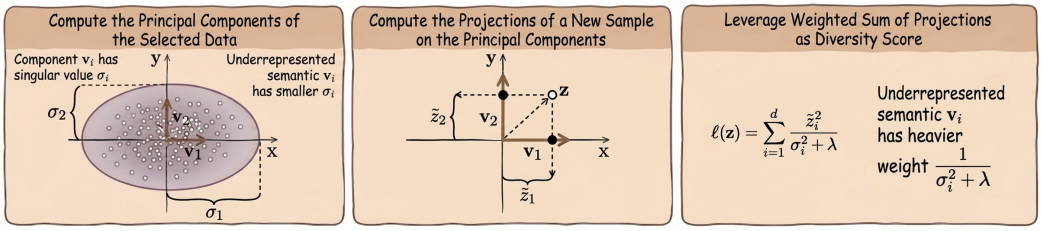


Figure 3: The statistical mechanism of leverage score functions: underrepresented semantic components are assigned higher weights, highlighting samples that stretch along these components.

3.3 How do leverage scores identify underrepresented semantics?

To further dive into the mechanism of how exactly leverage score works, we consider the structure of the Gram matrix. Let $X/\sqrt{n} = U\Sigma V^\top$ be the singular value decomposition, where U and V are orthonormal matrices and $\Sigma = \text{diag}\{\sigma_1, \ldots, \sigma_d\}$. Then σ_i^2 is the i-th eigenvalue of $X^\top X/n$. We define the columns v_i of V as the *principal semantic vectors* of the current data pool X. Because $\sum_{i=1}^{d} \sigma_i^2 = 1$, the eigenvalue σ_i^2 reflects the *occurrence frequencies* of v_i (see Appendix B.3 for details). More frequently occured semantics share larger values of σ_i^2. For a new candidate z, let $\tilde{z}_i = v_i^\top z$ denote its projection onto the i-th principal semantic vector. We have the following result.

Theorem 3.2 *The leverage score function has a decomposed expression as follows.*

$$\ell^\lambda(z; X) = \sum_{i=1}^{d} \frac{\tilde{z}_i^2}{\sigma_i^2 + \lambda}. \tag{4}$$

This theorem reveals that $\ell^\lambda(z; X)$ aggregates the contributions of z across all principal semantic vectors, weighted by the *inverse* of their occurrence frequencies and the regularization parameter $\lambda \ll 1$. If v_i has almost never occurred in X, then $\sigma_i^2 \approx 0$. Thus the magnitude of $1/(\sigma_i^2 + \lambda) \approx 1/\lambda$ becomes significant such that it effectively highlights samples that cover more semantics along v_i. If $\sigma_i^2 > 0$, then $1/(\sigma_i^2 + \lambda) \approx 1/\sigma_i^2$ is dominated by the occurrence frequency, where semantics with lower frequencies share heavier weights. Grounded in this statistical mechanism, the score is named after *leverage*. Empirically, setting $\lambda = (100 \times d)^{-1}$ is sufficient to highlight the uncovered semantic signals. We provide the proofs of both theorems and a detailed discussion on λ in Appendex B.

4 Experiments

4.1 Setup

Datasets and models. We consider two domain data, web and code. For web data, we process a **CommonCrawl (CC)** dump with `datatrove` [34] following similar pipeline steps in FineWeb [34], including heuristic filtering and deduplication and excluding quality filtering to preserve the original degree of data diversity. For code data, we employ **StarCoderData** [25] and restrict our experiments to six programming languages including Python, Java, JavaScript, Go, PHP, and Ruby. We adopt the model architecture of Llama 3.1 [18] with an adapted configuration of 1.1B parameters detailed in Appendix C.4, and consider a training budget of 60B token data sampled by various strategies. With a batch size of 1024 and a context length of 2048, this configuration results in 28610 training iterations. The training configuration is applied for both the main experiments and the ablation study.

Baselines. For diversity-based methods, we include **DiSF** [14] that maximizes the uniformity of the eigenvalues of the data covariance matrix, and **Density** sampling [38] that uses density estimation for uniform sampling over the original corpus. **DSIR** [57] matches samples with targeted domains, *Wikipedia* and *Book*, using n-gram features. **Random** sampling provides a non-selective baseline. Additionally, we apply **DCLM-fastText** [24] to obtain high-quality subsets from CC for comparative experiments. DCLM-fastText is a strong quality scorer widely used for CC quality filtering. It is trained on positive samples of instruction-formatted and question-answer style text spanning a broad range of topics, along with negative samples from the original RefinedWeb [35]. Our method, **Lev**,

Table 1: Downstream performance on CommonCrawl subset selection. Higher is better. Accuracy(%) is reported with subscripts indicating the ranking. (Best in bold. Second best in italics.)

Method	50% Selection Ratio								40% Ratio
	ARC-e	ARC-c	WG	HS	PIQA	SIQA	CSQA	Mean ↑	Mean ↑
Random	48.50	27.82	52.09	46.50	**69.40**	**46.30**	**43.00**	$47.66_{(3)}$	$47.66_{(4)}$
DSIR	**53.50**	27.73	**53.04**	42.00	65.20	44.30	42.26	$46.86_{(5)}$	$44.84_{(5)}$
Density	51.70	27.99	50.75	**48.20**	66.40	45.00	42.51	$47.51_{(4)}$	$47.98_{(3)}$
DiSF	*52.90*	*28.16*	*52.49*	46.40	67.80	44.50	*42.59*	$47.83_{(2)}$	$48.35_{(2)}$
Lev (ours)	52.70	**29.35**	52.33	*47.10*	*68.30*	*45.10*	42.34	$\mathbf{48.17}_{(1)}$	$\mathbf{48.91}_{(1)}$

introduces the leverage score function as a principled diversity measurement. For each baseline, we consider two selection ratios, 40% and 50%, for empirical validations.

Evaluations. We evaluate all models using the OLMES framework [19]. For natural language understanding, we assess performance on seven established benchmarks: **ARC-easy**, **ARC-challenge** [10], **WinoGrande** [39], **HellaSwag** [63], **PIQA** [3], **Social IQa** [40] and **CommonSenseQA** [46]. The choice with the lowest perplexity is considered the response of the model and the overall accuracy is the primary metric. We define the mean score of the seven tasks as **Base Mean**, describing the base language understanding ability of a model. For code evaluation, we adopt tasks of six languages in **Multilingual MBPP** [5] that correspond to our datasets and **HumanEval** [6]. Bits per byte (BPB) is the primary metric. We define the mean BPB value of the seven benchmarks as **Code Mean**.

4.2 Main results

4.2.1 Lev outperforms existing diversity-based selection methods

We conduct a systematic comparison against existing diversity-based methods across three dimensions: direct dataset diversity evaluation, downstream language performance on CommonCrawl subsets and downstream code performance on StarCoderData subsets. For diversity measurements, we adopt seven diversity metrics from [59] to evaluate dataset diversity. Specifically, CosDist, KNN, and CluIne are distance-based metrics that emphasize sample-level uniqueness or dispersion. Entropy, Vendi score, and LogDet are spectrum-based metrics that reflect geometric diversity through the eigenstructure of the Gram or covariance matrix. NovelSum integrates both perspectives for a more comprehensive diversity characterization. We normalize the raw values of each metric via affine transformations that map the observed minimum and maximum to 0.3 and 1.0 respectively. The normalized values are shown in Figure 4. We provide full definitions of metrics in Appendix C.6 and full results in Appendix D.1.

Diversity measurement of selected subsets. Figure 4 shows that Lev achieves the best performance across all diversity metrics on Problem (1), surpassing the strong diversity-oriented baseline, DiSF. Specifically, Lev improves the Vendi score by 21.3% over Random (168.06 vs. 203.81) and by 9.2% over DiSF (186.63 vs. 203.81). In contrast, DSIR collapses into two specific domains (Wikipedia and books), leading to severe diversity degradation. Density, being a coverage-oriented method, does not directly optimize for geometric diversity and thus trails behind DiSF and Lev. Finally, Lev outperforms DiSF because it better captures the per-sample contribution to subspace span, eventually yielding selected datasets of substantially higher diversity. We provide a comparative theoretical analysis over the optimization objectives of Lev and DiSF in Appendix E.1, which shows that Lev obtains better diversity results because it is more sensitive in detecting diverse samples.

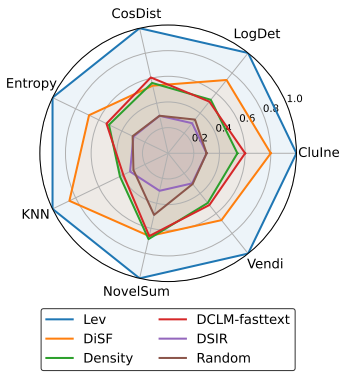
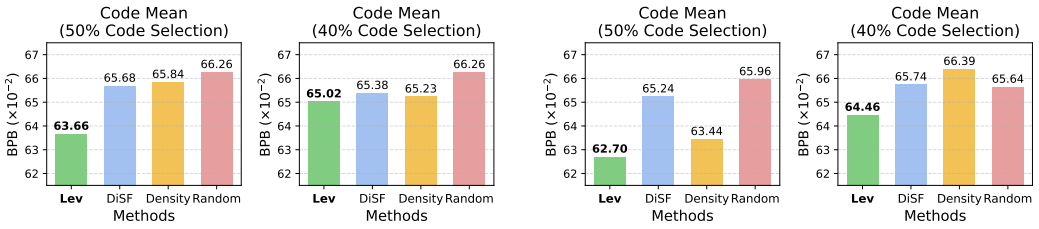


Figure 4: Diversity comparison.

Downstream performance for CommonCrawl subset selection. We evaluate models trained on CommonCrawl subsets selected by different methods across seven language understanding



(a) Combined with quality-filtered CC data.

(b) Combined with Lev-sampled CC data.

Figure 5: Code performance of different diversity-based methods. Lower is better.

tasks. Random and DSIR serve as baselines. Table 1 shows that Lev consistently achieves the best performance among all diversity-based methods, yielding an average improvement of 1.31% over DSIR at a 50% selection ratio. DSIR peaks on ARC-e and WinoGrande but bottoms out on ARC-c and HellaSwag, exhibiting the performance imbalance caused by dimensional collapse. In contrast, Lev achieves the best or second-best performance on ARC-c, HellaSwag, PIQA, and SIQA, demonstrating that diversity enhancement fosters more balanced knowledge extraction from CC web data and yields consistently strong performance across tasks.

Downstream performance for StarCoderData subset selection. We show that for code data, where robust quality heuristics are difficult to define, Lev is effective at curating high-quality code datasets in an unsupervised manner. In fact, our findings reveal that diversification itself improves the utility of code data. To validate this, we prepare 60B-token pretraining corpora consisting of 70% CC data and 30% StarCoderData. We apply different diversity methods to select code subsets at 40% and 50% selection ratios, and combine the selected subsets with two types of CC data: Lev-sampled and quality-filtered CC subsets (DCLM-fastText serves as the quality scorer). This yields four data combinations. We then evaluate BPB of the resulting models on code-related tasks. The results are shown in Figure 5 (full results in Appendix D.3). We draw the following conclusions:

1. **Diversified code data improves code performance.** Almost all diversity-based methods applied to code data selection improve downstream performance on code tasks, as reflected by lower BPB values relative to the uncurated baseline (Random) in Figure 5.
2. **Lev achieves the best code performance among diversity-based methods.** Lev consistently attains the lowest code BPB across all four settings in Figure 5. Specifically, at a 50% code selection ratio with quality-filtered web data (Figure 5a), Lev reduces average code BPB by 3.92% over Random (from 66.26 to 63.66) and by 3.08% over DiSF (from 65.68 to 63.66). With Lev-sampled web data (Figure 5b), the improvement over Random increases to 4.94% (from 65.96 to 62.70). Results at the 40% selection ratio corroborate this finding.

4.2.2 Revisiting dimensional collapse: quality filtering demands diversity

Table 2: Comparison between DCLM-fastText (Qual) and Lev under two data recipe. BPB($\times 10^{-2}$) for Code Mean and Accuracy(%) for Base Mean. Subscripts indicate the ranking.

(a) Recipe: web only

Web Data	Code Mean ↓	Base Mean ↑
Lev (top 40%)	**256.43**$_{(1)}$	48.91$_{(2)}$
Lev (top 50%)	**257.35**$_{(2)}$	48.17$_{(3)}$
Qual (top 40%)	266.77$_{(3)}$	**50.21**$_{(1)}$
Random	273.22$_{(4)}$	47.66$_{(4)}$

(b) Recipe: 70% web + 30% Lev curated code data

Web Data	Code Mean ↓	Base Mean ↑
Lev (top 40%)	**62.48**$_{(1)}$	49.90$_{(2)}$
Lev (top 50%)	**62.70**$_{(2)}$	49.17$_{(4)}$
Qual (top 40%)	63.66$_{(3)}$	**51.53**$_{(1)}$
Random	64.46$_{(4)}$	49.28$_{(3)}$

Beyond diversity-based methods, quality filtering has long dominated pretraining data curation, owing to its stronger performance on downstream commonsense tasks. However, we overturn this conventional wisdom by uncovering a previously unknown phenomenon: quality filtering can induce cross-domain dimensional collapse, which leads to persistent suboptimality on domain-specific modeling capability.

Using DCLM-fastText as a case study, we find that it likely overweights high-quality text comprehension data relative to code-related data in CC, yielding a text–code imbalance in the filtered data. To demonstrate this, we first pretrain models separately on quality-filtered CC data and Lev-selected CC data. Table 2a shows that models trained on quality-filtered CC data exhibit significantly higher code BPB values than those trained on Lev-selected data, suggesting that the former contains less code-related content. We then augment both curated CC subsets with dedicated code data sampled from StarCoderData using Lev, and train additional models on these mixtures. Nevertheless, Table 2b shows that the code performance gap persists. We repeat this experiment with two additional code datasets, and observe consistent results, which are fully reported in Appendix D.4. We also conduct further investigation in Appendix E.2, demonstrating that the inadequate recall of code data is a common limitation among open-source data scorers. This indicates that quality filtering necessitates a broader consideration of diversity.

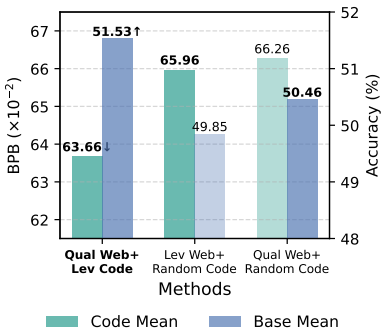


Figure 6: Quality-filtered CC + Lev-selected code data achieves the best performance.

Applying Lev to code data partially mitigates the code inferiority of quality-filtered web data, resulting in the best overall performance. Figure 6 shows that combining quality-filtered CC data with Lev-selected code data yields a model that achieves lower code BPB (63.66 vs. 65.96) than the one trained on Lev-selected CC mixed with unfiltered code, while also improving text comprehension accuracy (51.53% vs. 50.46%). However, we emphasize that deeply integrating diversity and quality objectives to sample a more holistic CC subset remains a critical direction for future research. Concurrent work has explored effective methods along these lines [26, 15]. Ultimately, our work highlights the indispensable role of dataset diversity in LLM pretraining data curation.

4.2.3 Lev enables efficient large-scale diversity selection

While Lev achieves strong diversification, it also proves more efficient than DiSF in both theoretical complexity and empirical runtime, enabling larger-scale data selection.

Table 3: Runtime comparison in two real scenarios: web and code selection.

Step size	CommonCrawl			StarCoderData		
	Lev	DiSF	Speedup	Lev	DiSF	Speedup
1	3.03h	217.28h	72×	1.03h	67.45h	65×
2	1.67h	108.48h	65×	34min	33.68h	59×
4	55min	56.12h	61×	19min	17.05h	54×
8	35min	26.88h	46×	11min	8.58h	44×

Theoretical Analysis. For Lev, Algorithm 1 processes the corpus in $M = D/B$ batches. Within each batch, it performs $m = \alpha B/b$ iterations. The dominant computational cost per iteration is $O(Bd^2 + d^3)$ for computing leverage scores, comprising: (1) $O(d^3)$ for Cholesky decomposition, (2) $O(Bd^2)$ for batch-wise leverage score computation, and (3) $O(bd^2)$ for Gram matrix updates. To conclude, Lev has an overall time complexity of $O(Mm(Bd^2 + d^3)) = O(DBd^2)$ when $B > d$.

In contrast, DiSF's equivalent score function is $f(z_i) = e^{-\|C(\mathcal{S}\cup\{z_i\})\|_F}$, where $C(\mathcal{S})$ is the normalized feature covariance matrix with respect to the selected data $\mathcal{S}$. This involves $O(\alpha B^2 d^2)$ complexity per iteration, as computing $f(z_i)$ for B candidates requires $O(\alpha B d^2)$ operations each. Consequently, DiSF's overall complexity is $O(Mm\alpha B^2 d^2) = O(DB^2 d^2)$. We provide a more elaborated analysis in Appendix E.3.

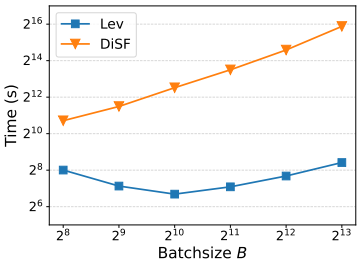
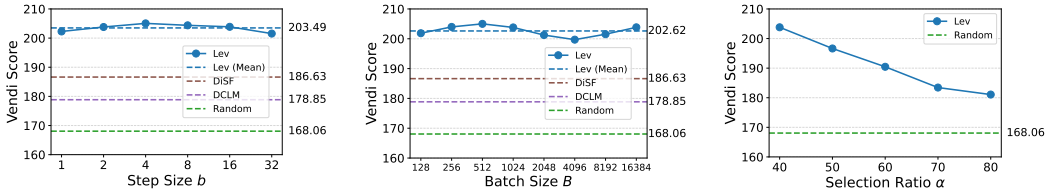
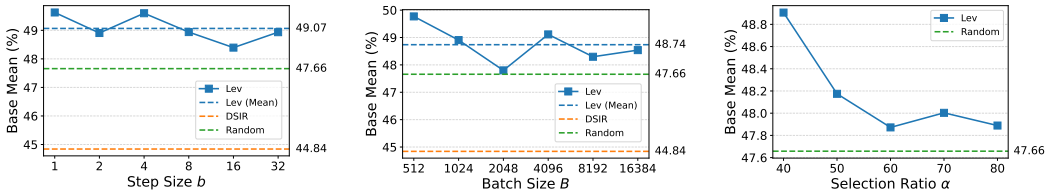


Figure 7: Runtime comparison between Lev and DiSF with respect to batch size B.

Runtime Comparison. The former analysis indicates that Lev eliminates one order of B compared to DiSF. To validate this, we measure the runtime of data selection with respect to the batch size B.



(a) Vendi score vs. Step size. (b) Vendi score vs. Batch size. (c) Vendi score vs. Selection ratio.



(d) Base mean vs. Step size. (e) Base mean vs. Batch size. (f) Base mean vs. Selection ratio.

Figure 8: Sensitivity to hyperparameters.

We select 40% of the documents from a CommonCrawl subset using 8 NVIDIA RTX 5090 GPUs that process 8 files in parallel. Figure 7 shows that Lev is faster than DiSF by several orders of magnitude overall, and the gap widens as B grows. We then evaluate both algorithms in two realistic settings: CommonCrawl and StarCoderData subset selection. We adopt a batch size of 1024, which is the best-practice setting for DiSF in [14], and measure the runtimes across different step sizes. The greedy selection for submodular function optimization in DiSF corresponds to the step size of 1. Table 3 shows that with a step size of 1, Lev processes a CommonCrawl dump in only about 3.03 hours, whereas DiSF requires 217.28 hours, *confirming a* $72\times$ *speedup*. For Lev, Figure 7 exhibits an anomalous monotonic decrease when B is small. This occurs because leverage sampling is extremely efficient in this regime, shifting the bottleneck to other parts of the pipeline. Overall, Lev exhibits superior scalability compared to DiSF.

4.3 Ablation Study

We investigate the sensitivity of Algorithm 1 to key hyperparameters: step size, batch size, and selection ratio. To isolate effects from cross-domain interference, we consider only CC web data at a 40% selection ratio for evaluating step size and batch size. We show that Lev is robust to variations in step size and batch size, and remains effective across a range of selection ratios for both CC and code.

Step size b is insensitive. A larger step size implies that more samples are selected in a single update. Intuitively, this might appear to compromise diversity. Contrary to this intuition, Figure 8a suggests otherwise. Because the raw data is shuffled, the b samples selected at each step are highly likely to be distinct. Thus, a larger step size preserves data diversity. This is further corroborated by Figure 8d, which shows that downstream scores fluctuate around the mean without exhibiting significant monotonicity, confirming that step size is an insensitive parameter in Algorithm 1. Empirically, a larger step size is preferred because of its substantial computational speedups (Table 3).

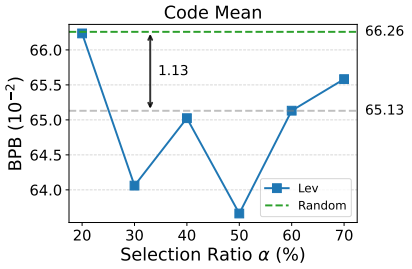


Figure 9: Code mean vs. Code selection ratio. Lower is better.

Batch size B is insensitive. Intuitively, a larger batch size might appear to ensure higher data diversity, as the reference data pool X covers more selected samples during each iteration. However, Figures 8b and 8e contradict this expectation. Both diversity and downstream scores fluctuate around their respective means without exhibiting a significant monotonic trend with respect to B, indicating that the batch size is also an insensitive parameter in our algorithm. Since a larger B incurs greater computational overhead without yielding corresponding diversity gains, we select $B = 1024$ to

balance efficiency and performance. As shown in Table 5, this setting already exhibits a high degree of diversity and achieves a considerable advantage over other baselines.

Selection ratio α. Table 8c shows that web data diversity decreases as the selection ratio increases, leading to a corresponding drop in downstream performance in Table 8f. However, Lev consistently outperforms the Random baseline even when α exceeds 50%, underscoring the effectiveness of data diversification. Particularly for code data selection shown in Figure 9, we observe that Lev is consistently effective within a wide range of selection ratios (e.g., 30% to 60%), yielding a stable reduction of over 1.13×10^{-2} in bits-per-byte compared to the baseline.

5 Conclusion

We present Leverage Score Sampling (Lev), a scalable diversity-oriented method that efficiently expands the representation subspace by iteratively selecting high-leverage samples. Empirically, Lev outperforms existing diversification baselines across diversity metrics, downstream task performance, and runtime efficiency while simultaneously achieving a $72\times$ speedup over DiSF. Our analysis further reveals a collateral cross-domain dimensional collapse induced by quality filtering, exemplified by the systematic exclusion of code-related content from CommonCrawl. These findings underscore the essential complementary role of diversity-aware selection to quality filtering and inspire tighter integration of diversity and quality objectives in building more robust data curation pipelines.

Acknowledgments and Disclosure of Funding

This work is sponsored by the National Natural Science Foundation of China (No. 22494644, No. 62561160152). Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the funding agencies. This work was done during the first author's internship at Wizard Intelligence Learning Lab.

References

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

[2] Ahmed Alaoui and Michael W Mahoney. Fast randomized kernel ridge regression with statistical guarantees. *Advances in Neural Information Processing Systems*, 28, 2015.

[3] Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439, 2020.

[4] Alexander Bukharin, Shiyang Li, Zhengyang Wang, Jingfeng Yang, Bing Yin, Xian Li, Chao Zhang, Tuo Zhao, and Haoming Jiang. Data diversity matters for robust instruction tuning. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 3411–3425, Miami, Florida, USA, 2024. Association for Computational Linguistics.

[5] Federico Cassano, John Gouwar, Daniel Nguyen, Sydney Nguyen, Luna Phipps-Costin, Donald Pinckney, Ming-Ho Yee, Yangtian Zi, Carolyn Jane Anderson, Molly Q Feldman, Arjun Guha, Michael Greenberg, and Abhinav Jangda. Multipl-e: A scalable and extensible approach to benchmarking neural code generation, 2022.

[6] Mark Chen. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[7] Yicheng Chen, Yining Li, Kai Hu, Zerun Ma, HaochenYe HaochenYe, and Kai Chen. Mig: Automatic data selection for instruction tuning by maximizing information gain in semantic space. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 9902–9915, Vienna, Austria, 2025. Association for Computational Linguistics.

[8] Yilei Chen, Souradip Chakraborty, Lorenz Wolf, Ioannis Paschalidis, and Aldo Pacchiano. Post-training large language models for diverse high-quality responses. In *International Conference on Learning Representations*, volume 2026, pages 127946–127975, 2026.

[9] Farah Cherfaoui, Hachem Kadri, and Liva Ralaivola. Scalable ridge leverage score sampling for the nyström method. In *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4163–4167, 2022.

[10] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

[11] Yalun Dai, Yangyu Huang, Xin Zhang, Wenshan Wu, Chong Li, Wenhui Lu, Shijie Cao, Li Dong, and Scarlett Li. Data efficacy for language model training. *arXiv preprint arXiv:2506.21545*, 2025.

[12] DeepSeek-AI. Deepseek-v4: Towards highly efficient million-token context intelligence, 2026.

[13] Wenchao Du and Alan W Black. Boosting dialog response generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 38–43, 2019.

[14] Ziqing Fan, Siyuan Du, Shengchao Hu, Pingjie Wang, Li Shen, Ya Zhang, Dacheng Tao, and Yanfeng Wang. Combatting dimensional collapse in llm pre-training data via submodular file selection. In *The Thirteenth International Conference on Learning Representations*, 2025.

[15] Ziqing Fan, Yuqiao Xian, Yan Sun, Ke Shen, and Li Shen. Joint selection for large-scale pre-training data via policy gradient-based mask learning. In *The Fourteenth International Conference on Learning Representations*, 2026.

[16] Yuan Ge, Yilun Liu, Chi Hu, Weibin Meng, Shimin Tao, Xiaofeng Zhao, Mahong Xia, Zhang Li, Boxing Chen, Hao Yang, Bei Li, Tong Xiao, and JingBo Zhu. Clustering and ranking: Diversity-preserved instruction selection through expert-aligned quality estimation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 464–478, Miami, Florida, USA, 2024. Association for Computational Linguistics.

[17] Gene H. Golub and Charles F. Van Loan. *Matrix Computations - 4th Edition*. Johns Hopkins University Press, Philadelphia, PA, 2013.

[18] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, and et al. Ahmad Al-Dahle. The llama 3 herd of models, 2024.

[19] Yuling Gu, Oyvind Tafjord, Bailey Kuehl, Dany Haddad, Jesse Dodge, and Hannaneh Hajishirzi. Olmes: A standard for language model evaluations. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 5005–5033, 2025.

[20] Yuxian Gu, Li Dong, Hongning Wang, Yaru Hao, Qingxiu Dong, Furu Wei, and Minlie Huang. Data selection via optimal control for language models. In *The Thirteenth International Conference on Learning Representations*, 2025.

[21] Jie Hao, Rui Yu, Wei Zhang, Huixia Wang, Jie Xu, and Mingrui Liu. BLISS: A lightweight bilevel influence scoring method for data selection in language model pretraining. In *Forty-third International Conference on Machine Learning*, 2026.

[22] Animesh Jha, Ananjan Nandi, and Harshit Gupta. RL-guided data selection for language model finetuning. In *NeurIPS 2025 Workshop: Reliable ML from Unreliable Data*, 2025.

[23] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification, 2016.

[24] Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Yitzhak Gadre, Hritik Bansal, Etash Guha, Sedrick Scott Keh, Kushal Arora, et al. Datacomp-lm: In search of the next generation of training sets for language models. *Advances in Neural Information Processing Systems*, 37:14200–14282, 2024.

[25] Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, et al. Starcoder: may the source be with you! *Transactions on Machine Learning Research*, 2023.

[26] Fengze Liu, Weidong Zhou, Binbin Liu, Zhimiao Yu, Yifan Zhang, Haobin Lin, Yifeng Yu, Bingni Zhang, Xiaohuan Zhou, Taifeng Wang, et al. Quadmix: Quality-diversity balanced data selection for efficient llm pretraining. *arXiv preprint arXiv:2504.16511*, 2025.

[27] Shannon McCurdy. Ridge regression and provable deterministic ridge leverage score sampling. In *Advances in Neural Information Processing Systems*, 2018.

[28] Chenlin Ming, Chendi Qu, Mengzhang Cai, Qizhi Pei, Zhuoshi Pan, Yu Li, Xiaoming Duan, Lijun Wu, and Conghui He. Ideal: Data equilibrium adaptation for multi-capability language model alignment. In *The Fourteenth International Conference on Learning Representations*, 2026.

[29] Cameron Musco and Christopher Musco. Recursive sampling for the nyström method. In *Advances in Neural Information Processing Systems*, 2017.

[30] Thao Nguyen, Yang Li, Olga Golovneva, Luke Zettlemoyer, Sewoong Oh, Ludwig Schmidt, and Xian Li. Recycling the web: A method to enhance pre-training data quality and quantity for language models. In *Second Conference on Language Modeling*, 2025.

[31] Lalchand Pandia, Kanishka Misra, and Allyson Ettinger. Rethinking data selection: The importance of coverage over difficulty in generative fine-tuning, 2026.

[32] Amey P. Pasarkar and Adji Bousso Dieng. Cousins of the Vendi score: A family of similarity-based diversity metrics for science and machine learning. In *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, 2024.

[33] Saurabh Paul and Petros Drineas. Feature selection for ridge regression with provable guarantees. *Neural Computation*, 28(4):716–742, 2016.

[34] Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin A Raffel, Leandro Von Werra, Thomas Wolf, et al. The fineweb datasets: Decanting the web for the finest text data at scale. *Advances in Neural Information Processing Systems*, 37:30811–30849, 2024.

[35] Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Hamza Alobeidli, Alessandro Cappelli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. The refinedweb dataset for falcon llm: Outperforming curated corpora with web data only. *Advances in Neural Information Processing Systems*, 36:79155–79172, 2023.

[36] Ru Peng, Kexin Yang, Yawen Zeng, Junyang Lin, Dayiheng Liu, and Junbo Zhao. Dataman: Data manager for pre-training large language models. In *International Conference on Learning Representations*, volume 2025, pages 9747–9799, 2025.

[37] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

[38] Noveen Sachdeva, Benjamin Coleman, Wang-Cheng Kang, Jianmo Ni, Lichan Hong, Ed H. Chi, James Caverlee, Julian McAuley, and Derek Zhiyuan Cheng. How to train data-efficient LLMs. In *The Fourteenth International Conference on Learning Representations*, 2026.

[39] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8732–8740, 2020.

[40] Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. Social iqa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pages 4463–4473, 2019.

[41] Yunfan Shao, Linyang Li, Zhaoye Fei, Hang Yan, Dahua Lin, and Xipeng Qiu. Balanced data sampling for language model training with clustering. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 14012–14023, 2024.

[42] Han Shen, Pin-Yu Chen, Payel Das, and Tianyi Chen. Seal: Safety-enhanced aligned llm fine-tuning via bilevel data selection. In *The Thirteenth International Conference on Learning Representations*, 2025.

[43] Luca Soldaini, Rodney Kinney, and et al. Akshita Bhagia. Dolma: an open corpus of three trillion tokens for language model pretraining research. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15725–15788, Bangkok, Thailand, 2024. Association for Computational Linguistics.

[44] Katherine Stasaski and Marti A Hearst. Semantic diversity in dialogue with natural language inference. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 85–98, 2022.

[45] Dan Su, Kezhi Kong, Ying Lin, Joseph Jennings, Brandon Norick, Markus Kliegl, Mostofa Patwary, Mohammad Shoeybi, and Bryan Catanzaro. Nemotron-cc: Transforming common crawl into a refined long-horizon pretraining dataset. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2459–2475, 2025.

[46] Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. Commonsenseqa: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, 2019.

[47] Kimi Team. Kimi k2.5: Visual agentic intelligence, 2026.

[48] Hong Wang, Zhezheng Hao, Jian Luo, Chenxing Wei, Yao Shu, Lei Liu, Qiang Lin, Hande Dong, and Jiawei Chen. Scheduling your llm reinforcement learning with reasoning trees. In *International Conference on Learning Representations*, volume 2026, pages 154734–154753, 2026.

[49] Jiachen T. Wang, Prateek Mittal, Dawn Song, and Ruoxi Jia. Data shapley in one training run. In *The Thirteenth International Conference on Learning Representations*, 2025.

[50] Peiqi Wang, Yikang Shen, Zhen Guo, Matthew Stallone, Yoon Kim, Polina Golland, and Rameswar Panda. Diversity measurement and subset selection for instruction tuning datasets. In *ICLR 2025 Workshop on Navigating and Addressing Data Problems for Foundation Models*, 2025.

[51] Shaobo Wang, Xuan Ouyang, Tianyi Xu, Yuzheng Hu, Jialin Liu, Guo Chen, Tianyu Zhang, Junhao Zheng, Kexin Yang, Xingzhang Ren, Dayiheng Liu, and Linfeng Zhang. OPUS: Towards efficient and principled data selection in large language model pre-training in every iteration. In *Forty-third International Conference on Machine Learning*, 2026.

[52] Yudong Wang, Zixuan Fu, Jie Cai, Peijun Tang, Hongya Lyu, Yewei Fang, Zhi Zheng, Jie Zhou, Guoyang Zeng, Chaojun Xiao, Xu Han, and Zhiyuan Liu. Ultra-fineweb: Efficient data filtering and verification for high-quality llm training data, 2025.

[53] Maurice Weber, Daniel Y. Fu, Quentin Anthony, Yonatan Oren, Shane Adams, Anton Alexandrov, Xiaozhong Lyu, Huu Nguyen, Xiaozhe Yao, Virginia Adams, Ben Athiwaratkun, Rahul Chalamala, Kezhen Chen, Max Ryabinin, Tri Dao, Percy Liang, Christopher Ré, Irina Rish, and Ce Zhang. Redpajama: an open dataset for training large language models. *Advances in Neural Information Processing Systems*, 37, 2024.

[54] Alexander Wettig, Aatmik Gupta, Saumya Malik, and Danqi Chen. QuRating: Selecting high-quality data for training language models. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

[55] Alexander Wettig, Kyle Lo, Sewon Min, Hannaneh Hajishirzi, Danqi Chen, and Luca Soldaini. Organize the web: Constructing domains enhances pre-training data curation. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025.

[56] 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*, 36:69798–69818, 2023.

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

[58] Xianjun Yang, Shaoliang Nie, Lijuan Liu, Suchin Gururangan, Ujjwal Karn, Rui Hou, Madian Khabsa, and Yuning Mao. Diversity-driven data selection for language model tuning through sparse autoencoder, 2025.

[59] Yuming Yang, Yang Nan, Junjie Ye, Shihan Dou, Xiao Wang, Shuo Li, Huijie Lv, Tao Gui, Qi Zhang, and Xuan-Jing Huang. Measuring data diversity for instruction tuning: A systematic analysis and a reliable metric. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 18530–18549, 2025.

[60] Zichun Yu, Spandan Das, and Chenyan Xiong. Mates: Model-aware data selection for efficient pretraining with data influence models. *Advances in Neural Information Processing Systems*, 37:108735–108759, 2024.

[61] Zichun Yu, Fei Peng, Jie Lei, Arnold Overwijk, Wen-tau Yih, and Chenyan Xiong. Group-level data selection for efficient pretraining. In *Advances in Neural Information Processing Systems*, 2025.

[62] Zichun Yu and Chenyan Xiong. Repro: Training language models to faithfully recycle the web for pretraining. In *Forty-third International Conference on Machine Learning*, 2026.

[63] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pages 4791–4800, 2019.

[64] Dun Zeng, Yong Dai, Pengyu Cheng, Longyue Wang, Tianhao Hu, Wanshun Chen, Nan Du, and Zenglin Xu. On diversified preferences of large language model alignment. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 9194–9210, Miami, Florida, USA, 2024. Association for Computational Linguistics.

[65] Chi Zhang, Huaping Zhong, Kuan Zhang, Chengliang Chai, Rui Wang, Xinlin Zhuang, Tianyi Bai, Qiu Jiantao, Lei Cao, Ju Fan, Ye Yuan, Guoren Wang, and Conghui He. Harnessing diversity for important data selection in pretraining large language models. In *The Thirteenth International Conference on Learning Representations*, 2025.

[66] Jia Zhang, Yao Liu, Chen-Xi Zhang, Yi Liu, Yi-Xuan Jin, Lan-Zhe Guo, and Yu-Feng Li. Beyond single: A data selection principle for llm alignment via fine-grained preference signals. *arXiv preprint arXiv:2508.07638*, 2025.

[67] Jia Zhang, Chen-Xi Zhang, Yao Liu, Yi-Xuan Jin, Xiao-Wen Yang, Bo Zheng, Yi Liu, and Lan-Zhe Guo. D3: diversity, difficulty, and dependability-aware data selection for sample-efficient llm instruction tuning. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence*, IJCAI '25, 2025.

[68] Xuemiao Zhang, Xu Liangyu, Feiyu Duan, Yongwei Zhou, Sirui Wang, Rongxiang Weng, Jingang Wang, and Xunliang Cai. Preference curriculum: Llms should always be pretrained on their preferred data. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 21181–21198, 2025.

[69] Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, et al. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025.

Contents

1 Introduction **1**

2 Preliminaries **3**
- 2.1 Problem statement . . . 3
- 2.2 Recent progress . . . 3

3 Method **3**
- 3.1 Leverage score as diversity signal . . . 3
- 3.2 Algorithm . . . 4
- 3.3 How do leverage scores identify underrepresented semantics? . . . 5

4 Experiments **5**
- 4.1 Setup . . . 5
- 4.2 Main results . . . 6
  - 4.2.1 Lev outperforms existing diversity-based selection methods . . . 6
  - 4.2.2 Revisiting dimensional collapse: quality filtering demands diversity . . . 7
  - 4.2.3 Lev enables efficient large-scale diversity selection . . . 8
- 4.3 Ablation Study . . . 9

5 Conclusion **10**

A Related work **16**

B Supplementary mathematical analyses **16**
- B.1 Proof of Theorem 3.1 . . . 16
- B.2 Proof of Theorem 3.2 . . . 17
- B.3 Interpretation of the occurrence frequencies . . . 17
- B.4 The choice of the regularization parameter . . . 19

C Implementation details of experiments **19**
- C.1 Datasets . . . 19
- C.2 Embedding model . . . 20
- C.3 Baselines . . . 20
- C.4 Pretrained model configurations and training details . . . 22
- C.5 Benchmarks . . . 22
- C.6 Diversity metrics . . . 24

D Detailed experimental results **26**
- D.1 Diversity metrics . . . 26
- D.2 Downstream performance on CommonCrawl subset selection . . . 26
- D.3 Downstream performance on StarCoderData subset selection . . . 27
- D.4 Quality-filtered CC data shows inferior code performance relative to Lev-selected CC data . . . 28

E Additional discussions **29**
- E.1 Why does Lev achieve better diversity optimization results than DiSF? . . . 29

E.2 Insufficient recall of code data is a general problem among open-source data scorers 30
E.3 Detailed analysis of the time complexity of Lev and DiSF 31

A Related work

Data selection for pretraining. Research on LLM pretraining data selection largely evolved from Common-Crawl cleaning and filtering. Early pipelines such as C4 [37], RefinedWeb [35], RedPajama [53], SemDeDup [1], FineWeb [34], and Dolma-V1 [43] established the standard recipe of heuristic filtering and deduplication, defining the candidate corpora on which later selection methods operate. A more explicit formulation appeared in DSIR [57], which selects data by matching a target reference distribution (e.g., Wikipedia and books) via importance resampling in an n-gram feature space. Subsequent work moved toward model-based quality filtering, where a lightweight model is trained and scales selection to web-scale corpora, as in DCLM-fastText [24], FineWeb-Edu [34], QuRater [54], AskLLM [38] and DataMan [36]. More recent methods make selection increasingly training-aware: PDS [20], LQS [11] and PDPC [68] incorporate model's gradient or perplexity preference during training, while OPUS [51], MATES [60], Group-MATES [61], BLISS [21], and In-Run Data Shapley [49] use optimizer, model-state, data influence, or bilevel optimization signals to adapt selection to the evolving model. The quality-filtering methods can be further applied within each domain organized by [55] to strengthen the effectiveness. In parallel, another line improves data quality through rewriting rather than filtering alone, including REWIRE [30], RePro [62], and Nemotron-CC [45]. Finally, several recent works emphasize that pretraining data selection is fundamentally a set-level optimization problem, not only a pointwise scoring problem: DiSF [14] and Density [38] highlight the role of diversity, DoReMi [56] optimizes source mixture weights, and Quad [65], QuaDMix [26] and DATAMASK [15] jointly optimize multiple signals such as quality and diversity. These developments motivate studying diversified data selection as a corpus-level objective that balances quality, redundancy, and coverage under a fixed training budget.

Data selection beyond pretraining. Beyond pretraining, data selection in later LLM training stages has developed along three closely related directions. First, a substantial line of work studies subset selection for instruction tuning, where the main question is how to identify a small but high-value subset from a large instruction pool. Early and concurrent studies emphasize the role of diversity and representative coverage, including sparse autoencoder as representatives [58], DPP-based subset selection [50], robustness-oriented diversity analysis [4], systematic diversity measurement and NovelSum [59], semantic-space information gain maximization [7], diversity-preserved clustering-and-ranking pipelines [16], and joint criteria over diversity, difficulty, and dependability [67]. Recent evidence further suggests that, in generative fine-tuning, coverage can be more important than difficulty-based scoring alone [31]. Second, in alignment and post-training, data selection becomes more tightly coupled with preference structure, response diversity, and safety objectives. Representative examples include diversity-quality joint optimization in post-training [8], modeling diversified human preferences in alignment [64], selecting preference data using fine-grained preference signals [66], and safety-aware bilevel data selection for aligned fine-tuning [42]. Third, several recent works move beyond static subset choice and incorporate scheduling or mixture adaptation into the selection problem itself, such as RL-guided subset search for finetuning [22], curriculum-style scheduling for RL-based post-training [48], and dynamic mixture rebalancing across capabilities in multi-domain SFT [28]. Overall, compared with pretraining, these later-stage methods typically operate on much smaller but cleaner datasets, place stricter requirements on annotation quality and formatting consistency, and treat data selection less as web-scale noise filtering and more as controlled optimization over diversity, preference structure, curriculum, safety, and capability balance.

B Supplementary mathematical analyses

B.1 Proof of Theorem 3.1

Proof. Let $G = \frac{1}{n}X^\top X + \lambda I_d$. We consider the matrix determinant lemma [17], a basic relationship in linear algebra as follows.

$$\det(G + uv^\top) = \det(G) \cdot (1 + v^\top G^{-1} u).$$

Substituting u, v both with z, we have

$$\begin{aligned}\det(G + zz^\top) &= \det(G) \cdot (1 + z^\top G^{-1} z) \\ &= \det(G) \cdot (1 + \ell^\lambda(z; X)).\end{aligned}$$

Rearranging this yields

$$\ell^\lambda(z; X) = \frac{\det(G + zz^\top)}{\det(G)} - 1.$$

□

B.2 Proof of Theorem 3.2

Proof. Suppose the SVD of $\frac{1}{\sqrt{n}}X = U\Sigma V^\top$, where U and V are orthonormal matrices. So we have $U^\top U = VV^\top = I_d$. Then, the regularized Gram matrix should become

$$\begin{aligned}&\frac{1}{n}X^\top X + \lambda I_d \\ &= V\Sigma^\top U^\top U\Sigma V^\top + \lambda VV^\top \\ &= V\Sigma^\top \Sigma V^\top + \lambda V I_d V^\top \\ &= V(\Sigma^\top \Sigma + \lambda I_d)V^\top.\end{aligned}$$

Hence, the inverse matrix has the form of

$$\begin{aligned}&\left(\frac{1}{n}X^\top X + \lambda I_d\right)^{-1} \\ &= (V(\Sigma^\top \Sigma + \lambda I_d)V^\top)^{-1} \\ &= (V^\top)^{-1}(\Sigma^\top \Sigma + \lambda I_d)^{-1} V^{-1} \\ &= V(\Sigma^\top \Sigma + \lambda I_d)^{-1} V^\top.\end{aligned}$$

Finally, by definition we have

$$\begin{aligned}\ell^\lambda(z; X) &= z^\top \left(\frac{1}{n}X^\top X + \lambda I_d\right)^{-1} z \\ &= z^\top V(\Sigma^\top \Sigma + \lambda I_d)^{-1} V^\top z \\ &= [\tilde{z}_1, ..., \tilde{z}_d] \cdot \text{diag}\left\{\frac{1}{\sigma_1^2 + \lambda}, ..., \frac{1}{\sigma_d^2 + \lambda}\right\} \cdot [\tilde{z}_1, ..., \tilde{z}_d]^\top \\ &= \sum_{i=1}^{d} \frac{\tilde{z}_i^2}{\sigma_i^2 + \lambda}.\end{aligned}$$

□

B.3 Interpretation of the occurrence frequencies

We first prove that $\sum_{i=1}^{d} \sigma_i^2 = 1$.

Proof. Consider the trace of the Gram matrix, we have

$$\begin{aligned}
\mathrm{Tr}(1/n \cdot X^\top X) &= 1/n \cdot \mathrm{Tr}\left(\sum_{i=1}^{n} x_i x_i^\top\right) \\
&= 1/n \cdot \sum_{i=1}^{n} \mathrm{Tr}(x_i x_i^\top) \\
&= 1/n \cdot \sum_{i=1}^{n} \mathrm{Tr}(x_i^\top x_i) \\
&= 1/n \cdot \sum_{i=1}^{n} x_i^\top x_i \\
&= 1/n \cdot \sum_{i=1}^{n} \|x_i\|_2^2 \\
&= 1.
\end{aligned}$$

The third equality comes from the fact that the operator Tr is invariant to matrix commutation operation. The last equality holds due to the fact that all embedding vectors are normalized.

On the other hand, considering the SVD of matrix X, we have

$$\begin{aligned}
\mathrm{Tr}(1/n \cdot X^\top X) &= \mathrm{Tr}(V\Sigma^\top U^\top U \Sigma V^\top) \\
&= \mathrm{Tr}(V\Sigma^\top \Sigma V^\top) \\
&= \mathrm{Tr}(V^\top V\Sigma^\top \Sigma) \\
&= \mathrm{Tr}(\Sigma^\top \Sigma) \\
&= \sum_{i=1}^{d} \sigma_i^2.
\end{aligned}$$

Combining the above two relations, we have

$$\sum_{i=1}^{d} \sigma_i^2 = 1.$$

□

In the following elaborations, we use a simple case to show why $\{\sigma_i^2\}_{i=1}^d$ can be interpreted as frequencies. We already have $\sum_{i=1}^d \sigma_i^2 = 1$. Given the non-negativity of $\sigma_i^2 \geq 0$, the set $\{\sigma_i^2\}$ can be viewed as a valid frequency distribution. Next, we consider an extreme and trivial case. Suppose the embedding samples are $X = [x_1, \ldots, x_n]^\top$, but the dataset consists of only the orthonormal bases $\{v_1, \ldots, v_d\}$ of the space $\mathbb{R}^d$. Therefore, some samples in X acutally occur multiple times. Let the frequencies of $\{v_1, \ldots, v_d\}$ be $\{p_1, \ldots, p_d\}$. Then we have

$$\begin{aligned}
\frac{1}{n} X^\top X &= \frac{1}{n} \sum_{i=1}^{n} x_i x_i^\top \\
&= \sum_{i=1}^{d} p_i \, v_i v_i^\top.
\end{aligned}$$

Now consider the singular value decomposition of X, there exists some orthonormal matrix U such that

$$1/\sqrt{n} \cdot X = U \, \mathrm{diag}\{\sqrt{p_1}, \ldots, \sqrt{p_d}\} \, V^\top = U\Sigma V^\top.$$

Consequently, we obtain $\sigma_i^2 = p_i$. Extending this intuition to a general dataset X, we can intuitively regard the variance σ_i^2 of the principal component v_i as its occurrence frequency.

B.4 The choice of the regularization parameter

We show that λ should be a small positive number such that $\lambda \ll d^{-1}$.

We first consider the case of a perfectly diverse distribution where no more samples contribute to increasing the diversity of the data. In this case, all eigenvalues of the Gram matrix should be equal: $\sigma_1^2 = \cdots = \sigma_d^2 = 1/d$. If we ensure $\lambda \ll d^{-1}$, then every sample has a leverage score $\ell^\lambda(z) \approx d$, because

$$\ell^\lambda(z) = \sum_{i=1}^{d} \frac{\tilde{z}_i^2}{1/d + \lambda} \approx \sum_{i=1}^{d} d\tilde{z}_i^2 = d\|V^\top z\|_2^2 = d.$$

Now if the distribution is not perfectly diverse, then there exists a sample z such that $\ell^\lambda(z) > d$. Then there exist some principal semantic vectors such that $\sigma_i^2 < 1/d$, and z is represented mostly along these vectors. Specifically, if a principal semantic vector v_i is rarely covered in the selected data, then its corresponding σ_i^2 is approximately zero. Therefore, if we set λ as a very small numeric, then the uncovered semantic signals $\frac{\tilde{z}_i^2}{\sigma_i^2+\lambda} \approx \frac{\tilde{z}_i^2}{\lambda}$ will be significantly highlighted. Empirically, setting $\lambda = (100 \times d)^{-1}$ is sufficient for the above purposes.

C Implementation details of experiments

C.1 Datasets

FineWeb [34]. FineWeb is a large-scale English web corpus constructed from CommonCrawl and released as a cleaned and deduplicated pretraining dataset. FineWeb contains more than 18.5 trillion tokens of web text and is explicitly designed as a high-quality training corpus for large language models. Its pipeline applies large-scale web data cleaning and deduplication, with the stated goal of improving downstream language modeling performance while preserving the broad coverage and natural diversity of web text. The dataset is built using the `datatrove` processing framework and is intended as a strong web-data baseline for pretraining research.

For our own web data, we process a CommonCrawl dump with `datatrove` following similar pipeline steps in FineWeb, including heuristic filtering and deduplication while deliberately excluding quality filtering to preserve the original degree of data diversity. This results in a total amount of around 200B tokens web data.

Datatrove [34]. `datatrove` is a large-scale text data processing library developed for filtering, transforming, and deduplicating raw corpora such as CommonCrawl. The library provides a collection of modular processing blocks together with an extensible framework for adding custom functionality. Its pipeline abstraction is platform-agnostic and can run locally or on cluster environments such as Slurm, while maintaining relatively low memory usage. This design makes `datatrove` particularly suitable for reproducible construction of web-scale pretraining datasets, and FineWeb is one of its flagship use cases.

StarCoderData [25]. StarCoderData is the training corpus used for StarCoder and StarCoderBase. It is a large-scale collection containing 783GB of code across 86 programming languages, together with 54GB of GitHub Issues, 13GB of Jupyter notebooks in script and text-code formats, and 32GB of GitHub commits, amounting to approximately 250 billion tokens in total. The dataset is designed to support code language modeling at scale while going beyond plain source files by incorporating additional software-development artifacts such as issues and notebooks. As a result, it provides a broad substrate for studying code generation, code completion, and code understanding across multiple programming languages.

For our own code data, we exclusively consider six programming languages including Python, Java, JavaScript, Go, PHP, and Ruby, aiming to obtain clearer signals from the models' downstream performance. The results in a total amound of around 80B tokens code data.

C.2 Embedding model

Qwen3-Embedding-0.6B [69]. For the computation of the embedding vectors of the samples, we adopt Qwen3-Embedding-0.6B. This is a lightweight text embedding model from the Qwen3 Embedding series, built on the dense Qwen3 foundation models and designed for efficient semantic representation and retrieval tasks. Despite its compact 0.6B parameter size, it inherits the strong multilingual understanding and reasoning capabilities of the Qwen3 family, supporting over 100 languages and various programming languages. The model is optimized for a wide range of embedding applications such as text retrieval, code retrieval, clustering, and classification. It supports long-context inputs of up to 32k tokens and allows flexible user-defined embedding dimensions (from 32 up to 4096), enabling adaptation to different storage or latency constraints. For CommonCrawl data, we truncate each sample to the first 512 tokens and compute the corresponding embedding vectors. For StarCoderData corpus, we truncate each sample to the first 1500 tokens for embedding computation. The computed embeddings are then used for all diversity-based methods, including Lev, DiSF and Density.

C.3 Baselines

DiSF [14]. DiSF is a diversity-oriented data selection method designed to mitigate dimensional collapse in language model pretraining data selection. It operates by encouraging a more uniform eigenvalue spectrum of the data covariance matrix, thereby promoting richer and more expressive feature representations. We provide a concise and self-contained introduction of this method below.

Let $\mathcal{D} = \{z_1, \ldots, z_D\} \subset \mathbb{R}^d$ denote the full corpus of embeddings, and let $\mathcal{S} \subset \mathcal{D}$ denote the selected subset. The goal of DiSF is to construct $\mathcal{S}$ such that the representation space avoids dimensional collapse by encouraging a uniform spectrum of the feature covariance matrix.

Let the normalized feature covariance matrix of a batch $\mathcal{B}$ be defined as

$$C(\mathcal{B}) \;=\; \frac{1}{|\mathcal{B}|-1} \sum_{z \in \mathcal{B}} \tilde{z}\tilde{z}^\top,$$

where $\tilde{z} = (z-\mu)/\sigma$ denotes the embedding vector normalized with batch mean μ and batch standard deviation σ. Suppose C has eigenvalues $\{\lambda_i\}_{i=1}^d$. Then the following equality holds.

$$\mathrm{Var}(\{\lambda_i\}_{i=1}^d) \;=\; \sum_{i=1}^{d} \left(\lambda_i - \frac{1}{d} \sum_{i=1}^{d} \lambda_i \right)^2 \;=\; \|C\|_F^2 - d.$$

Equivalently, up to a constant scaling factor, minimizing the eigenvalue variance is equivalent to minimizing $\|C\|_F^2 - d$. This characterization provides the theoretical foundation for using the Frobenius norm as a proxy for spectral uniformity. Therefore, DiSF formulates data selection as a greedy maximization problem based on a marginal score function. Specifically, at each step, it selects the sample $z_i \in \mathcal{D} \setminus \mathcal{S}$ that maximizes

$$f(z_i) \;=\; \exp\!\big(-\big\|C(\mathcal{S} \cup \{z_i\})\big\|_F\big)\,,$$

which favors samples that reduce the Frobenius norm of the covariance matrix when added to the current set. In practice, the algorithm first separate the corpus into batches of size B and conduct greedy selection within each batch. Equivalently, the overall objective can be interpreted as approximately minimizing the spectral concentration of the selected set

$$\min_{\mathcal{S} \subset \mathcal{B},\, |\mathcal{S}| = \alpha B} \;\; \big\|C(\mathcal{S})\big\|_F,$$

where $\alpha \in (0, 1)$ is the selection ratio.

Density [38]. Density sampling is a data selection strategy based on density estimation in the embedding space. The Density method aims to construct a subset $\mathcal{S} \subset \mathcal{D}$ that improves coverage of the corpus by downweighting samples from highly populated regions and upweighting those from relatively sparse regions. This approach ensures that both dense and sparse regions of the data space are adequately represented, leading to improved coverage and density-uniformity in training data. We provide a concise and self-contained introduction of this method here.

Let $\mathcal{D} = \{x_1, \ldots, x_D\} \subset \mathbb{R}^d$ denote the full corpus represented in an embedding space. We approximate the sample density through a randomized hashing scheme based on Signed Random Projections (SRP), which provides an efficient sketch of local sample concentration.

Let R denote the number of independent hash functions and H denote the hash range. For each hash function $r \in \{1, \ldots, R\}$, the method draws random projection directions and maps each sample $x \in \mathbb{R}^d$ to a binary code according to the signs of its projected coordinates. This code is then converted into an integer bucket index

$$h_r(x) \in \{0, 1, \ldots, H-1\}.$$

Each hash function therefore partitions the embedding space into random cells, and samples that fall into the same bucket are treated as locally similar under that projection.

After hashing the corpus, a histogram sketch is constructed

$$S_{r,b} = \sum_{j=1}^{D} \mathbf{1}[h_r(x_j) = b],$$

which counts how many samples fall into bucket b under hash function r. The density score of a sample x_i is then estimated as the average bucket occupancy across all hash functions:

$$\text{score}(x_i) = \frac{1}{R} \sum_{r=1}^{R} S_{r,h_r(x_i)}.$$

A larger score indicates that x_i lies in a denser region of the embedding space.

Sampling probabilities are defined inversely with respect to this density estimate. Specifically, the unnormalized weight is

$$w(x_i) \propto \frac{1}{\text{score}(x_i)}.$$

Thus, samples from dense regions receive smaller weights, while samples from sparse regions receive larger weights. A subset of size k is then drawn without replacement according to these inverse-density weights.

In our implementation, the sketch uses $R = 1024$ hash functions and a hash range of $H = 262{,}144$ buckets. Each hash function is constructed from

$$m = \lceil \log_2 H \rceil$$

random hyperplanes, yielding an m-bit binary code (with $m = 18$ in our setting). The resulting projection tensor therefore has shape $R \times d \times m$.

For efficiency, the corpus is processed in mini-batches of size $20{,}000$ when constructing the histogram sketch $S \in \mathbb{N}^{R \times H}$, and all hashing operations are executed on GPU. Random projections are initialized with a fixed seed to ensure reproducibility, while a separate seed is used during sampling to obtain deterministic results.

DSIR [57]. DSIR (Data Selection via Importance Resampling) selects training samples by aligning the empirical distribution of the selected dataset with a target domain distribution. It leverages n-gram text features to estimate importance weights, favoring samples that better match domains such as Wikipedia or books. This method is particularly effective for domain adaptation and improving performance on targeted evaluation benchmarks. However, its targeted domain-matching behavior sometimes induces serious dimensional collpase problem.

DCLM-fastText [24]. DCLM-fastText is a quality-based data scoring method introduced in the DataComp-LM benchmark. It trains a lightweight fastText binary classifier to distinguish high-quality text from noisy web data, thereby enabling efficient large-scale filtering. Importantly, the positively labeled reference data is not drawn from conventional curated corpora such as Wikipedia or books. Instead, the classifier is trained on a mixture of **OpenHermes 2.5** and high-scoring posts from the Reddit **r/ExplainLikeImFive (ELI5)** community. Concretely, the DataComp-LM pipeline samples examples from OpenHermes 2.5, a synthetic instruction-following dataset for general-purpose assistants, and combines ELI5 question threads with their top-scoring answers selected

according to community karma scores. This positive set is therefore dominated by high-quality *instruction-formatted* and *question-answer* style text that spans a broad range of topics, rather than encyclopedic or book-style prose. Negative examples are sampled randomly from a RefinedWeb-style web corpus. After training, the classifier assigns each document a quality score based on its resemblance to this positive distribution, and the corpus is filtered by retaining the top-scoring documents. Owing to the efficiency of fastText, this approach scales well to web-scale corpora while yielding strong empirical performance.

C.4 Pretrained model configurations and training details

We train all 1.1B models using the Megatron framework on 60B pre-tokenized tokens. With a batch size of 1024 and a context length of 2048, this configuration results in 28,610 training iterations. Detailed model hyperparameters are provided in Table 4. The training and evaluation pipeline proceeds as follows: First, the selected corpus is converted into a pre-tokenized binary format compatible with Megatron. Pre-training is then performed under a distributed setup. Upon completion, the Megatron-format checkpoints are converted to the Hugging Face format to facilitate standardized evaluation. Finally, we assess the models using OLMES, which provides a unified pipeline for measuring performance across benchmarks.

Table 4: Detailed configurations of the 1.1B model of Llama 3.1 architecture.

Parameter name	Value
Parameter number	1,143,556,096
Hidden size	2048
Intermediate Hidden Size	4096
Context Len	2048
Heads	32
Layers	20
Minimum learning rate	4e-5
Maximum learning rate	4e-4
β_1 of optimizer	0.9
β_2 of optimizer	0.95
Warmup steps	1000
Batch size	1024×2048 tokens
Weight decay	0.1
Activation function	SwiGLU
Gradient clipping threshold	1.0

C.5 Benchmarks

ARC-Easy [10]. ARC-Easy is the easier subset of the AI2 Reasoning Challenge, a benchmark of grade-school science multiple-choice questions. Each example consists of a natural-language science question and a small set of answer candidates, and the task evaluates factual understanding together with elementary scientific and commonsense reasoning. According to the Hugging Face dataset card, ARC-Easy contains 2,251 training examples, 570 validation examples, and 2,376 test examples. It is commonly used to assess broad knowledge recall and relatively direct reasoning ability under a multiple-choice setup.

ARC-Challenge [10]. ARC-Challenge is the harder subset of the AI2 Reasoning Challenge. In contrast to ARC-Easy, its questions were selected to be difficult for simple retrieval-based and co-occurrence-based baselines, making the benchmark substantially more reasoning-intensive. The Hugging Face dataset card reports 1,119 training examples, 299 validation examples, and 1,172 test examples. The benchmark is designed to probe scientific question answering under more demanding compositional and commonsense reasoning conditions.

WinoGrande [39]. WinoGrande is a large-scale commonsense reasoning benchmark inspired by the Winograd Schema Challenge. It is formulated as a fill-in-the-blank task with two candidate options, where the model must resolve an ambiguous reference using commonsense knowledge rather than shallow lexical cues. The Hugging Face dataset card describes WinoGrande as a collection of roughly 44K problems. In the commonly used debiased configuration on Hugging Face, the dataset contains 9,248 training examples, 1,267 validation examples, and 1,767 test examples. This benchmark primarily evaluates pronoun resolution and referential commonsense reasoning.

HellaSwag [63]. HellaSwag is a benchmark for commonsense natural language inference and grounded sentence completion. Each instance presents a context and several candidate continuations, and the model must choose the most plausible continuation. The dataset is constructed adversarially so that the incorrect endings are highly confusable, making the task difficult for models despite being relatively easy for humans. The Hugging Face dataset card reports a total of 59,950 examples. HellaSwag is widely used to evaluate everyday event understanding, plausibility judgment, and broad commonsense inference.

PIQA [3]. PIQA, short for Physical Interaction Question Answering, is a benchmark for physical commonsense reasoning. Its questions are grounded in everyday goal-oriented situations, often involving tools, materials, and basic interactions with physical objects. The Hugging Face dataset card notes that PIQA is motivated by practical scenarios such as those found on instructional websites, and reports 16,000 training examples, 2,000 development examples, and 3,000 test examples. The benchmark primarily measures whether a model can reason about plausible physical actions and affordances.

Social IQa [40]. Social IQa is a multiple-choice question answering benchmark for social commonsense intelligence. Each example describes a social situation and asks a question about motives, reactions, or likely consequences, requiring the model to reason about human behavior and social implications. The Hugging Face dataset card states that the benchmark contains over 37,000 question-answer pairs. In the Hugging Face release, the labeled splits contain 33,410 training examples and 1,954 validation examples. Social IQa is intended to evaluate models' ability to infer mental states, intentions, and likely social outcomes.

CommonSenseQA [46]. CommonSenseQA is a multiple-choice commonsense question answering benchmark that requires diverse forms of everyday background knowledge. Each question is paired with one correct answer and four distractors, making it a five-way classification problem. The Hugging Face dataset card reports that the dataset contains 12,102 questions in total. The benchmark is designed to test broad commonsense reasoning, including taxonomic, causal, spatial, and functional knowledge.

Multilingual MBPP [5]. The multilingual MBPP benchmark used in our experiments corresponds to the MBPP portion of MultiPL-E. MultiPL-E extends code generation benchmarks to multiple programming languages by translating the original Python tasks into target languages while preserving executable test-based evaluation. According to the Hugging Face dataset card, MultiPL-E supports 22 programming languages. The underlying MBPP benchmark consists of 974 crowd-sourced programming problems designed for entry-level programmers, covering programming fundamentals and standard library usage. In our setting, we evaluate the six language subsets corresponding to the programming languages considered in the main experiments. This benchmark primarily measures multilingual code synthesis from natural-language problem descriptions.

HumanEval [6]. HumanEval is a code generation benchmark consisting of handwritten Python programming problems with unit-test-based evaluation. Each example contains a function signature, a natural-language docstring, a canonical solution, and a hidden correctness test. The Hugging Face dataset card reports that HumanEval contains 164 problems and only provides a test split. The benchmark is widely used to evaluate functional code generation accuracy under an execution-based metric, and is particularly useful for assessing whether a model can synthesize correct code from concise natural-language specifications.

C.6 Diversity metrics

Let $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$ denote the embedding vectors of n samples where each vector has norm $\|x_i\|_2 = 1$. Let $d(x_i, x_j)$ denote the L_2 distance between x_i and x_j. We denote by $N_k(x_i)$ the k-th nearest neighbor of x_i under $d(\cdot, \cdot)$. We explicitly define the seven diversity metrics below.

CluIne [13]. Let $\{C_k\}_{k=1}^K$ be a partition of X obtained by the clustering algorithm, K-means. The cluster inertia is defined as

$$\mathrm{CluIne}(X) \;=\; \sum_{k=1}^{K} \sum_{x_i \in C_k} \|x_i - \mu_k\|_2^2, \quad \mu_k = \frac{1}{|C_k|} \sum_{x_j \in C_k} x_j.$$

This metric quantifies the within-cluster variance of the dataset under the induced partition. A larger value indicates that samples are more dispersed relative to their assigned centroids, reflecting greater global spread across clusters. However, its interpretation depends on the choice of K and the clustering procedure, and it does not explicitly capture fine-grained pairwise relationships or local redundancy. For all metrics that incorporates K-means, we set $K = 25$ for the training of clustering.

LogDet [50]. The log-determinant diversity is defined as

$$\mathrm{LogDet}(X) \;=\; \frac{1}{n} \log\det\big(L\big),$$

where L is the element-wise exponential of the similarity matrix of X such that $L_{ij} = \exp(2x_i^\top x_j)$. This metric measures the volume spanned by the sample embeddings in the induced feature space. A larger value indicates that the samples are more linearly independent and less redundant, thereby capturing global diversity through the geometry of the representation space.

CosDist [59]. The cosine-distance-based diversity is defined as

$$\mathrm{CosDist}(X) \;=\; \frac{1}{n(n-1)} \sum_{i \neq j} \big(1 - x_i^\top x_j\big)\,.$$

This metric aggregates pairwise dissimilarities between all samples based on cosine distance. It promotes global separation among samples by assigning larger values to configurations where embeddings are mutually dissimilar. Nevertheless, it treats all pairwise relationships uniformly and may overemphasize contributions from distant pairs while underrepresenting local geometric structure.

Entropy. Let $\lambda_1, \ldots, \lambda_n$ denote the eigenvalues of the Gram matrix $G = \frac{1}{n} X^\top X$. The spectral entropy is defined as

$$\mathrm{Entropy}(X) \;=\; -\sum_{i=1}^{n} \lambda_i \log \lambda_i.$$

This metric captures how evenly the variance of the dataset is distributed across principal components. A higher entropy indicates that information is spread across multiple directions in the embedding space, implying a more isotropic and diverse representation. In contrast, low entropy corresponds to concentration along a small number of dominant components, which suggests redundancy.

KNN [44]. The KNN-based diversity is defined as

$$\mathrm{KNN}(X) \;=\; \frac{1}{n} \sum_{i=1}^{n} d\big(x_i, N_k(x_i)\big),$$

typically with $k = 1$. This metric focuses on local geometry by measuring the distance between each sample and its nearest neighbors. Larger values indicate that samples are locally isolated, which reflects stronger sample-level uniqueness. Compared to global aggregation methods, this formulation is more sensitive to redundancy among nearby samples.

NovelSum [59]. The NovelSum metric defines dataset diversity as the sum of sample-level novelty scores

$$\mathrm{NovelSum}(X) \;=\; \sum_{i=1}^{n} v(x_i), \quad v(x_i) \;=\; \sum_{j \neq i} w_{ij}^{\alpha}\, \Delta(x_i, x_j),$$

where $w_{ij} = \phi(\pi_i(j))$ is a monotonically decreasing function of the rank $\pi_i(j)$ of x_j in the sorted list of distances from x_i. For instance, $\pi_i(j) = 1$ indicates that x_j is the nearest neighbor of x_i. Typically, we set $\phi(\pi_i(j)) = \frac{1}{\pi_i(j)}$. The distance is further defined as

$$\Delta(x_i, x_j) \;=\; \sigma(x_j)^{\beta}\, d(x_i, x_j),$$

where $d(\cdot, \cdot)$ is the L_2 distance metric and $\sigma(x_j)$ is a density factor that approximates the inverse local density around x_j. For example, we use

$$\sigma(x_j) = \frac{1}{\sum_{k=1}^{K} d(x_j, N_k(x_j))}.$$

This formulation integrates proximity-aware weighting and density-aware scaling, thereby emphasizing differences with nearby samples while accounting for heterogeneous information density in the space. As a result, NovelSum provides a principled estimate of each sample's unique contribution to the overall dataset.

Vendi score [32]. Let $\tilde{\lambda}_1, \ldots, \tilde{\lambda}_n$ denote the normalized eigenvalues of the sample similarity matrix $XX^{\top}$ (not the Gram matrix, thus different from Entropy). The Vendi Score of order 1 has an equivalent definition of

$$\mathrm{Vendi}(X) \;=\; \exp\left(-\sum_{i=1}^{n} \tilde{\lambda}_i \log \tilde{\lambda}_i \right).$$

This metric can be interpreted as the effective number of components contributing to the representation. It transforms spectral entropy into a quantity with a more direct interpretation, where larger values correspond to a richer and more uniformly distributed structure in the embedding space.

Discussion. These metrics can be broadly categorized into two classes. Distance-based metrics, including CosDist, KNN, and CluIne, primarily quantify diversity through pairwise or local separations in the embedding space, emphasizing sample-level uniqueness or dispersion. Spectral and distributional metrics, including Entropy, Vendi Score, and LogDet, instead characterize diversity through the global structure of the dataset as reflected in the eigen-spectrum of the Gram matrix. NovelSum integrates both perspectives by jointly modeling local proximity, inter-sample differences, and heterogeneous density, thereby providing a more comprehensive characterization of dataset diversity.

D Detailed experimental results

D.1 Diversity metrics

In Figure 4, we normalize the results of each diversity metric by applying a linear mapping from $[\min, \max]$ to $[0.30, 1.00]$ to compare the relative values of different methods. The original values are reported in Table 5. Lev consistently achieves the best performance.

Table 5: Dataset diversity comparison at 40% selection ratio. Higher is better.

Method	CluIne	LogDet	CosDist	Entropy	KNN	NovelSum	Vendi
Lev (ours)	**0.767**	**1.638**	**0.853**	**5.317**	**1.076**	**0.640**	**203.805**
DiSF	*0.757*	*1.623*	0.832	*5.229*	*1.071*	0.626	*186.629*
Density	0.744	1.612	0.833	5.181	1.056	*0.627*	177.802
DCLM-fastText	0.747	1.611	*0.835*	5.187	1.055	0.626	178.855
DSIR	0.732	1.599	0.821	5.122	1.053	0.611	167.730
Random	0.732	1.601	0.821	5.124	1.052	0.619	168.060

D.2 Downstream performance on CommonCrawl subset selection

We provide full results of the downstream performance over seven language understanding tasks across all baseline methods and Lev for the second evaluation in section 4.2.1. The evaluated datasets are selected from CommonCrawl corpus using various diversification strategies at 40% and 50% selection ratios.

Table 6: Downstream performance at 40% and 50% web data selection.
Accuracies (%) are Reported. Higher is Better.

Method	40% web data selection							
	ARC-e	ARC-c	WG	HS	PIQA	SIQA	CSQA	Mean ↑
Random	$48.50_{(5)}$	$27.82_{(4)}$	$52.09_{(3)}$	$46.50_{(3)}$	$\mathbf{69.40}_{(1)}$	$\mathbf{46.30}_{(1)}$	$43.00_{(3)}$	$47.66_{(4)}$
DSIR	$49.10_{(4)}$	$29.69_{(2)}$	$51.22_{(4)}$	$39.90_{(5)}$	$64.40_{(5)}$	$43.70_{(5)}$	$35.87_{(5)}$	$44.84_{(5)}$
Density	$51.50_{(3)}$	$28.50_{(3)}$	$\mathbf{53.12}_{(1)}$	$47.40_{(2)}$	$68.50_{(3)}$	$44.40_{(4)}$	$42.42_{(4)}$	$47.98_{(3)}$
DiSF	$53.70_{(2)}$	$\mathbf{30.12}_{(1)}$	$51.07_{(5)}$	$45.20_{(4)}$	$67.80_{(4)}$	$45.60_{(2)}$	$44.96_{(2)}$	$48.35_{(2)}$
Lev (ours)	$\mathbf{54.20}_{(1)}$	$27.47_{(5)}$	$52.72_{(2)}$	$\mathbf{48.60}_{(1)}$	$69.30_{(2)}$	$45.00_{(3)}$	$\mathbf{45.05}_{(1)}$	$\mathbf{48.91}_{(1)}$
Method	50% web data selection							
	ARC-e	ARC-c	WG	HS	PIQA	SIQA	CSQA	Mean ↑
Random	$48.50_{(5)}$	$27.82_{(4)}$	$52.09_{(4)}$	$46.50_{(3)}$	$\mathbf{69.40}_{(1)}$	$\mathbf{46.30}_{(1)}$	$\mathbf{43.00}_{(1)}$	$47.66_{(3)}$
DSIR	$\mathbf{53.50}_{(1)}$	$27.73_{(5)}$	$\mathbf{53.04}_{(1)}$	$42.00_{(5)}$	$65.20_{(5)}$	$44.30_{(5)}$	$42.26_{(5)}$	$46.86_{(5)}$
Density	$51.70_{(4)}$	$27.99_{(3)}$	$50.75_{(5)}$	$\mathbf{48.20}_{(1)}$	$66.40_{(4)}$	$45.00_{(3)}$	$42.51_{(3)}$	$47.51_{(4)}$
DiSF	$52.90_{(2)}$	$28.16_{(2)}$	$52.49_{(2)}$	$46.40_{(4)}$	$67.80_{(3)}$	$44.50_{(4)}$	$42.59_{(2)}$	$47.83_{(2)}$
Lev (ours)	$52.70_{(3)}$	$\mathbf{29.35}_{(1)}$	$52.33_{(3)}$	$47.10_{(2)}$	$68.30_{(2)}$	$45.10_{(2)}$	$42.34_{(4)}$	$\mathbf{48.17}_{(1)}$

D.3 Downstream performance on StarCoderData subset selection

We provide full results of the code bits-per-byte over seven code-related tasks across all the diversity-based selection methods. The evaluated dataset are selected from StarCoderData corpus using various diversification strategies at 40% and 50% selection ratios.

Table 7: Downstream performance at 50% and 40% code data selection. Bits-per-byte($\times 10^{-2}$) is reported. Lower is better.

DCLM-fastText for the web data.								
Method	50% code data selection							
	Python	Go	PHP	Ruby	Java	JS	HumanEval	Mean ↓
Random	$92.65_{(3)}$	$59.22_{(2)}$	$\mathbf{52.15}_{(1)}$	$98.89_{(4)}$	$38.90_{(2)}$	$68.34_{(4)}$	$53.66_{(3)}$	$66.26_{(4)}$
Density	$92.28_{(2)}$	$61.26_{(3)}$	$53.71_{(3)}$	$91.98_{(2)}$	$40.68_{(4)}$	$63.55_{(2)}$	$57.44_{(4)}$	$65.84_{(3)}$
DiSF	$96.19_{(4)}$	$61.58_{(4)}$	$54.90_{(4)}$	$\mathbf{91.08}_{(1)}$	$\mathbf{38.48}_{(1)}$	$64.35_{(3)}$	$\mathbf{53.22}_{(1)}$	$65.68_{(2)}$
Lev (ours)	$\mathbf{88.23}_{(1)}$	$\mathbf{57.26}_{(1)}$	$53.34_{(2)}$	$92.23_{(3)}$	$40.60_{(3)}$	$\mathbf{60.69}_{(1)}$	$53.27_{(2)}$	$\mathbf{63.66}_{(1)}$
Method	40% code data selection							
	Python	Go	PHP	Ruby	Java	JS	HumanEval	Mean ↓
Random	$92.65_{(4)}$	$\mathbf{59.22}_{(1)}$	$52.15_{(4)}$	$98.89_{(4)}$	$38.90_{(2)}$	$68.34_{(4)}$	$\mathbf{53.66}_{(1)}$	$66.26_{(4)}$
Density	$91.61_{(2)}$	$61.67_{(2)}$	$\mathbf{50.25}_{(1)}$	$93.57_{(2)}$	$41.22_{(4)}$	$62.23_{(2)}$	$56.07_{(4)}$	$65.23_{(2)}$
DiSF	$91.97_{(3)}$	$62.58_{(3)}$	$51.26_{(2)}$	$\mathbf{93.11}_{(1)}$	$39.64_{(3)}$	$64.12_{(3)}$	$54.96_{(3)}$	$65.38_{(3)}$
Lev (ours)	$\mathbf{90.35}_{(1)}$	$66.65_{(4)}$	$51.87_{(3)}$	$94.99_{(3)}$	$\mathbf{38.43}_{(1)}$	$\mathbf{58.76}_{(1)}$	$54.11_{(2)}$	$\mathbf{65.02}_{(1)}$
Leverage Sampling for the web data.								
Method	50% code data selection							
	Python	Go	PHP	Ruby	Java	JS	HumanEval	Mean ↓
Random	$94.59_{(4)}$	$60.57_{(3)}$	$52.60_{(4)}$	$92.30_{(3)}$	$38.74_{(4)}$	$66.27_{(4)}$	$56.65_{(2)}$	$65.96_{(4)}$
Density	$89.78_{(2)}$	$\mathbf{58.89}_{(1)}$	$52.25_{(3)}$	$\mathbf{87.96}_{(1)}$	$\mathbf{36.64}_{(1)}$	$59.45_{(2)}$	$59.10_{(4)}$	$63.44_{(2)}$
DiSF	$92.51_{(3)}$	$62.17_{(4)}$	$51.16_{(2)}$	$93.40_{(4)}$	$38.60_{(3)}$	$61.90_{(3)}$	$56.92_{(3)}$	$65.24_{(3)}$
Lev (ours)	$\mathbf{89.58}_{(1)}$	$60.11_{(2)}$	$\mathbf{50.26}_{(1)}$	$90.69_{(2)}$	$37.60_{(2)}$	$\mathbf{59.41}_{(1)}$	$\mathbf{51.25}_{(1)}$	$\mathbf{62.70}_{(1)}$
Method	40% code data selection							
	Python	Go	PHP	Ruby	Java	JS	HumanEval	Mean ↓
Random	$89.14_{(2)}$	$63.69_{(4)}$	$52.39_{(2)}$	$96.05_{(4)}$	$39.88_{(3)}$	$\mathbf{61.87}_{(1)}$	$56.48_{(3)}$	$65.64_{(2)}$
Density	$94.44_{(3)}$	$\mathbf{61.98}_{(1)}$	$53.84_{(3)}$	$94.44_{(3)}$	$39.93_{(4)}$	$62.18_{(2)}$	$57.91_{(4)}$	$66.39_{(4)}$
DiSF	$94.70_{(4)}$	$62.04_{(2)}$	$54.22_{(4)}$	$\mathbf{92.48}_{(1)}$	$\mathbf{39.09}_{(1)}$	$63.65_{(3)}$	$53.99_{(2)}$	$65.74_{(3)}$
Lev (ours)	$\mathbf{88.93}_{(1)}$	$63.34_{(3)}$	$\mathbf{52.36}_{(1)}$	$92.92_{(2)}$	$39.15_{(2)}$	$63.96_{(4)}$	$\mathbf{50.56}_{(1)}$	$\mathbf{64.46}_{(1)}$

D.4 Quality-filtered CC data shows inferior code performance relative to Lev-selected CC data

To isolate the impact of quality-filtered CC on code performance, we conduct a two-stage analysis. First, pretraining solely on CC data reveals that quality-filtered CC yields markedly higher code BPB than Lev-selected CC (Table 2a), indicating substantial loss of code-related content. We then augment both CC subsets with dedicated code data from StarCoderData and retrain. Despite this supplementation, the quality-filtered mixture continues to underperform in code evaluation (Table 2b), confirming that the induced code degradation persists. This robustness holds across two additional code datasets selected by Lev in Table 8.

Table 8: Comparison between DCLM-fastText and leverage score sampling under different code data. BBP($\times 10^{-2}$) for Code Mean and Accuracy(%) for Base Mean.

Code Data	Web Data	Code Mean ↓	Base Mean ↑
Random	Lev (top 40%)	**65.64**$_{(1)}$	49.72$_{(3)}$
	Lev (top 50%)	65.96$_{(2)}$	49.85$_{(2)}$
	Qual (top 40%)	66.26$_{(3)}$	**50.46**$_{(1)}$
	Random	66.48$_{(4)}$	48.76$_{(4)}$
Lev (top 50%)	Lev (top 40%)	**62.48**$_{(1)}$	49.90$_{(2)}$
	Lev (top 50%)	62.70$_{(2)}$	49.17$_{(4)}$
	Qual (top 40%)	63.66$_{(3)}$	**51.53**$_{(1)}$
	Random	64.46$_{(4)}$	49.28$_{(3)}$
Lev (top 40%)	Lev (top 40%)	64.46$_{(2)}$	49.96$_{(3)}$
	Lev (top 50%)	**63.55**$_{(1)}$	50.04$_{(2)}$
	Qual (top 40%)	65.02$_{(4)}$	**50.10**$_{(1)}$
	Random	64.75$_{(3)}$	49.81$_{(4)}$

E Additional discussions

E.1 Why does Lev achieve better diversity optimization results than DiSF?

We answer an equivalent question: *Why the leverage-score objective should be preferable to DiSF's Frobenius objective?* This question can be answered theoretically. We show that Lev is more sensitive in detecting diverse samples. We demonstrate this in two steps.

Step 1. Recall that in our Theorem 3.2, $\ell(z) = \sum \tilde{z}_i{}^2/\sigma_i^2$, where $\tilde{z}_i = v_i^\top z$ is the projection of z on the i-th principal vector of $G = X^\top X/n$ and σ_i^2 be the eigenvalues of G. We omit λI_d for leverage scores for brevity, since this does not affect our analysis. The criterion $F(z) = -||(1-1/n)G + 1/n \cdot zz^\top||_F^2$ that DiSF uses has a similar statement. We briefly deduce as follows.

Let $f(\eta) = -||(1-\eta)G + \eta zz^\top||_F^2$. The directional derivative of the matrix function $M(G) := -||G||_F^2$ toward direction $zz^\top$ is exactly $\frac{df}{d\eta}|_{\eta=0}$. This quantity describes how fast the candidate sample z increases the DiSF criterion $M(G)$.

Let $E = zz^\top - G$, then

$$f(\eta) = -||G + \eta E||_F^2 = -Tr((G+\eta E)^2) = -Tr(G^2) - 2\eta Tr(GE) - o(\eta^2).$$

Hence, $f'(\eta) = -2Tr(GE) - o(\eta)$. Let $c = 2Tr(G^2)$ be the constant, we obtain

$$\begin{aligned}\frac{df}{d\eta}|_{\eta=0} &= -2Tr(GE) = -2Tr(G(zz^\top - G)) = -2Tr(Gzz^\top) + 2Tr(G^2) \\ &= -2Tr(z^\top Gz) + c = -2z^\top Gz + c.\end{aligned}$$

Thus, the criterion in DiSF is equivalent to

$$d(z) = -z^\top Gz,$$

which we refer to as DiSF score. Applying similar deductions in our Theorem 3.2 gives

$$d(z) = -\sum_{i=1}^{d} \sigma_i^2 \tilde{z}_i^2 \quad (*).$$

Step 2. Now, we show that DiSF can be less sensitive than Lev for capturing samples of rare semantics (i.e. diverse samples) in a very broad scenario. We illustrate this through a case study, in which we consider only three principal components of the Gram matrix for brevity.

Suppose for the currently selected samples X, we have three principal semantic vectors v_1, v_2, v_3, and their corresponding eigenvalues be $(\sigma_1^2, \sigma_2^2, \sigma_3^2) = (3/4, 1/4 - 10^{-2}, 10^{-2})$. This means that the third semantic is extremely rare. Considering maximizing the Shannon entropy of the eigenvalues (e.g., Vendi Score), samples that allocate more weights on the third dimension provide more diversity gains and should be preferred.

We illustrate the difference of the two methods with an example. Suppose two candidate samples S1 and S2, and suppose the $(\tilde{z}_1^2, \tilde{z}_2^2, \tilde{z}_3^2)$ of the two samples are

$$\begin{aligned}\text{S1}: &\quad (0, 1/2, 1/2) \\ \text{S2}: &\quad (1/6, 0, 5/6)\end{aligned}$$

Because S2 has more weight on the third dimension, we expect a good algorithm should prefer S2 over S1.

For DiSF:

$$\begin{aligned}d(\text{S1}) &= -(1/4 - 10^{-2}) \cdot 1/2 - 10^{-2} \cdot 1/2 = -1/8 \\ d(\text{S2}) &= -(3/4) \cdot 1/6 - 10^{-2} \cdot 5/6 = -1/8 - 10^{-2} \cdot 5/6\end{aligned}$$

Thus $d(\text{S1}) > d(\text{S2})$, DiSF mistakenly select S1 in this case.

For Lev:

$$\ell(\text{S1}) = 1/(1/4 - 10^{-2}) \cdot 1/2 + 1/10^{-2} \cdot 1/2 \approx 52$$

$$\ell(\text{S2}) = 1/(3/4) \cdot 1/6 + 1/10^{-2} \cdot 5/6 \approx 83.6$$

Thus $\ell(\text{S2}) > \ell(\text{S1})$, Lev correctly select S2.

This sensitivity is rooted in that Lev uses $1/\sigma_i^2$ as the component weights for $\tilde{z}_i^2$ (Theorem 3.2), which are more sensitive than the linear weights $-\sigma_i^2$ in DiSF (see Eq. $(*)$), especially when σ_i^2 is around 0, $1/\sigma_i^2$ goes to infinity. For DiSF with the linear weight, if a sample (e.g. sample S2) also allocate some semantic weights on the frequently occurred directions (measured by the $-\sigma_1^2\tilde{z}_1^2 - \sigma_2^2\tilde{z}_2^2$ value of sample S2), the contribution of the rare semantic ($-\sigma_3^2\tilde{z}_3^2$) might not provide signals significant enough to make the sample selected. In contrast, the leverage weight $1/\sigma_i^2$ that Lev uses is able to significantly amplify the impact of the rare semantic ($\tilde{z}_3^2/\sigma_3^2 \gg 1$) to make it selected.

This example actually covers a very broad scenario where DiSF could miss these desired samples. Because through the selection process, *as the size of the selected subset X gets larger, samples that partly contains rare semantics will also contain some other semantics that have occurred in X with very high probability*. In this case, Lev is still able to distinguish samples that are actually more diverse, but DiSF could miss them, which is exactly what the given example shows.

E.2 Insufficient recall of code data is a general problem among open-source data scorers

We first clarify the fairness of the quality-filtering comparison in the code setting, showing that the quality-filtering baseline *DCLM-fastText* in our paper does include code data in its training set.

Specifically, DCLM-fastText is trained with OpenHermes-2.5 (OH2.5) and reddit ELI5 as positive samples, and random documents from RefinedWeb as negative samples. OH2.5 is a high-quality QA-style dataset aggregating multiple instrunction-tuning datasets covering a diverse range of topics, which includes *glaive-code-assist*, a dataset of code problems and solutions covering python, c/c++, java and many other programming languages. Statistic shows that code data occupies 18.2% of all the approximately 1M samples in OH2.5.

So DCLM-fastText does include a proportion of code data in its training set. Yet, the recall of code data using DCLM-fastText is not so strong as Lev, which supports our claim that quality-filtering demands diversity for potentially higher recall of data in other domains.

To further substantiate our claim, we design a heurstic classifier to identify the code-related docs in each subset selected by different methods. Surprisingly, we find that the insufficent recall of code data may be a more general problem among open-source data scorers. We consider three additional quality scorers: dolma3-fasttext, qurater and nemocurater, which more or less include code data as positive examples. For subsets (select 40% from around 30M docs) using different methods, we compare two quantities: the number of code-related docs in the selected subsets, and the code modeling ability (bit-per-byte on code benchmark) of the models trained on these subsets:

Table 9: Recall of code-related docs and corresponding code modeling ability under different quality scorers. BBP($\times 10^{-2}$) for Code Mean.

Method	Number of Code Docs	Reduction of Code Docs relative to Lev	Code Mean ↓
Lev	1879	-0.00%	62.48
DCLM-fastText	1544	-17.83%	63.66
NemoCurater	1378	-26.66%	63.82
Random	1003	-46.62%	64.46
Qurater	837	-55.46%	64.01
Dolma3-fastText	712	-62.11%	65.64

In this table, we find that the code modeling ability is positively correlated to the recall of code-related docs. The more code-related docs are retrieved, the better the code modeling ability of the pre-trained model will be.

Results show that lev retrieves the most code-related content and the corresponding pretrained model shows the best code modeling ability, while quality filters generally retrieve less code-related content

and the corresponding pre-trained models show worse code modeling ability than Lev. Two of the quality filters retain even less code-related content than Random (i.e. even more biased away from the original distribution).

To conclude, a number of quality filters, albeit include code-domain data in their training sets, do not necessarily achieve recall as high as Lev. This demonstrates our claim that quality filters need diversity enhancement to retrieve more code-domain data.

E.3 Detailed analysis of the time complexity of Lev and DiSF

We provide a more detailed algorithmic comparison between Lev and DiSF. We show that the speedup comes from the faster computation of the leverage scores than that of DiSF scores.

Both algorithms follow the simple framework of greedy selection, the procedure of which is:

1. Divide the corpus data into batches of size B.
2. For every batch $\mathcal{B}$, it repeatedly select samples as follows:
   (a) Compute the DiSF/Lev scores (with respect to the currently selected samples) for every unselected sample in the batch.
   (b) Select the top-k samples and add them to the selected subset.
3. Repeat Step 2 until the number of the selected samples reaches the budget.

Hence, the difference of these two algorithm only consists in Step 2(a), namely the computation of DiSF scores or Lev scores. Next, we compare the time complexity of computing Lev and DiSF in Step 2(a).

Suppose n is the number of the currently selected samples, and $X \in \mathbb{R}^{n\times d}$ is the corresponding sample matrix. Suppose the sample matrix of the currently unselected samples is $Z \in \mathbb{R}^{(B-n)\times d}$, where B is the batch size and $0 \leq n \leq B$.

Step 2(a) of Lev proceeds as follows.

1. Compute $G = 1/n \cdot X^\top X + \lambda I_d$, which requires $O(nd^2)$ computation.
2. Compute the lower triangular matrix L, which is the Cholesky decomposition of $G = LL^\top$. This requires $O(d^3)$ computation.
3. Solving for matrix $W \in \mathbb{R}^{(B-n)\times d}$ such that $LW^\top = Z^\top$. Because L is a lower triangular matrix, solving this equations using Gaussian elimination only takes $O((B-n)\times d^2)$ steps.
4. Compute the ℓ_2 norm of ever row of W, which is exactly the leverage score of every sample in Z. This step requires $O((B-n)\times d)$ steps.

Because usually $B \geq d$, substep 3 of the above is dominant. Hence, the time complexity of Step 2(a) in the greedy selection for Lev is $O(Bd^2)$. The key step is that through the Cholesky decomposition over G, the leverage scores of all the samples can be computed all together through the above substep 3 and 4.

Step 2(a) of DiSF proceeds as follows.

1. For every unselected sample z, which is some row of matrix $Z \in \mathbb{R}^{(B-n)\times d}$:
   (a) Append z to the tail of X. We denote the new matrix as $\hat{X}$. This is $O(1)$.
   (b) Normalize every column of $\hat{X}$ (requires computation of mean and variance value of every column). This is $O(n \times d)$.
   (c) Compute $C = 1/n \cdot \hat{X}^\top \hat{X}$. This requires $O(n \times d^2)$.
   (d) Compute the Frobenius norm of C: $||C||_F^2$. This requires only $O(d^2)$.

The most computationally intensive step is Step 1(c), which requires $O(n \times d^2)$. Because we have $B-n$ unselected samples, the overall time complexity of Step 2(a) in the greedy selection for DiSF is $O((B-n)nd^2)$.

Because n is enumerated from $1, 2, ...\lceil \alpha B/k \rceil$, where α is the selection ratio (e.g., $\alpha = 40\%$), by summing up over n, we obtain:

- For Lev, Step 2 requires $\sum_n O(Bd^2) = O(\alpha B^2 d^2) = O(B^2 d^2)$.
- For DiSF, Step 2 requires $\sum_n O((B-n)nd^2) = \sum_n O(Bnd^2) = O(\alpha^2 B^3 d^2) = O(B^3 d^2)$.

To conclude, the time complexity of DiSF is of one order over B higher than Lev. Hence, from an algorithmic perspective, Lev is more efficient than DiSF.

Remark. The column-wise normalization step in DiSF is necessary. Without normalization, it becomes $\sum_{i=1}^{d}(\lambda_i - \bar{\lambda})^2 = ||C||_F^2 - (Tr(C))^2/d = ||C||_F^2 - (\sum_{i=1}^{d}\lambda_i)^2/d = ||C||_F^2 - d(\bar{\lambda})^2$. Hence, $||C||_F^2 = \sum_{i=1}^{d}\lambda_i^2$. Without normalization, minimizing $||C||_F^2$ becomes minimizing the eigenvalue magnitude instead of optimizing eigenvalue uniformity. In contrast, Lev does not require column-wise normalization over the Gram matrix. By conducting Cholesky decomposition over the Gram matrix only once, the leverage scores of all the samples can be computed. This is the key difference that enables higher efficiency of Lev compared to DiSF.

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: They accurately reflect the paper's contributions and scope.

   Guidelines:

   - The answer [N/A] 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 [N/A] 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: [N/A]

   Justification: No specific limitation to highlight.

   Guidelines:

   - The answer [N/A] 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: [Yes]

Justification: Details in problem statement and method sections.

Guidelines:

- The answer [N/A] 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: Details in the experimental part.

   Guidelines:

   - The answer [N/A] 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: [Yes]

Justification: Would release code in supplementary materials.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.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://neurips.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, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Important aspects are reported.

   Guidelines:

   - The answer [N/A] 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: [No]

   Justification: We train models with data of enough token to reduce the variance of the results, which has plausibly provides enough statistical confidence. This is an acceptable practice in existing literature, since multiple pretraining trials would be significantly expensive and time-consuming.

   Guidelines:

   - The answer [N/A] 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 the experimental part.

   Guidelines:

   - The answer [N/A] 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: Yes, it does.

   Guidelines:

   - The answer [N/A] 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: [No]

    Justification: No specific societal impact.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] 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., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: No specific risks.

    Guidelines:

    - The answer [N/A] 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: Citations are provided.

    Guidelines:

    - The answer [N/A] 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: [N/A]

    Justification: No specific asset.

    Guidelines:

    - The answer [N/A] 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: [N/A]

    Justification: This work does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] 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: [N/A]

    Justification: This work does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] 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.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LLM used only for writing.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.