BENCHMAXING SELECTING DATA THAT TARGETS MANY BENCHMARKS WITH SCALABLE GRADIENT-BASED METHODS

Anonymous authors
Paper under double-blind review

ABSTRACT

Data quality is multi-dimensional: whether a sample is "good" depends on the model's downstream use, the checkpoint's weaknesses, and the composition of the surrounding dataset. Mainstay methods rely on human-like heuristics that, while useful, cannot target specialized use cases in a principled way. We advocate for benchmark-targeted data filtering, which scores samples based on their predicted effect on downstream performance. Prior approaches have been computationally expensive and limited to older models and small training scales. Our method, Benchmax, utilizes coverage selection criteria that improve the diversity and effectiveness of selected data, and combines gradient sketching with fast KNN search for efficient selection at scale. We distill the selection criteria into a multi-head classifier that simultaneously predicts a sample's impact across many benchmarks. This low-cost classifier scales to modern training regimes, scoring 1 trillion tokens for approximately $500 in cloud costs, making principled data selection practical.

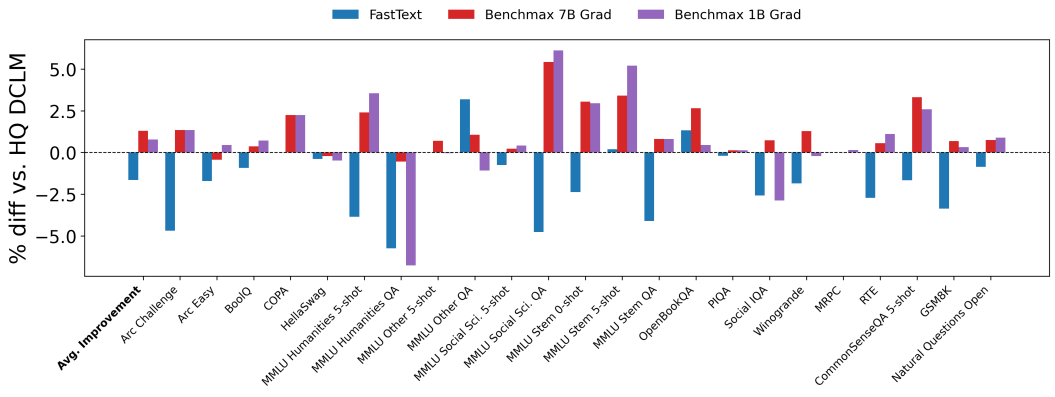


Figure 1: Benchmax's multi-task data filtering results for OLMo-2-7B across ∼20 benchmarks spanning knowledge, reasoning, and commonsense tasks (y-axis: % difference vs. HQ DCLM). Benchmax using 7B gradients consistently outperforms the HQ DCLM baseline and fastText filtering; Using gradient from OLMo-2-1B also generally outperforms the baseline, despite a mismatch of source model scale. fastText filtering sometimes performs better than the baseline, but is harmful to downstream tasks on average. We enumerate multiple splits and formats of MMLU for granularity.

1 INTRODUCTION

The quality of pretraining data has a critical impact on large language model (LLM) performance. Web-based training data must be carefully curated, as low-quality and unfiltered web data can actively harm model performance on downstream tasks (Li et al., 2025; Penedo et al., 2024; Lee et al., 2022). Mainstream filtering methods for LLM training typically score training samples along a one-dimensional axis, often based on heuristics or human intuition (e.g., "does this sample sound

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

educational?"). In practice, however, data filtering is inherently multi-dimensional. The "goodness" of a sample is a complex question, depending on the downstream use of the model, the weaknesses of the checkpoint being trained, and the presence of other related samples in the dataset. Compounding this, specialized models often need to be trained for high performance on particular subsets of downstream tasks. The central question we address is therefore: *How can we efficiently identify which pretraining examples are most beneficial for a downstream target task or set of tasks?*

In principle, the answer is to train on each sample one-at-a-time and then benchmark, keeping only examples that decreases the evaluation loss. While this approach is intractable, gradient-based methods provide a first-order approximation to this filter (Pruthi et al., 2020; Killamsetty et al., 2021b; Yu et al., 2024). In this approach, one chooses a model checkpoint to filter against, and a collection of labeled reference data for a downstream benchmark. One then pre-computes the gradient of an evaluation loss using this checkpoint and reference samples. For each candidate training sample, we then compute the training loss gradient, which represents the impact that sample will have on model weights. We then choose all samples for which the dot product between the sample gradient and reference gradient is large, as we expect training on these samples to effectively decrease the benchmark loss (Killamsetty et al., 2021b; Xia et al., 2024; Wang et al., 2024).

The gradient approach suffers from problems that have prevented its use for modern models. First, while tractable at small scales, computing gradients on each sample is exessively expensive at the pre-training scale. Second, the benchmark gradient is averaged over all reference examples. This collapsed/averaged approach discards the diversity in target task, and samples data that benefit only the most typical benchmark problems that contribute most to the loss. This results in a fast decrease in benchmark loss that quickly saturates as the tail of the reference distribution goes untouched.

In this work, we propose **Benchmax**, a method that bridges the gap between task-aware data selection and computational tractability. The centerpiece of Benchmax is a *coverage*-based insight: for a training set to remain effective over time, it must contain samples that collectively reduce the downstream loss on *all* individual reference examples. To enable pretraining-scale implementation, we develop an efficient process that uses CountSketch (Charikar et al., 2004) to collect the top-K candidate training examples for each individual reference example. These "golden examples" serve as a basis to distill a classifier that efficiently scores the entire training corpus for task-targeted filtering. Our distilled classifier has multiple heads, enabling it to score against many benchmarks simultanously and in a single pass.

We create a 25B-token filtered dataset out of the high-quality 250B DCLM split of OLMO mid-training corpus (OLMo et al., 2024) using Benchmax. When used to mid-train OLMo-2-7B, the performance of $\sim$30 target tasks improves simultaneously, out-performing DCLM and another fastText-classifier baseline.

2 Related Work

Coreset-based data selection aims to construct a small subset of training data that approximates a *global* objective, such as the average training losses or gradient vectors (Mirzasoleiman et al., 2020; Killamsetty et al., 2021c;a). These approaches typically do not require a reference set for data selection and not task-aware. Our method can be viewed as a task-conditioned coreset that preserves *per-example* structure: instead of approximating a global gradient, we explicitly maintain alignment with each reference example and select data via a coverage objective in gradient space. Our coverage-based construction is closely related to facility-location objectives (Wei et al., 2015) commonly used in submodular data selection, which aim to select a subset that maximizes the similarity between each target example and its nearest representative in the selected set. However, a key difference is that facility-location methods solve a globally budgeted optimization problem, where different target examples compete for representation, often resulting in a single selected example covering many similar targets. In contrast, our method performs per-example top-K retrieval and takes the union across all targets, ensuring that each reference example is explicitly represented.

Gradient-based data selection. A growing line of work selects training data based on per-sample gradients or gradient-derived quantities. Early work scores individual examples by their gradient norm (GraNd) or error norm (EL2N) (Paul et al., 2023). GRAD-MATCH (Killamsetty et al., 2021b) selects weighted subsets whose gradients approximate the full training or validation gradient via

orthogonal matching pursuit. However, these approaches are task-agnostic and do not adapt to any downstream reference set. LESS (Xia et al., 2024) uses gradient similarity to select instruction-tuning data, and online methods such as GREATS (Wang et al., 2024) perform batch selection during training by greedily optimizing a validation-aligned utility based on gradient inner products. ClusterUCB (Wang et al., 2025) scale target-aware selection by clustering candidate gradient vectors and framing retrieval as a multi-armed bandit problem. These gradient-based data selection methods share the principle that gradient alignment with a reference signal provides a useful measure of data utility. However, they focus on optimizing an aggregated gradient-based metric.

Data selection based on other metrics. BETR (Mizrahi et al., 2025) computes the "usefulness" of a training example by its maximum similarity to any reference example in encoder-embedding space. Similar to our method, BETR trains a classifier based on a subset of training examples with ground-truth similarity, and scores the remaining corpus predictively using the trained classifier. MATES (Yu et al., 2024) computes actual reference loss after updating the model on each candidate example in a subset of the training corpus, and trains a classifier accordingly to score the whole corpus. DsDm (Engstrom et al., 2024) attacks data selection from the angle of subset selection that optimizes reference loss reduction, and leverages linear modeling to approximate the influence of a subset of training examples on reference loss by a linear combination of contributions from individual examples. The approach is computationally expensive as it requires multiple sampled subsets for better fitting of linear model parameters. DSIR (Xie et al., 2023) matches a target distribution in a hashed n-gram space without consulting gradients or losses, carrying no model-aware signal.

3 Challenges

In this section, we discuss the challenges that informed our design choices. We consider the following data filtering problem. We have access to:

- A **base model** f_θ with parameters $\theta \in \mathbb{R}^d$: a language model checkpoint at some stage of pretraining (e.g., after the first stage of pretraining).
- A **source data pool** $\mathcal{D}_{\text{source}} = \{x_1, x_2, \ldots, x_N\}$ consisting of N candidate pretraining documents drawn from a large web-crawled corpus (e.g., DCLM, FineWeb).
- A **reference set** $\mathcal{D}_{\text{ref}} = \{z_1, z_2, \ldots, z_M\}$ consisting of M examples representative of the downstream task(s) on which we wish to improve performance.

Each of these components introduces its own computational or methodological challenge, which we address in turn.

3.1 Challenge 1: Computing Gradients for All Documents in the Corpus

Gradient-based filtering methods require computing a gradient inner product against every document in the corpus, which is prohibitively expensive at scale. To understand this challenge, consider a first-order Taylor expansion of the loss after a gradient step on x:

$$\mathcal{L}(z;\theta') - \mathcal{L}(z;\theta) \approx -\eta \left\langle \nabla_\theta \mathcal{L}(x;\theta),\ \nabla_\theta \mathcal{L}(z;\theta) \right\rangle \tag{1}$$

where $\theta' = \theta - \eta \nabla \mathcal{L}(x;\theta)$.

The inner product $\langle \nabla_\theta \mathcal{L}(x;\theta), \nabla_\theta \mathcal{L}(z;\theta) \rangle$ directly measures, to first order, how much training on x would reduce the loss on z, closely resembling influence functions (Koh & Liang, 2020) and TracIn (Pruthi et al., 2020). The optimal training example for z is therefore the x with the highest inner product score. However, computing this score requires evaluating gradients for every document in the pretraining corpus, which is computationally prohibitive.

We address this by approximating the inner product score via distillation into a transformer-based classifier, bypassing the need for per-document gradient computation at inference time.

3.2 Challenge 2: Storing Gradients and Computing Inner Products

Even setting aside the full corpus, distilling the inner product function still requires gradients for a large number of examples, on the order of 100k, along with their inner products against the reference

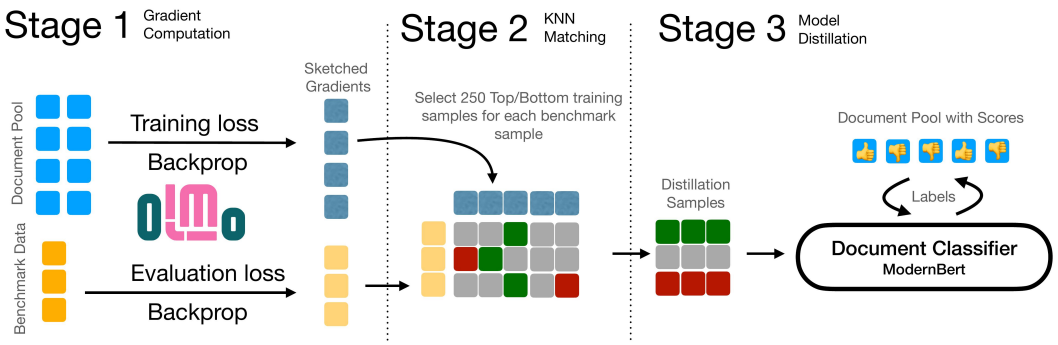


Figure 2: A very high-level overview of the Benchmax pipeline. **Stage 1 (Gradient Computation:** sketched gradient scoring. **Stage 2 (KNN Matching):** per-reference-example nearest-neighbor label construction. **Stage 3 (Distillation):** lightweight classifier distillation and filtering.

z. Storing and computing these scores is both memory- and compute-intensive. For OLMo-2-1B, a gradient d is $\approx 1.2 \times 10^9$, making it infeasible to store these vectors or compute inner products for millions of candidate examples.

We address this using the Johnson–Lindenstrauss Lemma (Johnson et al., 1984) and CountSketch (Charikar et al., 2004). CountSketch is a randomized linear sketching algorithm that compresses a d-dimensional gradient to $\mathbb{R}^k$ ($k \ll d$) via random coordinate hashing and sign assignments, preserving inner products in expectation. Applied layer-by-layer during the backward pass, it avoids ever materializing the full gradient. As an alternative, one could randomly sample a subset of gradient coordinates; however, CountSketch consistently achieves higher correlation with the true full-dimensional inner product scores, particularly at smaller sketch dimensions, and we therefore adopt it throughout.

3.3 Challenge 3: Preserving Diversity in the Target Benchmark

The two challenges above concern computational feasibility; this third challenge is methodological. The most common approach when optimizing for a target benchmark is to aggregate over the reference set—what we call the **collapsed gradient approach**. Formally, the collapsed gradient score of a candidate x is:

$$s_{\text{collapsed}}(x) = \langle \nabla_\theta \mathcal{L}(x;\theta),\ \bar{g}_{\text{ref}} \rangle \quad \text{where } \bar{g}_{\text{ref}} = \frac{1}{M} \sum_{j=1}^{M} \nabla_\theta \mathcal{L}(z_j;\theta). \tag{2}$$

This approach averages the reference gradients into a single vector and scores each candidate against it. While fast, collapsing the reference set in this way discards the diversity of the benchmark. Despite significant effort to scale the collapsed gradient approach to the midtraining setting, its performance never exceeded the HQ DCLM baseline (see Table 3 in Section C). This is likely because the collapsed gradient is dominated by a cluster of similar evaluation examples that contribute large gradients, and data selection targets only these samples.

Rather than collapsing the reference gradients, we adopt a coverage-based approach. Equipped with the dimensionality reduction technique described above, we use k-nearest neighbor (k-NN) search to select the top-scoring training examples for each individual validation example, preserving the diversity of the reference set throughout the selection process, and ensuring that the resulting training set covers the entire benchmark. This coverage criteria is an important facet of Benchmax — we see in the ablations that it results in dramatic improvements over previous gradient-based schemes.

4 Detailed Methodology

We now describe Benchmax. The method consists of three stages: (1) gradient scoring using CountSketch, (2) KNN-based label construction, and (3) classifier distillation for scalable filtering.

4.1 Stage 1: Sketched Gradient Scoring

Reference model. In our experiments, we use OLMo-2-1B or 7B at the end of its first pretraining stage (trained on ∼4T tokens). The choice of checkpoint matters: a model that has seen enough data to develop meaningful representations, but has not yet converged, provides the most informative gradients. We verify that the gradient signal is stable across nearby checkpoints (Figures 10 and 11).

Computing sketched gradients. For each example (whether from the reference set or the source pool), we perform a forward and backward pass through the reference model to obtain the gradient of the cross-entropy loss with respect to the model parameters. Rather than storing the full gradient $\nabla_\theta \mathcal{L}(x;\theta) \in \mathbb{R}^d$, we apply a CountSketch $S : \mathbb{R}^d \to \mathbb{R}^k$ layer-by-layer during the backward pass:

$$\tilde{g}(x) = \bigoplus_{\ell=1}^{L} S_\ell \left(\nabla_{\theta_\ell} \mathcal{L}(x;\theta) \right) \tag{3}$$

where $\bigoplus$ denotes concatenation, S_ℓ is the CountSketch applied to layer ℓ, and L is the number of layers (excluding the LM head). In practice, we sketch each layer's gradient to a fixed dimension (e.g., $k_\ell = 1 \times 1000 = 1000$ per layer for a sketch with 1 rows and 1000 columns), yielding a total sketch dimension of $k = k_\ell \times L$.

Masking strategy. For the reference set, we compute the gradient only on the tokens corresponding to the *answer* or *response* portion of each example, masking the prompt. This ensures the gradient reflects the model's behavior on the task-relevant output rather than the input formatting. For source pool examples (raw pretraining documents), we compute the gradient on all tokens using the standard next-token prediction loss, just as we would during the training loop.

4.2 Stage 2: Coverage-Based Label Construction

Direct application of the gradient score to rank all documents in a multi-trillion-token corpus is still expensive, as it requires computing sketched gradients for every document. To scale beyond the initial scored pool, we construct a labeled dataset for training a fast classifier. The key design principle—and the "coverage" in Benchmax—is that we construct labels *per reference example* rather than from a collapsed aggregate score.

Given sketched gradients $\tilde{g}(x)$ for a candidate x and $\{\tilde{g}(z_j)\}_{j=1}^{M}$ for the reference set, we compute the *pairwise* approximate cosing similarity between x and each reference example z_j:

$$\tilde{s}_j(x) = \frac{\langle \tilde{g}(x), \tilde{g}(z_j) \rangle}{\|\tilde{g}(x)\| \|\tilde{g}(z_j)\|} \approx s_j(x). \tag{4}$$

Scoring a candidate pool. We first score a manageable pool of ∼1M candidate training examples from the source corpus, computing the pairwise sketched gradient normalized inner product (or cosine similarity) between each candidate and each reference example as described above.

Per-reference-example nearest-neighbor retrieval. For each reference example z_j individually, we retrieve the top-K nearest neighbors from the candidate pool based on $\tilde{s}_j(x)$ (cosine similarity with that specific z_j). These are labeled as **positive** examples. These examples are the training examples most helpful for z_j. To identify harmful examples, we negate the sketched reference gradient, $-\tilde{g}(z_j)$, and repeat the process to obtain our **negative** examples.

We take the union over the positive and negative sets from all reference examples. For each training example, if that training example has a positive score for one or more reference examples, we keep one copy of it.

$$\mathcal{P} = \bigcup_{j=1}^{M} \text{top-}K\big(\{x \in \mathcal{D}_{\text{pool}} : \tilde{s}_j(x)\}\big), \qquad \mathcal{N} = \bigcup_{j=1}^{M} \text{top-}K\big(\{x \in \mathcal{D}_{\text{pool}} : -\tilde{s}_j(x)\}\big). \tag{5}$$

A **random** sample $\mathcal{R}$ is drawn from the remaining candidates to provide a neutral baseline. The final labeled dataset is:

$$\mathcal{D}^{\text{cls}}_{\text{train}} = \mathcal{P} \cup \mathcal{N} \cup \mathcal{R}. \tag{6}$$

4.3 Stage 3: Classifier Distillation

We train a **ModernBERT** (Warner et al., 2025) classifier to predict the gradient-based labels. The classifier is trained to predict whether a document is positive (beneficial for the target task), negative (harmful), or random (neutral). At inference time, we use the predicted probability of the positive class as a filtering score.

Threshold selection. Rather than fixing a probability cutoff directly, we derive the threshold τ from a target token count T_{target}. We apply the trained classifier to all documents in the source corpus and rank them by predicted positive-class probability. We then set τ such that the documents above τ collectively contain exactly T_{target} tokens, ensuring the filtered corpus matches the desired training budget regardless of the score distribution.

Multi-Task Filtering. When targeting multiple downstream tasks, we use an efficient shared-encoder approach. We start by training ModernBERT end-to-end on samples from the MMLU benchmarks using a single unified head. Then for each individual benchmark, we follow Stages 1–3 to produce training data, and train only a lightweight linear classification head on the frozen encoder, making the approach easily extensible. This approach also allows one to cache the pre-computed embedding vectors for each candidate sample, and run only the last linear layer when sampling is needed for a new benchmark.

At filtering time, each task's threshold τ_t is derived using the token-count method described earlier with a shared target T_{target}. A document is retained if it meets the threshold for any task, ensuring broad coverage across all targets.

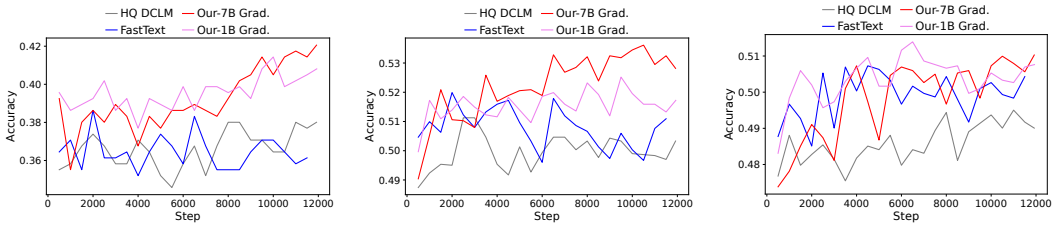
5 Experimental Set-up

Reference model. We use OLMo-2 1B and 7B after stage-1 pretraining (i.e., before mid-training) as the primary models for all experimental settings. Unless otherwise stated, we use the checkpoint at the end of the first pretraining stage. We additionally follow the experimental setup from OLMo et al. (2024) for mid-training over 25B tokens, with further details provided in Section F. This choice of 25B during mid-training is mainly explored due to the feasibility of exploring improvement from multiple datasets during this phase of training without expending too much compute. Prior work has found that mid-training performance is a good proxy for pre-training quality (Blakeney et al., 2024).

Source data. We use the DCLM split of the Dolmino mid-training dataset for all experiments. This split is of higher quality than the DCLM pretraining data but lower quality than the full Dolmino-mix, and it has already undergone decontamination as described in OLMo et al. (2024). We start from this DCLM baseline, rather than from the original mid-training mix, in order to test whether a less curated set of data can be refined to improve performance beyond what is achieved by the DCLM filter.

Methods. We compare the following methods in the results below: **HQ DCLM.** We randomly sample 25B tokens from the source corpus. **fastText.** We train a fastText classifier to distinguish high-quality from average-quality data, where the high-quality data is the target benchmark (e.g., MMLU) and the average-quality data is DCLM. This approach is similar to that of DCLM (Li et al., 2025), and gives us a competitive task-targeted baseline that does not make use of gradient information. **Benchmax.** We apply our full pipeline described in Section 4.

Reference sets and evaluations. For evaluation, we use the OLMo mid-training evaluation set, and we explore two settings. In the **first setting**, we apply Benchmax to MMLU STEM multiple choice in a zero-shot MC format that is not part of the original harness. This setting serves as a controlled diagnostic for whether the algorithm is behaving as intended. We note that we also explore collapsing all reference-example gradients into a single mean gradient; in both online and offline configurations, filtering with this collapsed gradient proved insufficient. In the **second setting**, we jointly optimize over more than 20 benchmarks: ARC Easy and Challenge (Clark et al., 2018), BoolQ (Clark et al., 2019), CommonSenseQA (Talmor et al., 2019), COPA (Su et al., 2026), HellaSwag (Zellers et al., 2019), MMLU (nine versions) (Hendrycks et al., 2021), OpenBookQA (Mihaylov et al., 2018), PiQA (Bisk et al., 2020), SciQ (Welbl et al., 2017), Winogrande (Sakaguchi et al., 2020), and Natural Questions (Kwiatkowski et al., 2019).



(a) MMLU-STEM, QA (b) MMLU-STEM, 5-shot MC (c) MMLU-STEM, zero-shot MC

Figure 3: Benchmax mid-training results on OLMo-2-7B with MMLU-STEM as the single reference set. Each subplot shows all four methods—HQ DCLM, fastText, Benchmax with 1B gradients, and Benchmax with 7B gradients—across three evaluation formats: QA, 5-shot MC, and zero-shot MC. Benchmax with 7B gradients outperforms fastText on 5-shot MC and QA; Benchmax with 1B gradients also outperforms fastText on QA. Both match fastText on zero-shot MC.

Scoring the Data We use both OLMo-2 1B and 7B in bfloat16 to score the data, producing two different scored sets for each reference set. We project each layer with CountSketch into a single row of 1,000 elements for both the 1B and 7B models, yielding a per-example projection of 112,000 elements — roughly a 10,000× reduction relative to the full gradient. We determine the projection dimension with ablation.

Distilling the Score To construct the classifier training data, we sample 1M documents from the source corpus and draw 100,000 examples across three classes: random documents, nearest neighbors (positive), and farthest neighbors (negative), using K=250 neighbors per reference example. We train a three-way ModernBERT (Warner et al., 2025) classifier on these classes, including the farthest-neighbor class to support future work on data *removal* even though we do not currently use it downstream. We train end-to-end on one selected dataset, freeze the backbone, then train a separate linear head per reference set—allowing us to precompute each document's hidden representation once and reuse it across reference sets.

Filtering the Documents We filter a subset of ∼250B tokens from the DCLM split of Dolmino-Mix-1124 down to ∼25B. The threshold for both single-task and multi-task filtering is derived following the respective procedures described in Section 4.

Reproducibility and Computational Infrastructure. We conduct all experiments on commodity GPUs; data filtering on A5000s and training on MI300Xs and H100s. Each stage of the data filtering is parallelizable and runs within a few hours on our small cluster. We provide the full breakdown of wall clock time and FLOPs usage in Section E. Based on current cloud pricing at the time of writing this report, we estimate the cost to run the entire filtering pipeline on 1T tokens at just under $500. We release all code in the zip and artifacts associated with our work.[1]

6 Results

We demonstrate the efficacy of Benchmax. First, we validate the method on a single task, MMLU-STEM, across different formats. We find that hill-climbing on MMLU-STEM zero-shot not only improves performance in that setting, from 49.0 to 51.1 over the baseline, but also generalizes to other settings such as open-ended QA (37.7 → 41.7) and 5-shot benchmarks (50.1 → 52.7). To further demonstrate the scalability, we expand the dataset selection to target multiple benchmarks, achieving an improvement of 3.00% over the baseline on the 7B model and a substantially larger 11.42% improvement on the 1B model. We then demonstrate the generalization capability of Benchmax by filtering against only a subset of the benchmarks and evaluating on the full evaluation suite, improving the targeted benchmarks by 3.58% and the full suite by 2.54% over the baseline. The positive results show that Benchmax is capable of selecting data that improves targeted benchmarks, and the filtered data remains useful for other held-out benchmarks and different formatting. Finally, we conduct an ablation study of different design choices for Benchmax in Section 6.1.

[1] https://huggingface.co/benchmaxxer

Targeted filter on MMLU-STEM. We filter a pretraining corpus using MMLU-stem as the single-task reference set, where each reference example is formatted as a zero-shot multiple choice question. Each reference sample is formatted as a zero-shot MC question with the letter as the gold label. We compare four selection strategies: (1) Benchmax using 1B gradients, (2) Benchmax using 7B gradients, (3) fastText, and (4) HQ DCLM. All filtered datasets are used to mid-train OLMo-2-7B, evaluated on three formats: MC zero-shot, MC 5-shot, and open-ended QA.

In Figure 3, Benchmax matches fastText on MC zero-shot for both gradient sources and on MC 5-shot when using 1B gradients, despite finding very different sets of filtered datasets (see Figure 4 in Section C). However, using 7B gradients outperforms fastText on MC 5-shot, and both gradient sources outperform fastText on the QA format. In short, our coverage approach provides a more robust, model-tailored solution compared to fastText. We also provide further analysis of selected samples in Section H.

Multi-task filtering results. We apply the multi-task filtering pipeline (Section 4.3), training a shared encoder end-to-end and attaching lightweight linear heads for each additional benchmark. We set the token-count threshold (Section 4.3) to yield a filtered training set of 25B tokens. See section F for how multi-benchmark fastText classifier is constructed.

Table 1 reports per-benchmark results for both OLMo-2-7B and OLMo-2-1B when we filter against all the listed benchmarks. For OLMo-2-7B, Benchmax with 7B gradients achieves consistent gains across 23 out of 27 benchmarks ($+3.00\%$ average relative difference vs. HQ DCLM), and Benchmax with 1B gradients also outperforms HQ DCLM ($+1.88\%$), while fastText is harmful on average (-0.70%). For OLMo-2-1B, Benchmax achieves an average improvement of $+11.42\%$ and fastText achieves an average improvement of $+11.65\%$. Furthermore, the same fastText dataset that helps the 1B model is harmful for the 7B model, underscoring that lexical style-matching is not a robust filtering strategy across model scales.

Generalization to held-out benchmarks. A natural worry about benchmark-targeted filtering is that it overfits the target and does not generalize to unseen benchmarks. We test the generalization capability of our method by performing an experiment on the OLMo-2-7B model where only the MMLU-related benchmark is used to filter the data. We evaluate the trained model on the full evaluation suite, and report the results in the last column of 7B model in Table 1. The results show that the trained model improves performance on the MMLU related benchmarks by $+3.58\%$ on average, improving 8 of the 9 MMLU rows, and also improves average performance over the whole evaluation suite by $+2.54\%$. This demonstrates that Benchmax is capable of generalization to unseen and potentially very different benchmarks.

6.1 Ablation

Ignoring the LM Head. The LM head, the final projection from hidden states to vocabulary logits, is large and dominates gradient magnitude, but contributes noise to the inner product signal. Scores computed with and without the LM head are highly correlated (Pearson r$\approx$0.93), and removing it *significantly improves* computational cost. We therefore exclude it from all gradient computations.

CountSketch Ablation. Figure 6 in Section C shows that performance improves as we increase the sketch dimensionality from $1 \rightarrow 75,000$. The improvement becomes marginal at 750, but we round up to 1,000 for a cleaner number. Note that the 1B model shows a very similar trend, with the correlation at 1,000 elements per layer also around 0.92.

Benchmax vs Collapsed Gradient. We also ablate the proposed coverage-based label construction by comparing against the naive filtering approach using collapsed gradients. Specifically, we obtain the mean reference gradient from MMLU stem with zero-shot MC format, and produce a filtered dataset accordingly with the collapsed gradient score Equation (2). The results (Table 3 in Section C) show that coverage-based label construction significantly improves filtering performance.

Distillation with Classifier: Linear head vs End-to-end training. In Section 4.3, we describe the design choice of training a shared encoder end-to-end and attaching lightweight linear heads for each additional benchmark. We ablate this design choice against end-to-end training of the same classifier architecture using CommonSenseQA as the reference set. Based on the precision recall curve of both approaches (Figure 7 in Section C), we find that the linear head approach can achieve similar performance to end-to-end training, while also admitting better scalability.

Table 1: **Benchmaxxing maximizes benchmark scores.** We evaluate models trained with all methods on a wide range of standard benchmarks. At the 7B scale, Benchmax improves by 3% over the competitive baseline of training on high-quality DCLM data, whereas filtering that data with a fastText classifier degrades performance. At the 1B scale, Benchmax and fastText seem to perform similarly. We also evaluate 7B models trained on data filtered by MMLU-related benchmarks alone using Benchmax, and find that our method generalizes to unseen benchmarks. Subscripts show standard error. Parentheses give each cell's difference against its group's HQ DCLM column. Bold indicates the best score for the model-size. Underline indicates second-best. MMLU-only filter is excluded from bold/underline comparison.

	7B Model					1B Model		
Benchmark	**HQ DCLM**	**fastText**	**Benchmax 7B Grad**	**Benchmax 1B Grad**	**Benchmax (MMLU)**	**HQ DCLM**	**Benchmax**	**fastText**
Avg. Rel. Diff. vs. HQ DCLM	—	-0.70%	**+3.00%**	+1.88%	+2.54%	—	+11.42%	**+11.65%**
Average ↑	61.3	59.8 (-1.5)	62.0 (+0.7)	**62.6** (+1.3)	63.1 (+1.8)	49.4	**53.5** (+4.1)	52.9 (+3.5)
Arc Challenge	$49.8_{2.9}$	$47.2_{2.9}$ (-2.6)	$\mathbf{50.5}_{2.9}$ (+0.7)	$50.2_{2.9}$ (+0.4)	$53.8_{2.9}$ (+4.0)	$45.2_{3.0}$	$44.8_{2.9}$ (-0.4)	$\mathbf{46.5}_{3.0}$ (+1.3)
Arc Easy	$81.6_{1.6}$	$80.4_{1.7}$ (-1.2)	$81.1_{1.6}$ (-0.5)	$\mathbf{82.3}_{1.6}$ (+0.7)	$81.8_{1.6}$ (+0.2)	$75.3_{1.8}$	$\mathbf{77.5}_{1.7}$ (+2.2)	$74.2_{1.8}$ (-1.1)
BoolQ	$80.9_{0.7}$	$79.8_{0.7}$ (-1.1)	$81.1_{0.7}$ (+0.2)	$\mathbf{81.6}_{0.7}$ (+0.7)	$81.5_{0.7}$ (+0.6)	$66.2_{0.8}$	$\mathbf{66.8}_{0.9}$ (+0.6)	$65.4_{0.8}$ (-0.8)
COPA	$89.0_{3.2}$	$88.0_{3.3}$ (-1.0)	$\mathbf{92.0}_{2.8}$ (+3.0)	$91.0_{3.0}$ (+2.0)	$89.0_{3.2}$ (-0.0)	$78.0_{4.3}$	$\mathbf{83.0}_{3.8}$ (+5.0)	$78.0_{4.2}$ (+0.0)
HellaSwag	$\mathbf{78.7}_{0.4}$	$78.5_{0.4}$ (-0.2)	$78.6_{0.4}$ (-0.1)	$78.5_{0.4}$ (-0.2)	$78.5_{0.4}$ (-0.2)	$67.8_{0.5}$	$68.3_{0.5}$ (+0.5)	$\mathbf{68.5}_{0.5}$ (+0.7)
MMLU Humanities 5-shot	$52.1_{2.2}$	$50.6_{2.1}$ (-1.5)	$\mathbf{55.0}_{2.2}$ (+2.9)	$52.5_{2.1}$ (+0.4)	$53.7_{2.1}$ (+1.6)	$33.6_{2.1}$	$\mathbf{40.5}_{2.2}$ (+6.9)	$39.8_{2.1}$ (+6.2)
MMLU Humanities QA	$36.5_{2.1}$	$35.1_{2.1}$ (-1.4)	$\mathbf{37.3}_{2.0}$ (+0.8)	$34.6_{2.0}$ (-1.9)	$36.9_{2.1}$ (+0.4)	$\mathbf{33.8}_{2.1}$	$32.2_{2.0}$ (-1.6)	$30.3_{2.0}$ (-3.5)
MMLU Other 5-shot	$70.4_{2.5}$	$69.3_{2.4}$ (-1.1)	$\mathbf{71.8}_{2.3}$ (+1.4)	$71.3_{2.4}$ (+0.9)	$71.0_{2.4}$ (+0.6)	$38.6_{2.6}$	$55.5_{2.7}$ (+16.9)	$\mathbf{58.3}_{2.7}$ (+19.7)
MMLU Other QA	$53.2_{2.7}$	$\mathbf{54.4}_{2.7}$ (+1.2)	$54.1_{2.8}$ (+0.9)	$52.4_{2.7}$ (-0.8)	$53.0_{2.7}$ (-0.2)	$49.0_{2.7}$	$\mathbf{51.0}_{2.7}$ (+2.0)	$48.5_{2.8}$ (-0.5)
MMLU Social Sci. 5-shot	$71.8_{2.3}$	$70.9_{2.5}$ (-0.9)	$\mathbf{73.6}_{2.3}$ (+1.8)	$70.9_{2.5}$ (-0.9)	$73.9_{2.4}$ (+2.1)	$35.6_{2.6}$	$\mathbf{52.2}_{2.8}$ (+16.6)	$51.3_{2.7}$ (+15.7)
MMLU Social Sci. QA	$44.2_{2.7}$	$40.9_{2.6}$ (-3.3)	$45.4_{2.7}$ (+1.2)	$\mathbf{46.3}_{2.7}$ (+2.1)	$46.6_{2.7}$ (+2.4)	$38.9_{2.6}$	$\mathbf{43.0}_{2.7}$ (+4.1)	$39.5_{2.6}$ (+0.6)
MMLU Stem zero-shot	$38.0_{2.8}$	$37.1_{2.7}$ (-0.9)	$\mathbf{38.6}_{2.8}$ (+0.6)	$38.3_{2.7}$ (+0.3)	$41.1_{2.8}$ (+3.1)	$33.3_{2.7}$	$34.0_{2.7}$ (+0.7)	$\mathbf{35.5}_{2.8}$ (+2.2)
MMLU Stem 5-shot	$44.5_{2.7}$	$45.8_{2.8}$ (+1.3)	$46.4_{2.8}$ (+1.9)	$\mathbf{46.7}_{2.7}$ (+2.2)	$45.8_{2.8}$ (+1.3)	$27.1_{2.5}$	$31.5_{2.7}$ (+4.4)	$\mathbf{34.3}_{2.7}$ (+7.2)
MMLU Stem QA	$38.0_{2.8}$	$37.1_{2.7}$ (-0.9)	$\mathbf{38.6}_{2.8}$ (+0.6)	$38.3_{2.7}$ (+0.3)	$41.1_{2.8}$ (+3.1)	$33.3_{2.7}$	$33.6_{2.7}$ (+0.3)	$\mathbf{35.5}_{2.8}$ (+2.2)
OpenBookQA	$44.8_{2.2}$	$45.6_{2.2}$ (+0.8)	$\mathbf{46.8}_{2.2}$ (+2.0)	$45.6_{2.2}$ (+0.8)	$45.8_{2.2}$ (+1.0)	$\mathbf{40.0}_{2.2}$	$39.6_{2.2}$ (-0.4)	$39.6_{2.2}$ (-0.4)
PIQA	$81.3_{0.9}$	$81.4_{0.9}$ (+0.1)	$81.5_{0.9}$ (+0.2)	$\mathbf{81.6}_{0.9}$ (+0.3)	$81.2_{0.9}$ (-0.1)	$76.7_{1.0}$	$77.1_{1.0}$ (+0.4)	$\mathbf{77.7}_{1.0}$ (+1.0)
Social IQA	$49.5_{1.1}$	$48.4_{1.1}$ (-1.1)	$\mathbf{49.8}_{1.1}$ (+0.3)	$48.1_{1.1}$ (-1.4)	$49.6_{1.1}$ (+0.1)	$48.0_{1.1}$	$48.1_{1.1}$ (+0.1)	$\mathbf{48.3}_{1.1}$ (+0.3)
Winogrande	$73.3_{1.2}$	$71.6_{1.3}$ (-1.7)	$\mathbf{74.2}_{1.2}$ (+0.9)	$72.8_{1.2}$ (-0.5)	$73.9_{1.2}$ (+0.6)	$64.0_{1.4}$	$64.5_{1.3}$ (+0.5)	$\mathbf{64.8}_{1.3}$ (+0.8)
Commitment Bank	$57.1_{6.7}$	$41.1_{6.5}$ (-16.0)	$51.8_{6.8}$ (-5.3)	$\mathbf{60.7}_{6.5}$ (+3.6)	$66.1_{6.4}$ (+9.0)	$\mathbf{39.3}_{6.4}$	$32.1_{6.1}$ (-7.2)	$32.1_{6.2}$ (-7.2)
CommonSenseQA 5-shot	$70.0_{0.5}$	$69.5_{0.5}$ (-0.5)	$\mathbf{70.9}_{0.5}$ (+0.9)	$70.7_{0.5}$ (+0.7)	$70.0_{0.5}$ (+0.0)	$27.3_{0.4}$	$52.9_{0.5}$ (+25.6)	$\mathbf{55.9}_{0.5}$ (+28.6)
MRPC	$81.2_{1.7}$	$81.2_{1.7}$ (+0.0)	$81.2_{1.7}$ (+0.0)	$\mathbf{81.3}_{1.7}$ (+0.1)	$82.1_{1.7}$ (+0.9)	$71.8_{2.1}$	$72.8_{2.1}$ (+1.0)	$\mathbf{80.8}_{1.7}$ (+9.0)
RTE	$66.1_{2.8}$	$64.3_{2.8}$ (-1.8)	$66.4_{2.8}$ (+0.3)	$\mathbf{66.8}_{2.8}$ (+0.7)	$66.4_{2.8}$ (+0.3)	$53.8_{2.9}$	$\mathbf{58.8}_{3.0}$ (+5.0)	$55.6_{3.0}$ (+1.8)
SST-2	$57.6_{1.7}$	$57.5_{1.7}$ (-0.1)	$58.1_{1.7}$ (+0.5)	$\mathbf{76.9}_{1.5}$ (+19.3)	$68.6_{1.6}$ (+11.0)	$60.7_{1.6}$	$\mathbf{70.0}_{1.6}$ (+9.3)	$56.1_{1.6}$ (-4.6)
Average ↓	68.3	66.6 (-1.7)	**64.3** (-4.0)	67.5 (-0.8)	68.6 (+0.3)	178	137 (-41.0)	**119** (-59.0)
CopyColors 10-way BPB	$23.0_{5.3}$	$11.9_{1.4}$ (-11.1)	$\mathbf{11.9}_{2.6}$ (-11.1)	$23.7_{8.3}$ (+0.7)	$26.0_{6.3}$ (+3.0)	$334_{11.2}$	$209_{11.5}$ (-125.0)	$\mathbf{143}_{8.2}$ (-191.0)
CommonSenseQA 5-shot BPB	$121_{1.5}$	$123_{1.5}$ (+2.0)	$\mathbf{117}_{1.4}$ (-4.0)	$118_{1.4}$ (-3.0)	$119_{1.5}$ (-2.0)	$224_{0.4}$	$187_{1.5}$ (-37.0)	$\mathbf{177}_{1.7}$ (-47.0)
GSM8K Gold 5-shot BPB	$42.7_{0.3}$	$44.2_{0.4}$ (+1.5)	$\mathbf{42.4}_{0.3}$ (-0.3)	$42.6_{0.3}$ (-0.1)	$42.9_{0.3}$ (+0.2)	$52.3_{0.4}$	$\mathbf{51.3}_{0.4}$ (-1.0)	$53.3_{0.4}$ (+1.0)
Natural Questions Open PPL	$86.2_{1.2}$	$86.9_{1.2}$ (+0.7)	$85.5_{1.2}$ (-0.7)	$\mathbf{85.4}_{1.2}$ (-0.8)	$86.2_{1.2}$ (+0.0)	$\mathbf{101}_{1.2}$	$\mathbf{101}_{1.3}$ (+0.0)	$104_{1.3}$ (+3.0)

Benchmax and fastText select different data. We examine correlations between the scores of a filtered subset of fastText models and Benchmax (see Figure 4 in Section C), and find that they are only weakly correlated, with at most 0.3 correlation for both model scales. The observation indicates that the selected examples are largely different between the two methods, and hints at the possibility of combining them for better performance. Our preliminary results on the 1B model show that combining both indeed yields a relative improvement of 13.36% over the HQ DCLM, and almost another 2% over Benchmax alone. We leave further investigation for future work.

Benchmark Contamination Analysis. Selecting documents by their alignment with benchmark examples raises the question of whether Benchmax simply pulls benchmark text into the training set. We perform overlap analysis between the benchmark examples and each corpus, and find contamination in 0.37% of benchmark examples in the selected 25B-token set, against 1.34% in the raw pool it was selected from. Our method in fact *decreases* the contamination, and the reported gains cannot be attributed to leakage. Further detail can be found in Section D.

7 Conclusion

Benchmax offers several advancements over prior gradient-based data filters, and enables benchmark-targeted filtering at scale. Several directions follow. First, ensembling multiple checkpoints trained on different data distributions could generalize coverage signals to unseen tasks. Second, combining Benchmax with complementary methods like fastText—which provide orthogonal signals—could substantially improve public datasets. Third, Benchmax's checkpoint-specificity makes it a natural fit for stage-wise filtering during pretraining: train on filtered data, update the classifiers, refilter the next batch, repeat.

Reproducibility Statement

We state the relevant training details in Section F, covering the mid-training configuration at both model scales, and describe each stage of the filtering pipeline in Section 4, with the FLOP and wall-clock breakdown in Section E. We include the code in the supplementary material and provide a repository containing all trained models and the filtered data.[2] We note that the code submitted as supplementary material cannot be executed as-is, because the OLMo code base includes evaluation data in its repository and that data exceeds the size limit for supplementary material.

AI Use Statement

We used generative AI tools in three ways in this work. First, as part of the supplementary method evaluation: in Section H we prompt an LLM to label each top-scored document on two axes, whether it is multiple-choice formatted and whether it is STEM-related. We reviewed every one of these labels by hand, and the analysis in that appendix reports the reviewed labels. Second, we used LLM assistance in preparing tables and figures from our logged evaluation results. The program used to generate those tables and figures is carefully checked for correctness. Third, we use LLM assistance to draft the outline of the paper, and manually write the content of each section. There is no AI assistance involved in the following subtasks that require disclosure:

- Generating synthetic data sets
- Helping develop theoretical models or conceptual frameworks
- Formulating mathematical claims
- Providing critical ingredients for proving mathematical claims
- Assisting in the writing of proofs
- Proposing or refining hypotheses
- Designing or providing feedback on research methodology or experiments
- Implementing methods
- Assisting with translation
- Cleaning and reformatting datasets

We have reviewed all AI-assisted work and take responsibility for the final content of this work, including all text, claims, and artifacts.

References

Yonatan Bisk, Rowan Zellers, Ronan Le bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about physical commonsense in natural language. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7432–7439, Apr. 2020. doi: 10.1609/aaai.v34i05.6239. URL https://ojs.aaai.org/index.php/AAAI/article/view/6239.

Cody Blakeney, Mansheej Paul, Brett W. Larsen, Sean Owen, and Jonathan Frankle. Does your data spark joy? performance gains from domain upsampling at the end of training. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=vwIIAot0ff.

Moses Charikar, Kevin Chen, and Martin Farach-Colton. Finding frequent items in data streams. *Theoretical Computer Science*, 312(1):3–15, 2004. ISSN 0304-3975. doi: https://doi.org/10.1016/S0304-3975(03)00400-6. URL https://www.sciencedirect.com/science/article/pii/S0304397503004006. Automata, Languages and Programming.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *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)*, pp. 2924–2936, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1300. URL https://aclanthology.org/N19-1300.

[2] https://huggingface.co/benchmaxxer

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. *CoRR*, arXiv:1803.05457, 2018.

Logan Engstrom, Axel Feldmann, and Aleksander Madry. Dsdm: Model-aware dataset selection with datamodels, 2024. URL https://arxiv.org/abs/2401.12926.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

William B Johnson, Joram Lindenstrauss, et al. Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics*, 26(189-206):1, 1984.

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

Krishnateja Killamsetty, Durga Sivasubramanian, Ganesh Ramakrishnan, Abir De, and Rishabh Iyer. Grad-match: Gradient matching based data subset selection for efficient deep model training, 2021b. URL https://arxiv.org/abs/2103.00123.

Krishnateja Killamsetty, Durga Sivasubramanian, Ganesh Ramakrishnan, and Rishabh Iyer. Glister: Generalization based data subset selection for efficient and robust learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 8110–8118, 2021c.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions, 2020. URL https://arxiv.org/abs/1703.04730.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019. doi: 10.1162/tacl_a_00276. URL https://aclanthology.org/Q19-1026.

Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. Deduplicating training data makes language models better. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8424–8445, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.577. URL https://aclanthology.org/2022.acl-long.577/.

Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Gadre, Hritik Bansal, Etash Guha, Sedrick Keh, Kushal Arora, Saurabh Garg, Rui Xin, Niklas Muennighoff, Reinhard Heckel, Jean Mercat, Mayee Chen, Suchin Gururangan, Mitchell Wortsman, Alon Albalak, Yonatan Bitton, Marianna Nezhurina, Amro Abbas, Cheng-Yu Hsieh, Dhruba Ghosh, Josh Gardner, Maciej Kilian, Hanlin Zhang, Rulin Shao, Sarah Pratt, Sunny Sanyal, Gabriel Ilharco, Giannis Daras, Kalyani Marathe, Aaron Gokaslan, Jieyu Zhang, Khyathi Chandu, Thao Nguyen, Igor Vasiljevic, Sham Kakade, Shuran Song, Sujay Sanghavi, Fartash Faghri, Sewoong Oh, Luke Zettlemoyer, Kyle Lo, Alaaeldin El-Nouby, Hadi Pouransari, Alexander Toshev, Stephanie Wang, Dirk Groeneveld, Luca Soldaini, Pang Wei Koh, Jenia Jitsev, Thomas Kollar, Alexandros G. Dimakis, Yair Carmon, Achal Dave, Ludwig Schmidt, and Vaishaal Shankar. Datacomp-lm: In search of the next generation of training sets for language models, 2025. URL https://arxiv.org/abs/2406.11794.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2381–2391, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1260. URL https://aclanthology.org/D18-1260.

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

David Mizrahi, Anders Boesen Lindbo Larsen, Jesse Allardice, Suzie Petryk, Yuri Gorokhov, Jeffrey Li, Alex Fang, Josh Gardner, Tom Gunter, and Afshin Dehghan. Language models improve when pretraining data matches target tasks, 2025. URL https://arxiv.org/abs/2507.12466.

Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, et al. 2 olmo 2 furious. *arXiv preprint arXiv:2501.00656*, 2024.

Mansheej Paul, Surya Ganguli, and Gintare Karolina Dziugaite. Deep learning on a data diet: Finding important examples early in training, 2023. URL https://arxiv.org/abs/2107.07075.

Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin 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.

Garima Pruthi, Frederick Liu, Mukund Sundararajan, and Satyen Kale. Estimating training data influence by tracing gradient descent, 2020. URL https://arxiv.org/abs/2002.08484.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial winograd schema challenge at scale. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8732–8740, Apr. 2020. doi: 10.1609/aaai.v34i05.6399. URL https://ojs.aaai.org/index.php/AAAI/article/view/6399.

Luca Soldaini, Rodney Kinney, Akshita Bhagia, Dustin Schwenk, David Atkinson, Russell Authur, Ben Bogin, Khyathi Chandu, Jennifer Dumas, Yanai Elazar, et al. 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)*, pp. 15725–15788, 2024.

Hang Su, Zequn Liu, Chen Hu, Xuesong Lu, Yingce Xia, and Zhen Liu. Copa: Benchmarking personalized question answering with data-informed cognitive factors. 2026. URL https://arxiv.org/abs/2604.14773.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *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)*, pp. 4149–4158, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1421. URL https://aclanthology.org/N19-1421.

Jiachen T. Wang, Tong Wu, Dawn Song, Prateek Mittal, and Ruoxi Jia. Greats: Online selection of high-quality data for llm training in every iteration. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 131197–131223. Curran Associates, Inc., 2024. doi: 10.52202/079017-4169. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/ed165f2ff227cf36c7e3ef88957dadd9-Paper-Conference.pdf.

Zige Wang, Qi Zhu, Fei Mi, Minghui Xu, Ruochun Jin, and Wenjing Yang. Clusterucb: Efficient gradient-based data selection for targeted fine-tuning of llms, 2025. URL https://arxiv.org/abs/2506.10288.

Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, Oskar Hallström, Said Taghadouini, Alexis Gallagher, Raja Biswas, Faisal Ladhak, Tom Aarsen, et al. Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2526–2547, 2025.

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

Johannes Welbl, Nelson F. Liu, and Matt Gardner. Crowdsourcing multiple choice science questions. In Leon Derczynski, Wei Xu, Alan Ritter, and Tim Baldwin (eds.), *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pp. 94–106, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-4413. URL https://aclanthology.org/W17-4413/.

Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. Less: Selecting influential data for targeted instruction tuning, 2024. URL https://arxiv.org/abs/2402.04333.

Sang Michael Xie, Shibani Santurkar, Tengyu Ma, and Percy Liang. Data selection for language models via importance resampling, 2023. URL https://arxiv.org/abs/2302.03169.

Zichun Yu, Spandan Das, and Chenyan Xiong. Mates: Model-aware data selection for efficient pretraining with data influence models, 2024. URL https://arxiv.org/abs/2406.06046.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In Anna Korhonen, David Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1472. URL https://aclanthology.org/P19-1472.

A LIMITATION

The experiment is currently limited to evaluation using OLMo-2-1B and OLMo-2-7B, and experiments on different base models need further exploration. All of our experiments are English-only, drawing both the reference sets and the source corpus from English data. Nothing in the method is language-specific — the reference set and the source pool are freely swappable — but we have not verified that the gradient signal behaves the same way for multilingual references or sources, and we leave that to future work.

Relatedly, we filter from DCLM, which is already a curated, quality-filtered corpus. How Benchmax behaves on a substantially noisier pool, such as raw CommonCrawl or unfiltered FineWeb, is untested. It is plausible that a noisier pool makes the coverage signal more valuable, since there is more to discard, but it is equally plausible that gradient alignment becomes less reliable when candidate quality varies more widely.

On the other hand, while Benchmax exhibits generalization capability to unseen benchmarks, ultimately the extent a filtering method can generalize is still limited by the source pool. If the source pool does not contain any data beneficial to a specific benchmark, no filtering method can improve performance on that benchmark. We currently only filter from DCLM, and generalization capability for larger, more diverse pools remains an open question.

Our experiments are also confined to mid-training on a 25B-token budget. The filtering pipeline itself scales computationally to far larger corpora, as the only component whose cost grows with corpus size is the classifier scoring pass. However, the actual effectiveness of Benchmax on trillion-token-scale pretraining remains an open question.

B ADDITIONAL RESULTS FOR GENERALIZATION EXPERIMENTS

We also perform an additional intra-benchmark generalization experiment, where each benchmark is split into a training and a test set. Only the training set is visible to the filtering method when deciding which data to select, and only the test set is used for evaluation. The results are reported in Table 2. We find that Benchmax is still able to select data that improves performance on the unseen test set, which further validates the generalization capability of our method.

Table 2: **Test-split results on the Table 1 benchmarks.** We evaluate models mid-trained on data selected by our filtering methods at two scales, reporting the *test* split of each benchmark, using the same benchmarks reported in Table 1. Each group is headed by its average and by the average relative difference against HQ DCLM over that group's benchmarks, with lower-is-better metrics sign-flipped so that positive values always indicate improvement. Each non-baseline cell also gives, in parentheses, its difference against that group's HQ DCLM column. Those are raw differences rather than sign-flipped ones, so on the lower-is-better rows a negative value is the improvement. Averages, relative differences, and bold/underline are all computed within a model-size group. Bold indicates the best score per row for that model size. Underline indicates second-best.

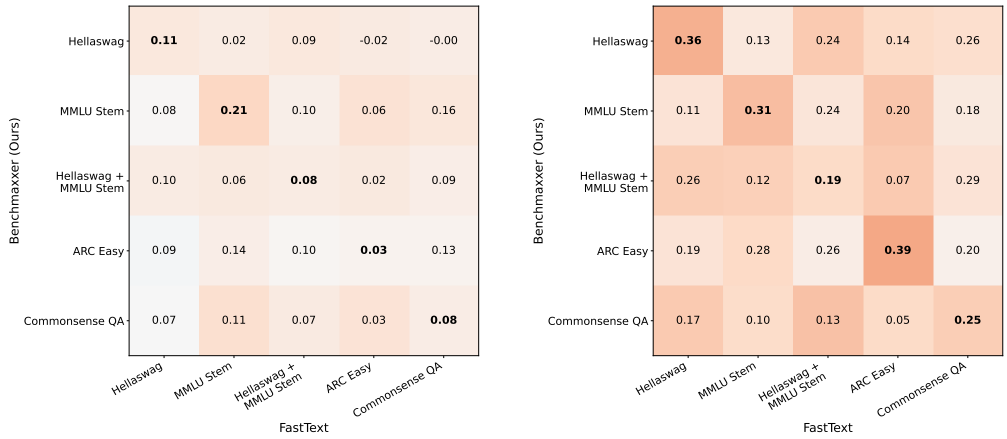
	7B Model			1B Model	
Benchmark	**HQ DCLM**	**Benchmax**	**fastText**	**HQ DCLM**	**Benchmax**
Average Test (↑)	61.7	**62.2** (+0.5)	60.1 (-1.6)	50.6	**54.9** (+4.3)
Avg. Rel. Diff. vs. HQ DCLM	—	**+1.12%**	-2.75%	—	**+12.85%**
Arc Challenge Test	46.7	**49.3** (+2.6)	44.7 (-2.0)	**44.0**	**44.0** (+0.0)
Arc Easy Test	82.1	**82.8** (+0.7)	81.1 (-1.0)	75.4	**76.5** (+1.1)
BoolQ Test	**82.1**	80.6 (-1.5)	79.8 (-2.3)	**66.7**	65.7 (-1.0)
Commitment Bank Test	**57.1**	50.0 (-7.1)	32.1 (-25.0)	**42.9**	**42.9** (+0.0)
CommonSenseQA Test MC 5-shot	69.7	**70.1** (+0.4)	69.4 (-0.3)	27.4	**55.7** (+28.3)
COPA Test	90.0	**92.0** (+2.0)	90.0 (+0.0)	**80.0**	78.0 (-2.0)
HellaSwag Test	**78.6**	77.9 (-0.7)	78.4 (-0.2)	67.5	**68.0** (+0.5)
MMLU Humanities Test MC 5-shot	55.7	**56.9** (+1.2)	53.8 (-1.9)	33.4	**42.7** (+9.3)
MMLU Humanities Test QA	**38.7**	37.8 (-0.9)	36.6 (-2.1)	34.3	**34.5** (+0.2)
MMLU Other Test MC 5-shot	67.8	**68.6** (+0.8)	67.5 (-0.3)	38.4	**52.2** (+13.8)
MMLU Other Test QA	49.8	**50.3** (+0.5)	49.0 (-0.8)	45.9	**46.5** (+0.6)

	7B Model			1B Model	
Benchmark	**HQ DCLM**	**Benchmax**	**fastText**	**HQ DCLM**	**Benchmax**
MMLU Social Sci. Test MC 5-shot	**72.5**	72.2 (-0.3)	72.1 (-0.4)	39.6	**55.4** (+15.8)
MMLU Social Sci. Test QA	45.5	**48.5** (+3.0)	45.6 (+0.1)	41.8	**44.0** (+2.2)
MMLU Stem Test 0-shot	49.6	**51.0** (+1.4)	48.1 (-1.5)	26.1	**35.6** (+9.5)
MMLU Stem Test MC 5-shot	49.7	**51.8** (+2.1)	50.3 (+0.6)	31.9	**40.2** (+8.3)
MMLU Stem Test QA	40.7	**42.6** (+1.9)	39.3 (-1.4)	34.5	**35.7** (+1.2)
MRPC Test	**82.1**	**82.1** (+0.0)	**82.1** (+0.0)	76.6	**80.9** (+4.3)
OpenBookQA Test	43.2	**45.2** (+2.0)	44.0 (+0.8)	39.2	**43.2** (+4.0)
PIQA Test	81.0	**81.9** (+0.9)	81.6 (+0.6)	75.5	**76.9** (+1.4)
RTE Test	60.4	61.2 (+0.8)	**62.6** (+2.2)	**59.7**	59.0 (-0.7)
Social IQA Test	**47.6**	46.1 (-1.5)	46.4 (-1.2)	**48.5**	46.7 (-1.8)
SST-2 Test	55.3	**59.9** (+4.6)	58.3 (+3.0)	69.0	**72.2** (+3.2)
Winogrande Test	**72.7**	72.6 (-0.1)	70.7 (-2.0)	65.5	**65.6** (+0.1)
Average (↓)	0.837	**0.826** (-0.011)	0.851 (+0.014)	1.266	**1.081** (-0.185)
Avg. Rel. Diff. vs. HQ DCLM	—	**+0.95%**	-1.94%	—	**+9.70%**
CommonSenseQA Test MC 5-shot BPB	1.212	**1.190** (-0.022)	1.234 (+0.022)	2.248	**1.742** (-0.506)
GSM8K Gold 5-shot BPB	**0.427**	0.429 (+0.002)	0.442 (+0.015)	0.527	**0.507** (-0.020)
Natural Qs. Open PPL Test	0.871	**0.858** (-0.013)	0.876 (+0.005)	1.023	**0.995** (-0.028)

C Additional Design Choice Analysis

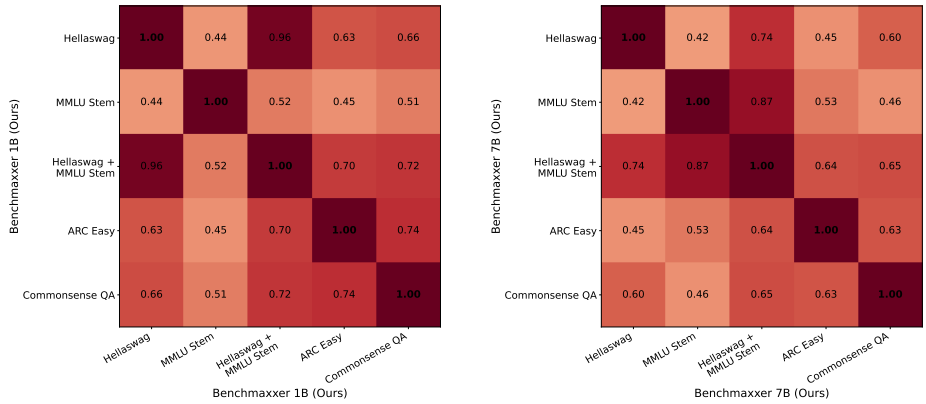
	HQ DCLM	Collapsed Grad	Benchmax	fastText	fastText + Benchmax
zero-shot	0.2628	0.2541	**0.3539**	0.3376	0.3486
5-shot	0.3148	0.2999	0.4019	0.3860	**0.4085**
QA	0.3271	0.3260	0.3302	0.3396	**0.3738**

Table 3: Comparison of HQ DCLM, Mean Grad, Benchmax, fastText, and Mix across zero-shot, 5-shot, and QA evaluation settings. Bold values indicate the best-performing method for each setting.



(a) 1B Gradients (b) 7B Gradients

Figure 4: Pearson Correlation between the fastText baseline and Benchmax over a 1000 DCLM examples. **Left** shows the correlation when Benchmax utilizes OLMo-1B for scoring; while, **right** shows the correlation when Benchmax utilizes OLMo-7B for scoring. Interestingly, we see here that there is a higher correlation in scores with the 7B model compared to the 1B model. Nevertheless, we see that there isn't a strong correlation between the scores produced.



(a) 1B Self Correlation (b) 7B Self Correlation

Figure 5: Pearson Correlation between Benchmax and itself over a 1000 DCLM examples. **Left** shows the correlation when Benchmax utilizes OLMo-1B for scoring; while, **right** shows the correlation when Benchmax utilizes OLMo-7B for scoring.

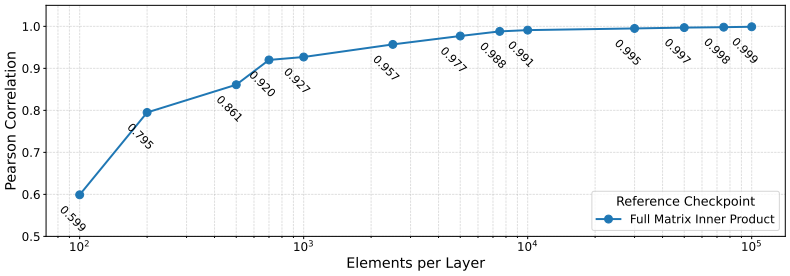


Figure 6: CountSketch ablation on 7B on different values for the dimension or number of elements per layer.

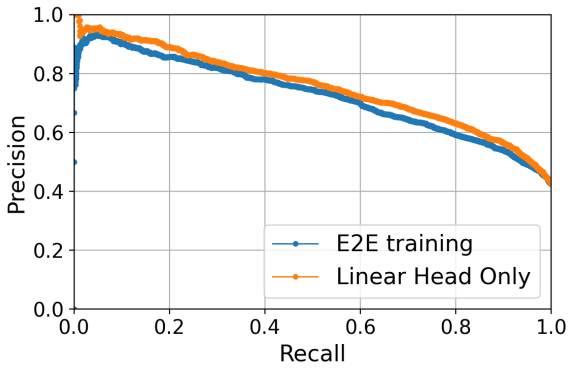


Figure 7: Precision–recall curve of the distilled classifier on CommonSenseQA.

D BENCHMARK CONTAMINATION

Because Benchmax selects documents by their alignment with benchmark examples, a natural concern is that it concentrates benchmark text itself into the training set. We measure this directly with a 13-gram near-duplicate overlap check (`dolma dedupe` from the Dolma toolkit, Soldaini et al., 2024) between the benchmark test/validation examples and two corpora: the raw pre-filter pool, and the 25B-token set selected by the Benchmax coverage filter using 7B-model gradients. A benchmark example counts as contaminated if at least 0.8 of its 13-gram windows match. 62,036 examples across 25 benchmarks are checked.

Overlap is **0.37% in the selected set versus 1.34% in the raw pool** — that is, selection more than halves the rate rather than concentrating benchmark text. Table 4 gives the per-benchmark breakdown for every benchmark above 1% in either corpus. The ranking differs between the two: BoolQ and the ARC family dominate in the selected set, while the MMLU families appear largely in pool content that the filter removes.

Table 4: **Benchmark contamination before and after filtering.** Fraction of benchmark examples whose 13-gram overlap with the corpus exceeds 0.8, with example counts in parentheses. "Before" is the raw pre-filter pool; "after" is the selected 25B-token set. Benchmarks above 1% in either corpus are listed individually.

Benchmark	# examples	Before	After
MMLU Humanities (QA)	3,626	9.85% (357)	0.17% (6)
BoolQ	3,270	7.28% (238)	5.87% (192)
MMLU Humanities (0-shot)	4,705	2.06% (97)	0.02% (1)
MMLU Social Sci. (0-shot)	3,077	1.75% (54)	0.00% (0)
Arc Challenge	299	1.34% (4)	1.34% (4)
All other benchmarks	47,059	0.18% (84)	0.05% (25)
Overall	**62,036**	**1.34% (834)**	**0.37% (228)**

E COST OF THE FILTERING PIPELINE

Table 5 gives the analytical FLOP cost of each pipeline stage, and Table 6 the measured GPU-hours on the hardware we actually used. From the breakdown, filtering is a modest fraction of the end-to-end cost: roughly 31% of the combined filter-plus-train budget at 1B, and 10% at 7B, where the training run dominates. Stage 3b looks large only because it is the one stage that touches the entire corpus. The scoring is forward-only and roughly 150× cheaper than a gradient pass, and the cost also amortizes across every downstream run that reuses the filtered set. Two observations are worth recording, both of which correct impressions that the raw numbers might otherwise give. First, Stage 2 uses only 14% of its allocated GPU time at 1B: the remainder is the GPU idling behind network storage while it reads one gradient file per document. This is an artifact of our storage layout, not of the method — sharding the pool gradients would cut Stage 2's wall clock by roughly 70% with no algorithmic change. Second, the per-document gradient cost ratio between the 7B and 1B proxy models is 4.65×, close to the parameter ratio.

Table 5: **Analytical FLOPs by pipeline stage.** A 1M-document candidate pool, 3,000 reference examples, 100k classifier-training samples over 3 epochs, ModernBERT-base, a 250B-token corpus to score, and a 25B-token mid-training run.

Stage	**1B**	**7B**
Stage 1 — gradient scoring	7.4×10^{18}	4.5×10^{19}
Stage 2 — sketched score matrix	7.2×10^{14}	4.4×10^{15}
Stage 3a — classifier distillation	1.4×10^{17}	1.4×10^{17}
Stage 3b — corpus scoring (250B tokens)	7.5×10^{19}	7.5×10^{19}
Filtering subtotal	$\approx 8.3 \times 10^{19}$	$\approx 1.2 \times 10^{20}$
One mid-training run (25B tokens)	1.8×10^{20}	1.1×10^{21}
Filtering as % of end-to-end	**∼31%**	**∼10%**

Table 6: **Measured cost on RTX A5000, bf16.** Stage 1 is paid once per (proxy model, pool) and reused by every target task; Stages 2 and 3 are paid per task. Figures carry roughly $\pm 10\%$, except the Stage 2 GPU-active column, which is dominated by a single measured rate and should be read as $\pm 25\%$.

	GPU-active hours		**GPU-allocated hours**	
Stage	**1B**	**7B**	**1B**	**7B**
Stage 1 — pool gradients (once)	87.2	406.7	98.4	429.6
Stage 2 — selection (per task)	0.20	0.97	1.46	2.42
Stage 3 — classifier FT (per task)	5.54	5.54	5.57	5.57

F Additional Experiment Details

Training details. For mid-training experiments, we continue training OLMo-2-1B from the stage-1 checkpoint on filtered data using the Dolmino-mix or DCLM corpus. Key hyperparameters:

- Optimizer: AdamW with lr $= 7.4487 \times 10^{-5}$ (scaled by $\sqrt{k}$ factor when filtering changes the effective batch size).
- Batch size: global batch size of 512–1024 before filtering.
- Sequence length: 4096 tokens ($\sim$4M tokens per step at batch size 1024).
- Scheduler: Linear decay.
- Target training tokens: 25B tokens.

To train 7B models, we use about 640 H100s gpu hours per model on the `https://github.com/togethercomputer/xorl` codebase. To train 1B models, we use about 512 AMD MI300A hours per model on the `https://github.com/allenai/olmo` codebase.

Construction of Multi-benchmark fastText-filtered data. We concatenate data from each of the benchmark used to train individual linear heads of Benchmax into a single giant reference dataset, and label is as the high quality split for fastText training. The average quality split is again the DCLM split of OLMO mid-training corpus (OLMo et al., 2024). We apply the same threshold selection procedure descrbied in section 4.3 to find a cutoff for the high-quality-split probability scored by the fastText classifier that yields a 25B-token filtered dataset. The accuracy of this model was $\sim$99%.

G ADDITIONAL ANALYSIS/GRAPHS

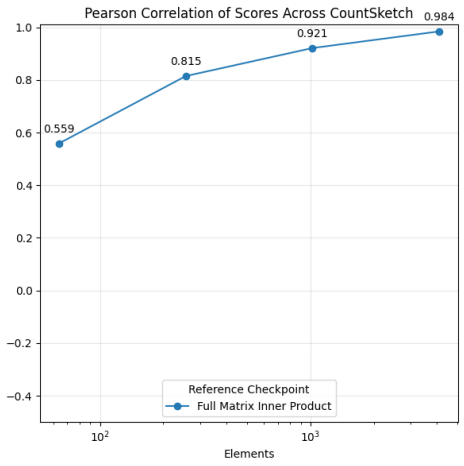


Figure 8: CountSketch Ablation on 1B OLMo Model.

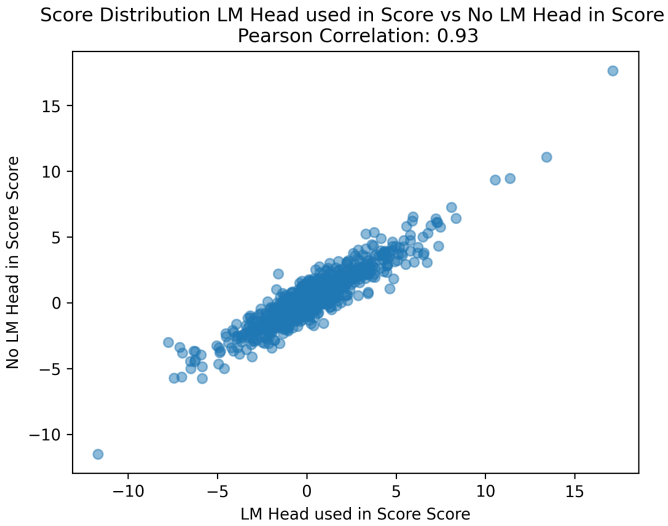


Figure 9: Measuring the Pearson correlation between considering the lm-head of a model vs not.

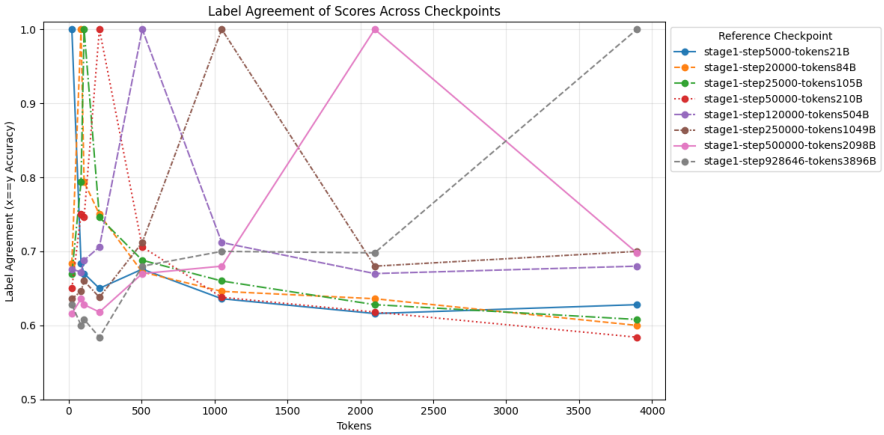


Figure 10: This graph shows the label agreement across various checkpoints during pretraining of the 7B OLMo model.

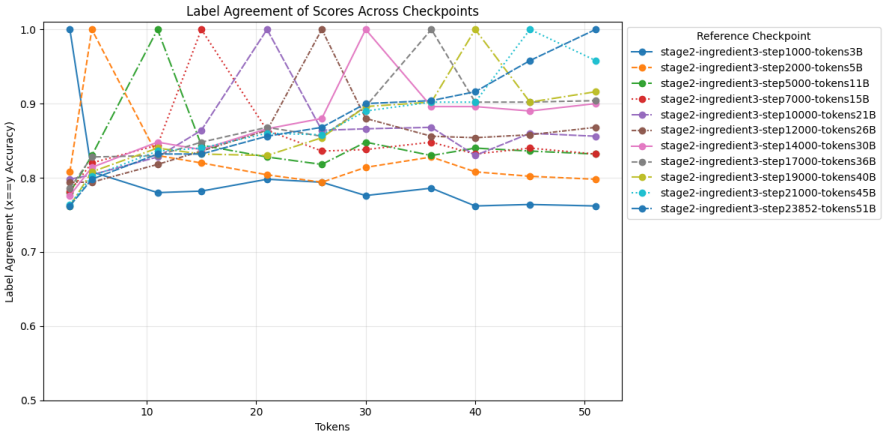


Figure 11: This graph shows the label agreement across various checkpoints during mid-training of the 7B OLMo model.

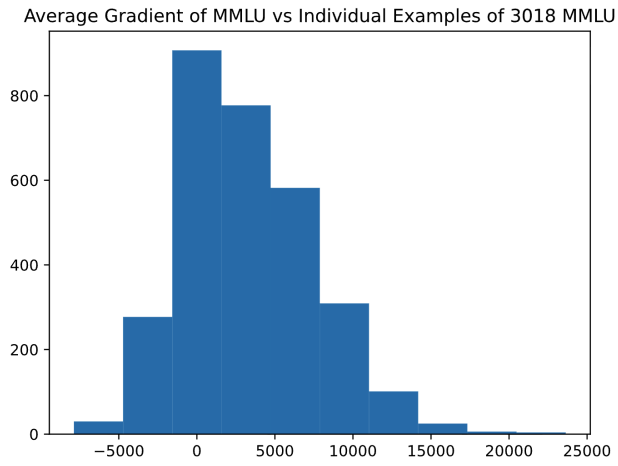


Figure 12: Measuring the inner product scores between the average gradient of MMLU STEM and the individual example gradients. This shows that the average gradient in MMLU STEM would not have selected many of the examples found in MMLU STEM.

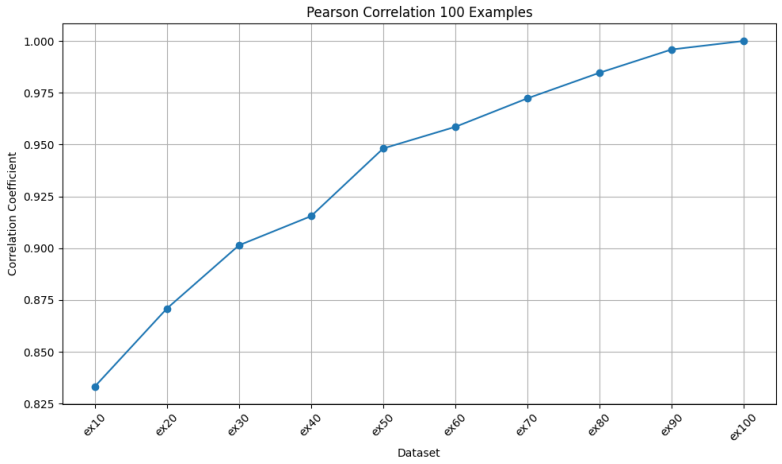


Figure 13: This experiment was to understand how well an averaged ten example gradients correlated averaged 100 gradients.

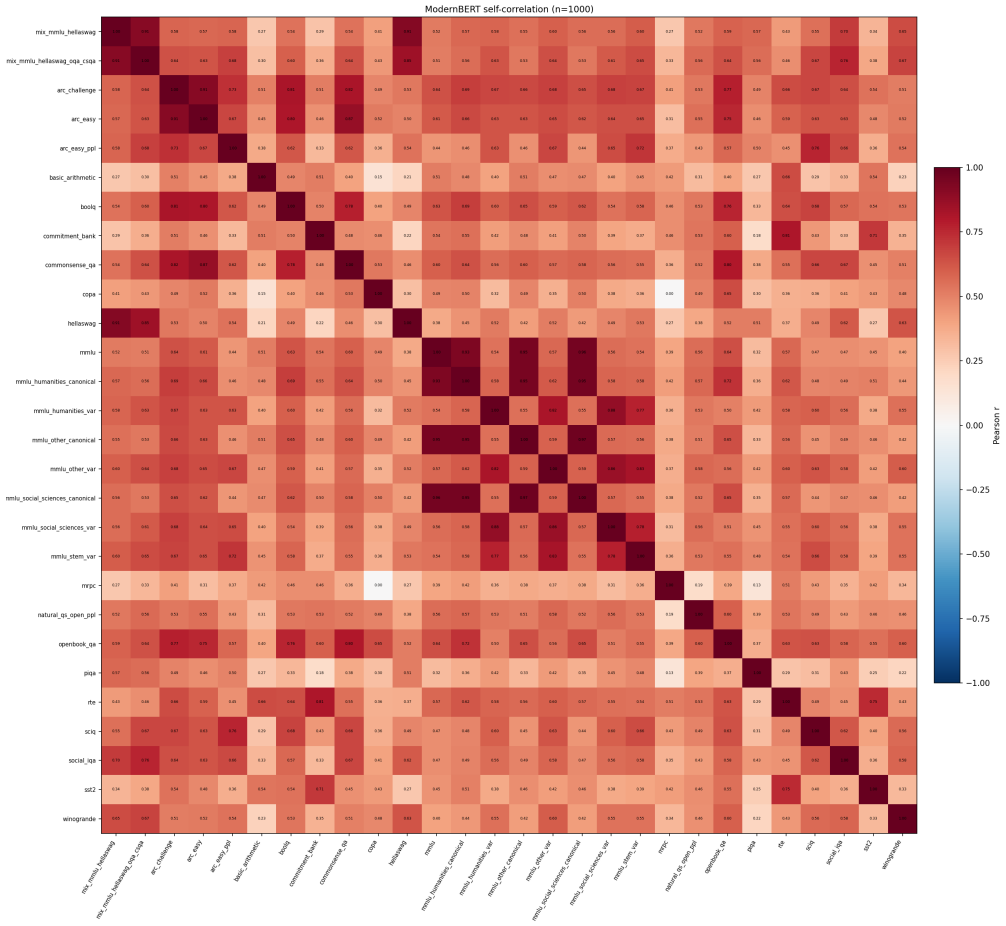


Figure 14: Self-correlation matrix of the modernBERT classifiers. Note that some of these are not used in the experimental settings in the main paper.

H Analysis of Top Scored Examples for fastText and Benchmax

We take the top 20 examples scored by fastText and Benchmax (with 1B gradients) and analyze their content to understand the differences in what each method retrieves. For fair comparison, we ask Claude to answer two questions about each example: (1) Does this example look like a multiple-choice question? and (2) Is this example STEM-related? Table 7 summarizes whether each top-scored example is multiple-choice formatted and STEM-related. We find that fastText disproportionately retrieves multiple-choice formatted text, while Benchmax retrieves thematically relevant scientific content even when it is not multiple-choice formatted. This supports our hypothesis that fastText's lexical matching captures stylistic cues (e.g., multiple-choice question phrasing) while Benchmax's gradient-based coverage captures both style and semantic relevance. Raw text of top three examples selected by each method is provided in Sections H.1 and H.2.

Table 7: **Top-20 selected documents.** The twenty highest-scored documents from fastText and from Benchmax, labeled by hand as multiple-choice formatted and as STEM by domain. fastText concentrates on multiple-choice format almost irrespective of subject, whereas Benchmax is less format-bound and skews toward more technical content. Sections H.1 and H.2 give the raw text.

Top-20 selected	MC-formatted	STEM (by domain)
fastText	90%	45%
Benchmax	60%	55%

H.1 fastText Top Samples

Example 1 (score: 0.5731) Multiple-choice: Yes STEM-related: No
Chinese English grammar exam with A/B/C/D options on vocabulary and syntax.

```
2019-11-12 13:41:00
[Chinese-language header omitted]
I. Phonetics (5 points)
Vocabulary and Structure (15 points)
A. walk   B. the walk
C. wallks   D. a walk
A. Why   B. Where
C. Who   D. Whose
8.  My father asked __ to help with his work.
A. I and Tom   B. Tom and me
C. me and Tom   D. Tom and I
9.  Nowadays little knowledge __ to be a dangerous thing.
A. seem   B. seemed
C. does seem   D. do seem
A. will increase   B. would be increasing
C. will have increased   D. would have been increasing
A. unless   B. in case
C. until   D. so that
A. too   B. as well
C. neither   D. either
A. As   B. Once
C. That   D. It
A. lacking   B. losing
C. missing   D. dropping
A. should he have arrived   B. would he have arrived
C. did he arrive   D. had he arrived
A. That   B. Whom
C. What   D. How
```

```
A. speech   B. debate
C. discussion   D. argument

A. faint   B. blank
C. dark   D. blind

A. solved   B. caused
C. covered   D. met

A. cares for   B. allows for
C. applies for   D. calls for
```

Example 2 (score: 0.5684) Multiple-choice: Yes STEM-related: Yes
Microbiology quiz on Streptococcus with A/B/C/D options and correct-answer labels.

```
Streptococcus - Section 1

A. Streptococcus mutans
B. S sanguis
C. Both (a) and (b)
D. Streptococcus pyogenes
Correct Answer:  C. Both (a) and (b)

A. Schick test
B. Disk test
C. ASO test
D. Precipitation test
Correct Answer:  B. Disk test

3.  Which of the following condition is non-suppurative
sequelae of
Streptococcus pyogenes infections?
A. Acute rheumatic fever
B. Acute glomerulonephritis
C. Erythema nodosum
D. All of these
Correct Answer:  D. All of these

4.  Which of the following cross-reacts with cell wall
carbohydrate of
Streptococcus pyogenes?
A. Cardiac valves
B. Synovial fluid
C. Myocardium
D. Vascular intima
Correct Answer:  A. Cardiac valves

5.  Streptococcus pyogenes can be differentiated from other
hemolytic
Streptococci on the basis of
A. Bacitracin sensitivity
B. Erythromycin sensitivity
C. Aminoglycosides sensitivity
D. Penicillin sensitivity
Correct Answer:  A. Bacitracin sensitivity

Leave A Comment?  ten - three =
```

Example 3 (score: 0.5680) Multiple-choice: Yes STEM-related: No
Indian general-knowledge quiz; only ∼2 of 20 questions touch science (vermicomposting, hydrogenation catalyst).

```
Home Gk question TOP GK QUESTIONS||GK question in english

Who was the master of Kabir
A. Vallabhacharya   B. Ramanand   C. Ramanuja   D. Namdev
Answer B. Ramananda

When was the "National Vyashree Yojana" launched?
A. April 1, 2016   B. April 1, 2017
C. 1st July 2017   D. July 1, 2016
```

Answer B. April 1, 2017

That constitution amendment under which the GST was passed
A. 122th B. 121th C. 123th D. 124th
Answer A. 122th

At present the maximum number of members of Rajya Sabha can be
A. 250 B. 252 C. 245 D. 238
Answer A. 250

Vermicomposting is done by
A. Fungus B. Animals C. worm D. bacteria
Answer C. worm

Which of the following is not located in Western Ghats?
A. Pipali Ghat B. Bhor Ghat C. Thal Ghat D. Pal Ghat
Answer A. Peepli Ghat

What was the original name of Swami Vivekananda
A. Narendranath Dat B. Surendra Dut
C. Krishna Dutt D. Batukeshwar Dutt
Answer A. Narendranath Dut

Bhilai Iron-Steel Plant is set up in collaboration with which country?
B. Russia C. Germany D. United States
Answer B. Russia

Which latitude line passes through the middle of India
A. Equation Line B. Cancer line
C. Capricorn Line D. Arctic line
Answer B. Cancer line

Oil is used for hydrogenation
B. Ni C. Mo D. Fe
Answer B. Ni

Which of the following was the ‘Sage movement’?
A. Sheikh Nuruddin B. Chaitanya Mahaprabhu
C. Ramanand D. Bithova
Answer: A

What is the literal meaning of ‘Arya’?
A. Veer B. Best or Elite C. Scholar D. Yagyaar
Answer: B

Vaishnava reached its climax in the preservation of the rulers?
A. Maurya B. Kushan C. Shung D. Secret
Answer: D

Which is mentioned in the tenth Circle of Rig Veda for the first time?
A warrior B. Priest C. Shudra D. Chandal
Answer: C

According to the Jain tradition, how many pilgrims have been in Jain religion?
A. 25 B. 20 C. 24 D. 23
Answer: C

Where did Baba Ramchandra organize the farmers?
A. Awadh B. Bihar C. Bengal D. Andhra Pradesh
Answer: A

Which of the following historians has called Akbar an enemy of Islam?
A. Abbas Khar Sarvani B. Badayun
C. Ahmed Khan D. Mir Alaudaula Kazivini
Answer: B

The column article ‘Mahaloli’ is related to which ruler?
A. Chandragupta II B. Chandragupta Maurya

```
C. Ashok   D. Samudragupta
Answer:  A

Which ruler of the Gupta period is called 'Kaviraj'?
A. Shri Gurupta   B. Chandragupta II
C. Samudragupta   D. Skand Gupta
Answer:  C

What was the currency of gold in secret carpet?
A. Dinar   B. Mercury   C. Absent   D. Kakini
Answer:  A
```

H.2 BENCHMAX TOP SAMPLES

Example 1 (score: 0.9453) Multiple-choice: Yes STEM-related: Yes
Geometry placement test on parallel-line angle relationships and similar-triangle ratios.

```
Geometry Placement

1.  In the figure below line segment AB, line segment CD,
and line segment EF
are parallel, and line segment PQ intersects all 3 lines at
points R, S, and
T, respectively.  If the measure of angleQTF is 33°, what is
the measure of
anglePRB?
A. 33°   B. 57°   C. 66°   D. 123°   E. 147°

2.  In triangleMPB below, line segment LA is parallel to
line segment MB.
If PL/LM = 5/3, then PB/PA = ?
A. 5/8   B. 2/3   C. 8/5   D. 5/3   E. 8/3

Answers:  1.  E   2.  C
```

Example 2 (score: 0.9414) Multiple-choice: No STEM-related: Borderline
Descriptive article on rotated-die coin mint errors; centers on degree measurements (geometric concept) but is primarily a numismatics reference.

```
Rotated Dies

The image to the left is an example of a 1994 cent whose
dies have been
rotated 165 degrees.

In a normal U.S. coin the obverse and reverse are struck in
such a manner
that if you are looking at the obverse image right side up
and flip the coin
over the reverse image will be right side up.

Classifications of rotated dies can be as high as 180
degrees.  Since it is
unknown which die, the obverse, reverse, or combination of
both, was out of
place, the error is referred to as a "rotated dies" mint
error.

The U.S. Mint allows for an error margin of less than 8
degrees.  Rotated die
errors are usually not collectible unless they are at least
15 degrees.
Rotated die errors are most valuable after 90 degrees with
180 degrees being
the most valuable.

1910:  90 degrees   1920:  110 degrees   1924-D: 120 degrees
1924-D: 180 degrees   1936-D: 95 degrees   1940-S: 129
degrees
1966:  135 degrees   1972-D: 90 degrees   1972-D: 100
```

```
degrees
1972-D: 120 degrees   1972-D: 140 degrees   1972-D: 150
degrees
1973-D: 90 degrees   1973-D: 100 degrees   1973-D: 105
degrees
1973-D: 120 degrees   1973-D: 135 degrees   1973-D: 170
degrees
1993-D: 90 degrees   1993-D: 180 degrees   1994:  165
degrees

If you are interested in figuring out how much your coin is
rotated, you can
build a "Coin Degree Finder" by clicking
```

Example 3 (score: 0.9375) Multiple-choice: Yes STEM-related: Yes
IMO Class 8 inverse-proportion word problem on food supply duration.

```
User Forum

Subject:  IMO   Class:  Class 8

In a camp there is sufficient food for 105 soldiers for 21
days.  If 42
soldiers leave the camp, how long would the food last?

A 30
B 35
C 65
D 45

the solution is wrong

Ans 1:
Class:  Class 9
Answer will be b solve it nicely

Post Your Answer
```