SubLIME: Subset Selection via Rank Correlation Prediction for Data-Efficient LLM Evaluation

Gayathri Saranathan[1*], Cong Xu[1*], Mahammad Parwez Alam[1], Tarun Kumar[2], Martin Foltin[3], Soon Yee Wong[1], Suparna Bhattacharya[2]
Hewlett Packard Labs, [1]Singapore, [2]Bangalore, [3]US
gayathri.saranathan@hpe.com, cong.xu@hpe.com

Abstract

The rapid expansion of Large Language Models (LLMs) and natural language processing datasets has made exhaustive benchmark evaluations computationally prohibitive. Inspired by high-stakes competitions like the International Mathematical Olympiad—where a few well-chosen problems suffice to differentiate top performers—we present **SubLIME**, which reduces evaluation costs by 80% to 99% while preserving ranking fidelity. It trains a *Rank Correlation Prediction* (RCP) model that combines limited performance data from only 5–20 anchor LLMs with dataset intrinsic metrics—*Difficulty*, *Quality*, and *Distributional Dispersion*—to predict how closely a candidate subset reflects full-benchmark rankings. Guided by these predictions, SubLIME selects a "winning" subset (1–20% of full set data) for evaluating new LLMs, preserving global rankings significant better than other data-efficient methods across ten diverse benchmarks.

1 Introduction

The exponential growth of large language models (LLMs) has made evaluation increasingly resource-intensive. With over 1 million open-source models available on HuggingFace (including more than 170,000 text generation models) and prominent leaderboards such as the Open LLM Leaderboard and AlpacaEval tracking over 100,000 models, there is a pressing need for scalable evaluation methodologies. Meanwhile, the existence of over 2,500 NLP benchmarks on PapersWithCode and nearly 200,000 NLP datasets on HuggingFace makes it impractical to evaluate every LLM on every benchmark. For example, the HELM project (Liang et al., 2023) spent \$50,000[1] to assess 30 models on 13 tasks. Scaling this to evaluate just 10% of existing text-generation LLMs on 100 benchmarks could approach \$100M. These costs are further exacerbated by inference-time scaling, where advanced LLMs (e.g., OpenAI's O1, DeepSeek R1) generate over 10 times more tokens per response than their base models.

In response to these escalating challenges, we introduce **SubLIME** (Subset-centric Less Is More Evaluation), a framework that achieves data-efficient evaluation by identifying highly representative subsets. Recent approaches (Vivek et al., 2024; Polo et al., 2024; Perlitz et al., 2024; Prabhu et al., 2024) explored benchmarking without the full dataset; however, the warm-up phase poses a significant challenge for most. LLM leaderboards often add new benchmarks before many models have been evaluated on them, creating a bootstrap problem for data-efficient methods that typically need extensive prior knowledge.

Our key solution is the *Rank Correlation Prediction (RCP) model*, which integrates partial evaluations from 5–20 anchor LLMs with intrinsic benchmark metrics—*Difficulty* (D), *Quality* (Q), and *Distributional Dispersion* (DD)—to select an optimal subset that preserves the full-benchmark ranking (Details provided in Appendix A.4.1). Table 1 demonstrates that examples with higher difficulty and lower quality tend to exhibit higher Oracle Difficulty—*a measure defined as the fraction of models failing to solve a datapoint* — which in turn signals potential performance degradation. This observation motivates our strategy of leveraging intrinsic benchmark characteristics to predict rank correlation and select an efficient evaluation subset without requiring exhaustive assessment.

Inspired by high-stakes competitions such as the International Mathematical Olympiad (IMO), where a handful of carefully chosen problems effectively differentiate top performers, SubLIME harnesses these universal "meta-principles" to design representative subsets comprising only 1–20% of the full data. Unlike approaches that rely on a fixed

[*] Equal contribution

[1]Actual cost breakdown: \$38,001 for commercial APIs, plus 19,500 A100 GPU hours at \$1/hr

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 30572–30593
July 27 - August 1, 2025 ©2025 Association for Computational Linguistics

Category	Input Text	D	Q	DD	Oracle D
High Oracle D	Find a movie similar to The Sword in the Stone, Fantasia, The Jungle Book, Willy Wonka & the Chocolate Factory: Options: (A) The Wizard of Oz (B) Captain Corelli's Mandolin (C) Timecrimes (D) Arlington Road	0.747	0.269	0.559	0.805
	Find a movie similar to The Shawshank Redemption, Saving Private Ryan, Braveheart, The Matrix: Options: (A) Dracula 2000 (B) 1492 Conquest of Paradise (C) Schindler's List (D) Auto Focus	0.739	0.168	0.59	0.859
Low Oracle D	I have a cow, three mice, a dog, a duck, and two snakes. How many animals do I have?	0.09	0.729	0.65	0.09
	I have a goat, a frog, five pigs, and a bear. How many animals do I have?	0.08	0.665	0.68	0.054

Table 1: Demystifying Input text's Difficulty (D), Quality (Q), Distributional Dispersion (DD) correlation with Oracle Difficulty (Oracle D) - BBH benchmark

strategy or necessitate full benchmark evaluations, SubLIME adapts to each benchmark's unique characteristics by combining partial evaluations with dataset-level D, Q, and DD metrics. This integration allows us to predict a subset that yields model rankings closely aligned with those from the complete dataset. Extensive experiments on 10 diverse benchmarks and over 100 LLMs—demonstrate that SubLIME preserves rank correlations more effectively than random sampling and other baseline methods. Our key contributions include:

1.**RCP model** that combines 3 intrinsic dataset metrics and evaluation results of 5 to 20 anchor LLMs to guide the selection of optimal subset.

2.**Framework** for efficient evaluation on new benchmarks without requiring extensive results.

2 Related Work

Data-efficient training has been widely studied for model training on image data (Ding et al., 2023; Sorscher et al., 2023) and language tasks (Marion et al., 2023; Xie et al., 2023). Methods include coreset selection, importance sampling, and difficulty sampling to use smaller, representative datasets (Zayed et al., 2023; Guo et al., 2022). SubLIME explores diverse sampling strategies in LLM and text-to-image model evaluation, aiming to maintain model rankings and score distributions.

Efficient LLM evaluation was recently introduced in techniques like AnchorPoints (Vivek et al., 2024) and TinyBenchmarks (Polo et al., 2024), which use coreset and item response theory (IRT) to select a subset of evaluation instances, closely estimating full benchmark scores. These methods align with Lifelong Benchmarks (Prabhu et al., 2024), which expands candidate examples and selects a subset based on difficulty. FlashHELM (Perlitz et al., 2024) optimizes evaluation resources based on estimated leaderboard positions, prioritizing higher-ranked models. However, these competitive approaches often assume the availability of extensive evaluation data across benchmarks, which may not be practical during the warm-up phase of a new benchmark. They don't address the challenge of initializing evaluations on new datasets with limited initial results. Our work uniquely addresses this gap by proposing a prediction model that can operate effectively with minimal initial data.

3 Analogy and Intuition of Our Approach

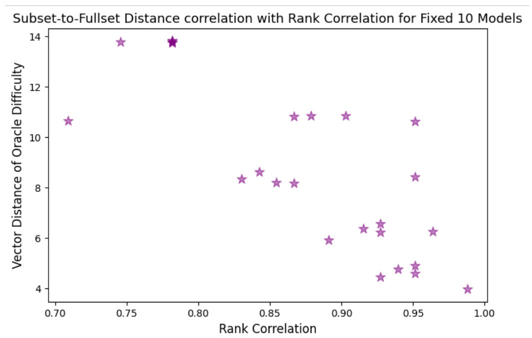


Figure 1: Inverse correlation Subset to Fullset (BBH Benchmark) Distribution Distance and Rank correlation

Six carefully chosen IMO problems differentiates contestants through several factors: (a) a structured *difficulty gradient*, allowing more nuanced discrimination among contestants; (b) *diversity* of problem types (e.g., geometry, combinatorics, and number theory) for comprehensive assessment; and (c) *Quality and clarity* of language, ensuring problem statements are unambiguous and robust indicators of mathematical ability. These principles ensure that a few well-selected problems can approximate the ranking of a much larger exam. Similarly, our RCP model selects a subset of data that preserves the overall ranking of LLMs relative to the full benchmark. To achieve this, we combine three intrinsic metrics: (a) **Difficulty** captures a gradient of challenge levels, ensuring the subset includes examples that effectively distinguish between stronger and weaker models. (b) **Quality** measures the clarity and precision of the examples, akin to the unambiguous language of well-crafted IMO problems. (c) **Distributional Dispersion** ensures comprehensive coverage of the benchmark's content by representing different "topics" or regions of the data distribution. In addition to dataset metrics, our method leverages empirical performance data from a small set of anchor LLMs—analogous to pilot testing problems on a sample of contestants to identify features that truly distinguish model performance.

Figure 1 illustrates that subsets with similar "difficulty ladder" might preserve rank well. Each point on the scatter plot represents a different subset. We calculate the distributional distance D_s of the difficulty indicators (y-axis), which will be discussed in detail in the next section, to see how well that subset's ranking aligns with the full set's ranking (x-axis). A clear negative correlation emerges: as the subset distance increases (i.e., it diverges from the full set's difficulty profile), the resulting Spearman coefficient decreases.

What Does the Model Learn? During training, the RCP model acquires *meta-knowledge* about D, Q, DD's effect on rank preservation. Rather than merely labeling benchmarks as "hard"/"easy", the model learns to interpret nuanced signals such as text complexity, distance-based diversity, rank alignment ambiguity— similar to an IMO organizer refining problem selection based on pilot testing.

Generalizing to New Benchmarks: Trained across diverse benchmarks, the RCP model internalizes that evaluation problems should be both representative and discriminative to effectively preserve overall rankings. When applied to a new benchmark, the model leverages its learned priors to assess the distribution of D, Q, DD, recommending a subset that closely mirrors the full distribution. As long as the goal remains to rank models in cost-effective evaluation setup, RCP model is well equipped to generalize to new benchmarks.

4 Our Solution - SubLIME

SubLIME consists of three key components:
Adaptive Subset-Selection (SubMS): with multiple sampling methods to select metric-based subsets at a given sampling rate.
Feature Matrix Generation & RCP Model Training (λ): RCP trained with feature matrix consisting of D,Q,DD metrics and evaluation results of a set of anchor LLMs to predict rank correlation.

Algorithm 1 SubMS: Experiment Design

Require: Initialize
Full benchmark data d_b where b is a benchmark in B - representing the collection of benchmarks
Collect sample-level results for $\mathcal{M}_b$ LLMs
Specify Sampling methods $\mathcal{S}$ {Readability, Quality, Clustering, Difficulty}
$d_b \subset b \forall \in B$; ($d_b$ is full data of given benchmark b)
Ensure: Adaptive Sampling for each Benchmark
for each benchmark $b \in \mathcal{B}$ **do**
  for each sampling technique $s \in \mathcal{S}$ **do**
    for sampling rate $x\%$ from 1 to 100 at step size 1% **do**
      $I_{s,x} \leftarrow$ apply s to get indices of $x\%$ subset of d_b
      $Sub_{s,x} \leftarrow d_b[I_{s,x}]$ (subset using $I_{s,x}$)
      $\mathbf{score}_{s,x} \leftarrow \{\text{eval}(m, Sub_{s,x}) \mid m \in \mathcal{M}_b\}$
      $\mathbf{rank}_{s,x} \leftarrow$ argsort($\mathbf{score}_{s,x}$)
      $\rho_{s,x} \leftarrow \rho(\mathbf{rank}_{s,x}, \mathbf{rank}_{\text{full}})$ ▷ Spearman
      $r_{s,x} \leftarrow$r($\mathbf{score}_{s,x}, \mathbf{score}_{\text{full}}$) ▷ Pearson
      $R_s[x] \leftarrow (\rho_{s,x}, r_{s,x})$
    end for
  end for
  criteria $\leftarrow$ Analyze R_s to find minimal x where $\rho \geq 0.9$ and $r \geq 0.9$
end for
$s_b^* \leftarrow$ criteria met by$s \in \mathcal{S}$ at $\min(x\%)$
return Optimal subset $Sub \leftarrow \{(s_b^*, x_b^*)\}_{b=1}^{10}$

Winner Subset Selection and Testset Evaluation: Identify the "winner subset" Sub_{Win} which achieves high rank correlation across different LLMs using RCP inference. Evaluating Sub_{Win} subset on unseen LLMs to ensure preservation of rank order and score distribution. For evaluation, we select 10 Benchmarks B from HuggingFace Open LLM Leaderboards v1 and v2 (Hugging Face, 2022) including TruthfulQA (Lin et al., 2022), ARC (AI2 Reasoning Challenge) (Clark et al., 2018), Winogrande (Sakaguchi et al., 2021), GSM8k (Grade School Math) (Cobbe et al., 2021), Hellaswag (Zellers et al., 2019), GPQA (Rein et al., 2024), MUSR (Sprague et al., 2024), BBH (Suzgun et al., 2022), MATH (Hendrycks et al., 2021), MMLU-Pro (Wang et al., 2024). In addition, we collected sample-level results of LLMs that is evaluated on v1 and v2 Leaderboards. We obtained results for 313 Models (provided in Appendix A). Out of these, 213 LLMs were used to train (M_{train} the RCP model (λ), while the remaining 100 (M_{test} served as a held-out set for Testset Evaluation.

4.1 Adaptive Subset Selection (SubMS)

SubMS is aimed at preserving rank and score distribution of LLMs. We consider the following metric-based sampling approaches: Random Sampling, Clustering-based Sampling, Quality-based Sampling, and Difficulty-based Sampling. Detailed

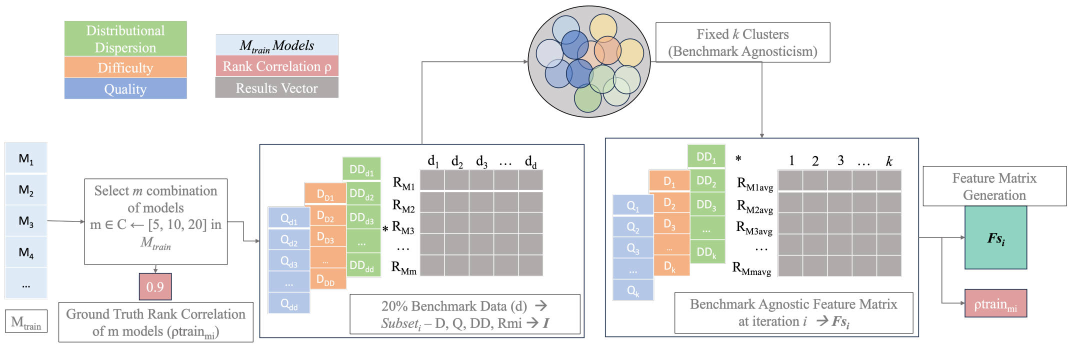


Figure 2: Feature Matrix F_{si} & Subset I_i Generation at Iteration i

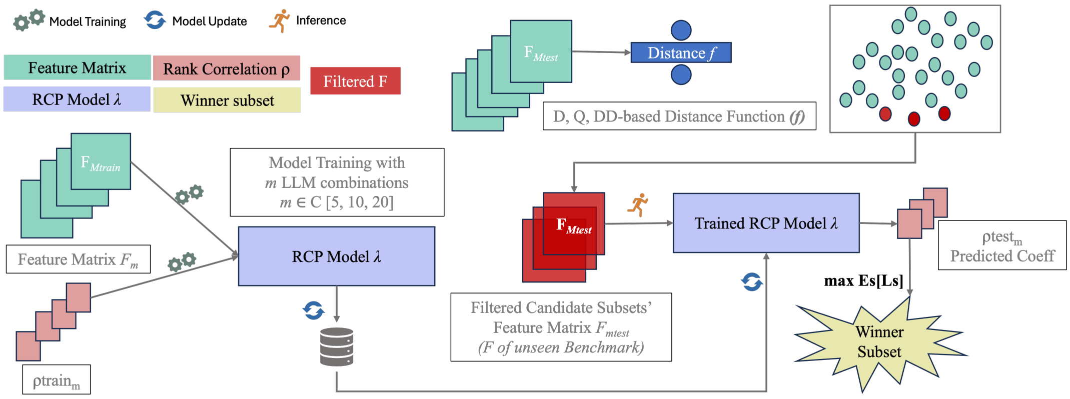


Figure 3: SubLIME: RCP (λ) model training with m combinations. Trained RCP selects Winner Subset Sub_{Win} during Test-time for a given Benchmark

descriptions of these methods can be found in Appendix section A.1.1. With each subset d obtained from respective sampling methods S, we evaluate it based on Rank and Score Preservation w.r.t full set results of the same list of LLMs. The Spearman Coefficient (Sedgwick, 2014) ρ is used for assessing the rank correlation and Pearson Coefficient r is used for assessing Score Distribution Preservation. The implementation of SubMS is described in Algorithm 1.(SubMS Subset Rank and Score correlation at different sampling rate for Hellaswag is shown in Figure 8. Evaluation of SubMS with TinyBenchmarks is provided in Appendix A.3).

4.2 Rank Correlation Prediction Model

To efficiently guide the selection of optimal subset during the warm-up phase of new benchmarks, we introduce a RCP model to select a winner subset (Sub_{Win}) of a given a benchmark. SubLIME incorporate four following phases for RCP Model Training and Evaluation, as shown in Figure 2 and 3. (i) The phases includes unified D, Q, DD metric selection; (ii) Feature Matrix Generation with selected D,Q, DD; (iii) RCP Model Training; (iv) Sub_{Win} Selection for unseen/new benchmarks.

4.2.1 Unifying D, Q, and DD Metrics

To construct the feature matrix, we identify unified metrics for D, Q and DD. For each metric, we use the Kolmogorov–Smirnov(KS) test to select the measure that best exhibits normal distribution properties. Specifically, we adopt **Flesch Readability for Q**, **Gunning Fog for D**, and **BERT-based clustering for DD** These unified metrics are applied consistently across all benchmarks,(see Figure 11 in the Appendix for GPQA benchmark, ablation studies are also included in Section A.4.1)

4.2.2 Feature Matrix Generation

The feature matrix is constructed in a structured to train RCP model, multi-step process to encode intrinsic dataset characteristics and partial model performance(R_m) for each candidate subset. Step-by-step approach of generating feature matrix F ($\forall b \in B$) in Algorithm 2. The feature matrix is designed for two aspects of subset effectiveness.

Intrinsic Characteristics: D, Q, DD metrics are benchmark-agnostic, & are computed without model evaluations. They provide a proxy for discriminative power: A good subset should have high-quality, diverse, appropriately difficult samples.

Algorithm 2 Feature Matrix Generation

Split $\mathcal{M}$ (313 models) into $\mathcal{M}_{train}$ (213 models) and $\mathcal{M}_{test}$ (100 models)
Require:
$\mathcal{S}$: Sampling techniques {Readability, Quality, Clustering, Difficulty}
$\mathcal{C} \leftarrow [5, 10, 20]$: Select Model combinations
$k \leftarrow 100$: Clusters per benchmark
$n_{iter} \leftarrow 1000$: Number of iterations
d_b is the full data of given benchmark $b \forall \in B$;
Ensure: Feature matrix $\mathbf{F}$, training correlations $\{\rho_{train_{m,i}}\}$, subset indices $\mathcal{I}$.
Initialize $\mathbf{F} \leftarrow \emptyset, \quad \mathcal{I} \leftarrow \emptyset$.
Select optimal feature set $\{D, Q, DD\}$ by ensuring $d_b(D, Q, DD) \sim \mathcal{N}(\mu, \sigma^2)$.
for each benchmark $b \in \mathcal{B}$ **do**
  $\mathbf{F}_b \leftarrow \emptyset, \quad \mathbf{F}_{Ss} \leftarrow \emptyset$.
  for each sampling technique $s \in \mathcal{S}$ **do**
    $\mathbf{F}_s \leftarrow \emptyset$.
    for $i = 1$ **to** n_{iter} **do**
      for each $m \in \mathcal{C}$ **do**
        Select unique m models from $\mathcal{M}_{train}$.
        $\mathbf{R}_m \leftarrow$ evaluation results for m models.
      end for
      $Sub_{m,i}, \rho_{train_{m,i}} \leftarrow$ SubMS(m,s,b,$d_{bs,20\%}$)
      Normalize $Sub_{m,i}$ on D^i, Q^i, DD^i
      Store indices $\mathcal{I} \leftarrow \mathcal{I} \cup \{Sub_{m,i}.indices\}$
      Cluster $Sub_{m,i}\{D^*, Q^*, DD^*\}$ into k
      Construct $\mathbf{F}_s i \in m \times k \times 4$ where:
        $\mathbf{s}_i[m, :] = [D^i, Q^i, DD^i, \mathbf{R}_m]$
      $\mathbf{F}_s \leftarrow \mathbf{F}_s \| \mathbf{s}_i, \rho_{train_{m,i}}$
    end for
    $\mathbf{F}_S s \leftarrow \mathbf{F}_s$
  end for
  $\mathbf{F}_b \leftarrow \mathbf{F}_S s$
end for
Aggregate $\mathbf{F} \leftarrow \bigcup_b \mathbf{F}_b$

return $\mathbf{F} \in \mathbf{B} \times S \times n_{iter} \times m \times k \times \{D, Q, DD\ R_m\}, \rho_{train}; \mathcal{I}$

Partial Model Performance: First we split split of LLMs the into 213 models as M_{train} (used in different model combinations in training) and 100 models as M_{test} (reserved for testing) from the 313 LLMs we use in this work.

The test models are used as global evaluation of (Sub_{Win}) selected by the RCP model across diverse set of B Benchmarks. A small combination of models is sampled from this M_{train} to create the Performance Patterns R_m. F includes Q, D, DD data from subset obtained using SubMS with x=20% data, and R_m from a small set (m) of initially evaluated models, where $m \leftarrow C \in$ [5,10,20]. This allows the RCP model to learn the relationship between partial evaluations and data characteristics.The F matrix is generated by sampling subsets using multiple sampling techniques $s \in S$.

We fix the sampling rate x=20% in the RCP model feature generation to perform benchmark agnostic training and evaluation. This is because the size of each benchmark varies hugely as shown in Table 8 in A. In order to train generalizable RCP model, we need feature vectors that are benchmark agnostic. To get this, we sample 20% subset from a benchmark such that it has at least 100 datapoints. These are then clustered into fixed k ($\bar{1}00$) clusters to achieve benchmark agnostic features.

The feature matrices are generated for multiple iterations (n_{iter}) for every s per benchmark b, such that

Algorithm 3 Winner Subset Selection and Leaderboard Evaluation

Require:
$\mathbf{F}$: Feature matrix for training RCP model λ
Feature Matrics containing intrinsic metrics: Difficulty (D), Quality (Q), Distribution Dispersion (DD)
Partial Evaluations R_m of m anchor models' performance patterns, where $m \in C \leftarrow [5, 10, 20]$
$\mathcal{I}$: Subset indices
10 B split with 5-fold CV $B^{(p)}_{train}, B^{(p)}_{val}\}_{p=1}^{5}$
$\mathcal{M}_{test}$: 100 test models
Sampling methods $\mathcal{S}$
Ensure: Winning subset Sub_{Win}, Leaderboard for $\mathcal{M}_{test}$
$Sub^b_{win} \leftarrow \emptyset$
{Phase 1: RCP Model Training}
for each fold $p \in \{1, ..., 5\}$ **do**
  $B_{train} \leftarrow B^{(p)}_{train}(8), B_{val} \leftarrow B^{(p)}_{val}(2)$
  $\mathbf{X}_{train} \leftarrow \mathbf{F}[F_{B_train}]$ {D, Q, DD, Rm Features}
  $\mathbf{y}_{train} \leftarrow \mathbf{F}_{\mathbf{B_{train}}}[\rho_{train}]$ {ρ values}
  Train RCP λ_p: $B^8 \rightarrow B$ with l-layers:
    $h_1 = \sigma(W_1 \mathbf{X}_{\mathbf{D,Q,DD}} + W_2 \mathbf{X}_{\mathbf{Rm}} + b_1)$
    $\vdots$
    $\hat{\rho} = W_l h_{l-1} + b_l$
  Validate on $\mathcal{B}_{val}$
  Compute $Accuracy_{val} \leftarrow 1 - \frac{\|\mathbf{y}_{val} - \lambda_p(\mathbf{X}_{val})\|^2}{\|\mathbf{y}_{val} - \bar{\mathbf{y}}_{val}\|^2}$
end for
{Phase 2: Selecting Winning Subset for every benchmark}

for each benchmark $b \in B$ **do**
  Load Full-Benchmark D, Q, DD Data: d_b
  $\lambda \leftarrow \lambda_p : b \notin B^{(p)}_{train}$ {Make sure b is not in p-CV}
  for each $indices \in \mathcal{I}$ **do**
    $subsets \leftarrow d_b.indices$
  end for
  Compute Sinkhorn distance matrix:
  $D_s =$ Sinkhorn($\mathbf{subsets}, \mathbf{d_b}$)
  $\mathcal{C}_s \leftarrow subsets$[argmin($D_S$)] {Candidates with low distance/S 9 in our experiment}
  Obtain the $\mathbf{F_{b,s}}$, ρ_s of $\mathcal{C}_s$
  $\hat{\rho}_s = \lambda_p(\mathbf{F_{b,s}}, \rho_s)$
  $idx^b_{Win} \leftarrow \max E_s[L_s]$
  $Sub^b_{Win} \leftarrow subsets[idx^b_{Win}]$
  Store the Sub^b_{Win}
end for
{Phase 3: Evaluation on M_{test} with Sub^b_{Win}}
for each benchmark $b \in B$ **do**
  $\rho^b_{M_{test}} \leftarrow$ Spearman($\mathcal{M}_{test}^{Sub^{b\,ranks}_{win}}, \mathcal{M}_{test}^{d_b ranks}$)
  $r^b_{M_{test}} \leftarrow$ Pearson Correlation($\mathcal{M}_{test}^{Sub^{b\,score}_{win}}, \mathcal{M}_{test}^{d_b scores}$)

  Populate $Leaderboard^b$ with Sub^b_{win} results
end for
return $Sub_{win} = \{Sub^b_{win}\}_{b=1}^{10}; Leaderboard^{10}_{b=1}$

we get: $F_b \leftarrow S \times n_{iter} \times m \times k \times D, Q, DD, R_m$ Along with this we also generate the rank correlation ρ of m models used in creating the feature, using Spearman Coefficient. We then aggregate F_b across all B, S, and n_{iter} to form the final feature matrix (F) containing about 9000 candidate subsets for training per benchmark.

4.2.3 Model Formulation & Training

The RCP model (λ) is trained to predict the Spearman rank correlation (ρ) between subset and full-benchmark rankings using a Rank Correlation Prediction model λ. Input to the RCP is Standardized feature matrix combining (i) Intrinsic metrics D, Q, DD, and (ii) Partial evaluations on anchor models'. As a result, the model captures meta-knowledge about how different dataset characteristics affect ranking preservation, enabling accurate predictions on new benchmarks. By training on thousands of candidate subsets across diverse benchmarks, the RCP model internalizes the relationship between these intrinsic characteristics and the resulting rank correlations. This learned mapping enables the model to predict the quality of a candidate subset on unseen benchmarks without the need for the expensive and time-consuming full warm-up phase employed in recent approaches. More information about this is tabulated in Table 4. λ is the N-layer neural network (N=6) with ReLU activations and a final sigmoid layer RCP model, transforming the input F of dimension to a scalar prediction $\hat{\rho}$. The network progressively reduces dimensionality through layers of size 2^{10}, 2^8, 2^7, 2^5, and 1. From the feature matrices, we split them into train and validation set to train our RCP model. In our case, using Weighted Adam optimizer (LR=1e-3) over 150–300 epochs to minimize regression loss on predicted rank correlation. RCP model aims to extrapolate performances of unseen/test models in the leaderboard using RCP selected subset at inference time. This reduces evaluation effort, and helps select a generalizable subset, by training on cross benchmark characteristics. The model minimizes Mean Squared Error(MSE) Loss $\mathcal{L}$ between predicted rank coefficient $\hat{\rho}$ and true rank corefficient ρ using 5-fold cross-validation, ensuring generalization across 8 training benchmarks and 2 unseen test benchmarks. During inference, candidate subsets are filtered via Sinkhorn distance D_s to ensure distributional alignment with the full benchmark on D, Q, and DD metrics. The RCP model(λ) predicts $\hat{\rho}_s$ for each subset, and the winning subset is selected based on the lowest prediction error estimates.

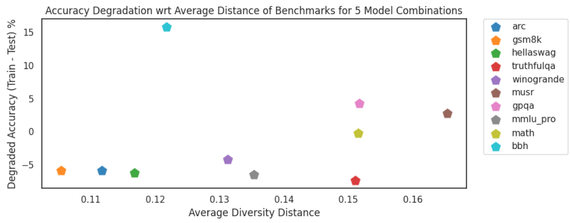


Figure 4: Accuracy Degradation of Benchmark (from Train to Test) for RCP Model Trained with C_5

4.2.4 Inference on New Benchmarks

When introducing a new benchmark, we first compute Q, D, and DD metrics for the full dataset. Next, we generate candidate subsets ($Sub_{m,i}$) from a small set of fully evaluated models using SubMS (e.g., m 5–20) under various sampling strategies. We then identify each candidate subset that minimizes the Sinkhorn distance (D_s) to the reference set, use a trained model to estimate rank preservation for each subset, and select the one with the smallest estimated error. This approach enables the efficient selection of sampling strategies even with limited warm-up data. Typically, it requires only 5 to 20 fully evaluated anchor LLMs to make accurate predictions. As demonstrated in the next section, it achieves stronger rank preservation compared to baselines on most benchmarks.

5 Experiments and Results

In this section, we present results for RCP model train and test, including distance-based filtering of candidate subsets and winner subset (Sub_{Win}) selection via RCP inference, followed by test set evaluation. To validate the effectiveness of SubLIME, we have performed comparisons with TinyBenchmarks and Random baseline.

5.1 RCP Model Training Results

The training accuracy remains high (92.63%–96.50% ref Table 5 for C_{10}), indicating the model effectively learns dataset characteristics (D, Q, DD) for ranking preservation. On test benchmarks, it achieves over 98% rank correlation for most, with GSM8K (99.62%), ARC (99.48%), and MMLU-Pro (99.71%) showing near-perfect rank preservation. We observe performance variation across benchmarks. BBH, despite 92.69% training accuracy, drops to 87.09% on test data, suggesting complexity. GPQA records the lowest accuracy (89.96%), likely due to high distributional dispersion, and GPQA consists of very complex scientific problems, making metric-based selection harder. MUSR (92.62%) also sees a moderate drop. The accuracy degradation aligns with their high diversity distances (Figure 25). The overall performance of benchmarks like **MUSR, Math and GPQA** was relatively lower than others. The degradation observed in certain benchmarks, particularly *GPQA* and *MUSR*, prompted an investigation into the underlying causes. We assessed the average Euclidean distance across Q, D, and DD metric-based distribution distances between all benchmark pairs (Figure 25 in Appendix A). We show the D,Q,DD distribution preservation of the winner subset w.r.t fullset based on Sub_{win} of GSM8K and GPQA in Figure 13 in Appendix A.4. Benchmarks like Math and MUSR exhibited relatively higher diversity distances. Math benchmark's complexity in readability resulted in elevated difficulty and distance metrics, while MUSR data comprised narrative and input text with higher context lengths and readability challenges.

Figure 4 in shows a proportional relationship between the diversity distance of a benchmark and the performance degradation observed from the training set to the test. Overall, the RCP model reliably predicts rank correlation, excelling on structured benchmarks like GSM8K and ARC. However, for very diverse datasets like GPQA and MUSR, more anchor models or refined sampling may enhance performance. RCP model performs better than TinyBenchmark in these complex benchmarks as well (ref Table 5,7), confirming SubLIME's effectiveness in efficient evaluation.

5.2 Test Set Evaluation with Winner Subset

Tables 2 compare the rank correlation and score preservation when evaluating **100 unseen LLMs** on subsets chosen by SubLIME (S_{Win} selected by RCP Model trained on C_5, C_{10}, and C_{20}), TinyBench, and Random-sampling selected through SubLIME RCP denoted as Random (RCP), and absolute randomly selected subset denoted as Random (Absolute). The Random (RCP) subset is selected through is a two stage process - (i) We generate 1000 candidate subsets using random sampling. (ii) These candidates are then filtered using our distance–based metric to retain the top 9 candidates, and finally, the RCP model selects the best subset based on its predicted rank correlation. This was designed to ensure consistency with SubLIME's evaluation framework. Each row-group corresponds to an RCP model trained with different anchor set sizes (5, 10, 20). Across most benchmarks, SubLIME achieves higher rank correlation than TinyBench. For example, in C_5, it attains **97.2%** correlation on Winogrande (vs. 90.6% for TinyBench) and **62.0%** on GPQA (vs. 56.0%), and **74.55%** on MUSR (vs 63.04%). Even where TinyBench performs well, such as Hellaswag, the difference remains small. With more anchor models (C_{10} and C_{20}), the correlation for harder benchmarks like MUSR improves from **74.6%** (C_5) to **82.8%** (C_{10}), while GPQA increases from **62.0%** to **66.9%**.

Random sampling achieves high correlation in some benchmarks like GSM8K and Hellaswag (99.4% and 99.8% with C_5), showing that in large datasets, even naive selection can be effective. However, for tasks with higher linguistic diversity, such as GPQA and MUSR, its correlation drops to **58.2%** and **76.9%** with C_5, lower than both SubLIME and TinyBench. However in C_{10} we can see that Random (RCP), has 62%, performs better than Random (Absolute), has 53%, in GPQA benchmark. A **high variance in random sampling results is observed**, suggests it may not generalize well across different benchmark types. Using more anchor models improves results across harder benchmarks. For instance, MUSR's rank correlation increases from **74.6%** (C_5) to **82.8%** (C_{10}), and GPQA improves from **62.0%**(C_5) to **74.3%** (C_{20}) leading to our study (5.3) on minimum required anchor LLMs for a given benchmark to achieve acceptable rank correlation.

For benchmarks with **high inherent redundancy**, even a random subset tends to perform well. The apparent minor differences between our approach and the random in these cases reflect the dataset characteristics rather than a lack of efficacy of our method. Improvements observed in **less-redundant benchmarks** like GPQA and MUSR. They exhibit low redundancy and high diversity in intrinsic characteristics. In such scenarios, SubLIME's guided subset selection via the RCP model is crucial for achieving representative evaluation subsets & offers advantages over existing methods.

5.3 Model Stability Analysis

We evaluated ranking stability by varying the number of anchor LLMs in our SubMS procedure from 5 to 50, repeating the sampling 40 times per setting with independent seeds. Using the unified D, Q, and DD metrics on a 20% subset, we computed the

RCP Model	Method	Correlation	ARC	GSM8K	Hellaswag	Truthfulqa	Winogrande	MUSR	GPQA	MMLU_Pro	Math	BBH
C_5	**SubLIME**	RC (ρ)	97.80%	99.48%	99.36%	98.42%	97.16%	74.55%	62.00%	99.46%	97.23%	98.72%
		SDC (ρ)	99.08%	99.72%	99.65%	99.37%	97.88%	75.82%	65.22%	99.85%	98.20%	99.32%
	TinyBench (Baseline)	RC	97.57%	99.02%	98.58%	97.88%	90.59%	63.04%	56.03%	96.55%	96.32%	98.66%
		SDC	97.11%	99.46%	97.43%	97.61%	89.57	69.69%	59.94%	96.77%	92.67%	99.10%
	Random (RCP)	RC	97.8%	99.35%	99.75%	98.91%	97.00%	76.96%	58.21%	99.34$	96.95%	99.06%
		SDC	99.08%	99.78%	99.84%	99.6%	97.70%	77.4%	59.5%	99.7%	97.48%	99.48%
C_{10}	**SubLIME**	RC (ρ)	98.95%	99.46%	99.72%	99.6%	96.38%	82.75%	66.91%	99.07%	97.02%	98.31%
		SDC (ρ)	99.43%	99.71%	99.96%	99.38%	99.06%	83.96%	66.63%	99.84%	97.78%	99.02%
	TinyBench (Baseline)	RC	97.44%	98.01%	98.55%	95.98%	93.77%	73.20%	59.80%%	97.05%	96.93%	98.56%
		SDC	97.84%	99.34%	97.00%	97.56%	92.34%	73.37%	61.94%	97.27%	92.32%	99.13%
	Random (RCP)	RC	99.31%	99.38%	98.85%	99.57%	96.38%	72.33%	61.14%	99.54%	97.00%	98.81%
		SDC	99.07%	99.7%	99.98%	99.66%	99.06%	75.01%	66.6%	99.85%	97.5%	99.08%
	Random (Absolute)	RC	98.82%	99.49%	99.6%	98.58%	95.31%	73.73%	52.82%	99.4%	97.31%	98.70%
		SDC	99.53%	99.73%	99.95%	99.11%	99.03%	74.71%	64.51%	99.78%	98.5%	99.3%
C_{20}	**SubLIME**	RC (ρ)	99.01%	99.34%	99.51%	98.95%	97.68%	77.89%	74.35%	99.37%	96.28%	99.03%
		SDC (ρ)	99.52%	99.80%	99.96%	99.46%	99.10%	79.71%	71.39%	99.83%	98.18%	99.24%
	TinyBench (Baseline)	RC	95.85%	98.95%	98.18%	96.25%	94.07%	69.69%	64.33%%	99.06%	95.80%	98.70%
		SDC	97.39%	99.45%	96.76%	97.79%	93.81%	73.05%	73.78%	98.14%	91.59%	98.74%
	Random (RCP)	RC	99.30%	99.68%	99.34%	99.22%	97.63%	76.95%	59.32%	99.16%	96.22%	98.57%
		SDC	99.52%	99.74%	99.95%	99.05%	98.87%	77.57%	57.68%	99.8%	98.15%	99.13%
	Random (Absolute)	RC	98.95%	99.11%	98.85%	99.57%	87.61%	75.28%	64.87%	99.56%	96.52%	98.89%
		SDC	99.43%	99.71%	99.97%	99.48%	98.56%	76.34%	66.94%	99.83%	98.53%	99.27%

Table 2: Evaluation on 100 M_{test} LLMs across Benchmarks for SubLIME vs TinyBench vs Random. SubLIME evaluates Rank Correlation(RC) and Score Distribution Correlation (SDC) on Winner Subset Sub_{Win} (20%)

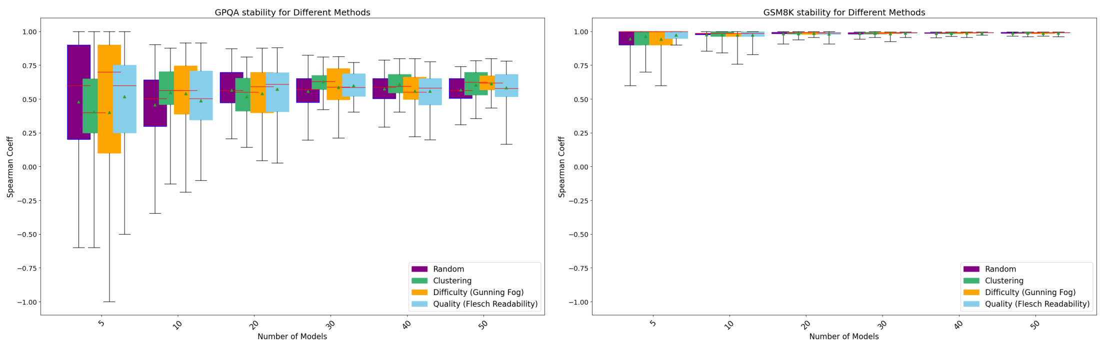


Figure 5: Model Stability Analysis & Variance - GPQA(left) & GSM8K(right)

ρ and variance for each configuration. As shown in Figure 5, *increasing the number of anchor LLMs improves rank consistency and reduce variance*. For challenging benchmarks like GPQA (left), 20 anchor models yield stable correlations, and for structured benchmarks such as GSM8K (right), 5 models are sufficient. These results guide the optimal selection of anchor models for RCP model.

6 Conclusion

We introduced **SubLIME**, a data-efficient LLM evaluation framework that preserves ranking fidelity while reducing computational costs by up to 99%. Inspired by mathematical olympiad problem selection, SubLIME employs a RCP model to guide subset selection using limited anchor LLM evaluations and thre intrinsic dataset metrics. SubLIME achieves over 98% rank correlation with full benchmark rankings across ten diverse LLM benchmarks, outperforming existing baselines like TinyBench. Model stability analysis reveals that as few as 5–20 anchor models suffice for robust rank correlation predictions. While SubLIME excels in structured benchmarks, performance degrades on datasets with extreme distributional dispersion, highlighting future work in uncertainty quantification and hybrid selection strategies. As LLMs and benchmarks grow exponentially, scalable evaluation methodologies like SubLIME are essential for cost-effective benchmarking. By enabling efficient assessments, our approach advances automated evaluation and leaderboard curation.

7 Limitations

While SubLIME demonstrates promising results in data-efficient evaluation of LLMs, several limitations remain that warrant discussion:

Anchor Model Evaluation Overhead: Although our method substantially reduces the evaluation cost for newly added LLMs by requiring full evaluation results for only 5 to 20 anchor models, it does not eliminate the need for complete evaluations of these anchor models. In scenarios where evaluating even a small number of models is expensive, this requirement may still represent a significant computational and time overhead.

Indicator Robustness: Our approach relies on intrinsic dataset metrics—Difficulty (D), Quality (Q), and Distributional Dispersion (DD)—to guide subset selection. However, these indicators do not always capture the nuanced characteristics of every benchmark perfectly. In some cases, more modern or domain-specific indicators might be necessary to fully characterize the benchmark's complexity and improve the rank preservation performance.

Assumptions on Benchmark Representativeness: SubLIME assumes that the anchor models' performance is representative of the overall model population. This assumption may not hold in rapidly evolving leaderboards or in domains where new models exhibit significantly different behavior from those used during the warm-up phase, potentially affecting the generalizability of the selected subset.

Applicability to specialized benchmarks: The applicability of SubLIME to emerging benchmark types and highly specialized domains may face challenges, which is an inspiration to our work in assessing intrinsic metrics that are scalable to any benchmark type and size. By considering a metric generalizable across all benchmarks, such as grade level requirement, or cognitive functionalities to solve a problem, helps us expand it to specialized domains.

Addressing these limitations in future work could further enhance the robustness and applicability of SubLIME across a wider range of benchmarks and evaluation scenarios.

References

Vladimir Bochkarev, Anna Shevlyakova, and Valery Solovyev. 2012. Average word length dynamics as indicator of cultural changes in society. *Social Evolution and History*, 14:153–175.

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

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Tianyu Ding, Tianyi Chen, Haidong Zhu, Jiachen Jiang, Yiqi Zhong, Jinxin Zhou, Guangzhi Wang, Zhihui Zhu, Ilya Zharkov, and Luming Liang. 2023. The efficiency spectrum of large language models: An algorithmic survey. *Preprint*, arXiv:2312.00678.

Chengcheng Guo, Bo Zhao, and Yanbing Bai. 2022. Deepcore: A comprehensive library for coreset selection in deep learning. *arXiv preprint arXiv:2204.08499*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *NeurIPS*.

Wenhao Hu, Dong Xu, and Zhihua Niu. 2021. Improved k-means text clustering algorithm based on bert and density peak. In *2021 2nd Information Communication Technologies Conference (ICTC)*, pages 260–264.

Yifei Hu, Xiaonan Jing, Youlim Ko, and Julia Taylor Rayz. 2024. Misspelling correction with pre-trained contextual language model.

Hugging Face. 2022. Open llm leaderboard. `https://huggingface.co/open-llm-leaderboard`. Retrieved February 3, 2022.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, Benjamin Newman, Binhang Yuan, Bobby Yan, Ce Zhang, Christian Alexander Cosgrove, Christopher D Manning, Christopher Re, Diana Acosta-Navas, Drew Arad Hudson, Eric Zelikman, Esin Durmus, Faisal Ladhak, Frieda Rong, Hongyu Ren, Huaxiu Yao, Jue WANG, Keshav Santhanam, Laurel Orr, Lucia Zheng, Mert Yuksekgonul, Mirac Suzgun, Nathan Kim, Neel Guha, Niladri S. Chatterji, Omar Khattab, Peter Henderson, Qian Huang, Ryan Andrew Chi, Sang Michael Xie, Shibani Santurkar, Surya Ganguli, Tatsunori Hashimoto, Thomas Icard, Tianyi Zhang, Vishrav Chaudhary, William Wang, Xuechen Li, Yifan Mai, Yuhui Zhang, and Yuta Koreeda. 2023. Holistic evaluation of language models. *Transactions on Machine Learning Research*. Featured Certification, Expert Certification.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. Truthfulqa: Measuring how models mimic human falsehoods. *Preprint*, arXiv:2109.07958.

H. P. Luhn. 1958. The automatic creation of literature abstracts. *IBM Journal of Research and Development*, 2(2):159–165.

Max Marion, Ahmet Üstün, Luiza Pozzobon, Alex Wang, Marzieh Fadaee, and Sara Hooker. 2023. When less is more: Investigating data pruning for pretraining llms at scale. *Preprint*, arXiv:2309.04564.

Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. 2022. Mteb: Massive text embedding benchmark. *arXiv preprint arXiv:2210.07316*. Version 3.

Yotam Perlitz, Elron Bandel, Ariel Gera, Ofir Arviv, Liat Ein-Dor, Eyal Shnarch, Noam Slonim, Michal Shmueli-Scheuer, and Leshem Choshen. 2024. Efficient benchmarking of language models. *Preprint*, arXiv:2308.11696.

Felipe Maia Polo, Lucas Weber, Leshem Choshen, Yuekai Sun, Gongjun Xu, and Mikhail Yurochkin. 2024. tinybenchmarks: evaluating llms with fewer examples. *Preprint*, arXiv:2402.14992.

Ameya Prabhu, Vishaal Udandarao, Philip Torr, Matthias Bethge, Adel Bibi, and Samuel Albanie. 2024. Lifelong benchmarks: Efficient model evaluation in an era of rapid progress. *Preprint*, arXiv:2402.19472.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2024. GPQA: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: an adversarial winograd schema challenge at scale. *Commun. ACM*, 64(9):99–106.

Philip Sedgwick. 2014. Spearman's rank correlation coefficient. *Bmj*, 349.

Ben Sorscher, Robert Geirhos, Shashank Shekhar, Surya Ganguli, and Ari S. Morcos. 2023. Beyond neural scaling laws: beating power law scaling via data pruning. *Preprint*, arXiv:2206.14486.

Zayne Sprague, Xi Ye, Kaj Bostrom, Swarat Chaudhuri, and Greg Durrett. 2024. Musr: Testing the limits of chain-of-thought with multistep soft reasoning. *Preprint*, arXiv:2310.16049.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, , and Jason Wei. 2022. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*.

Rajan Vivek, Kawin Ethayarajh, Diyi Yang, and Douwe Kiela. 2024. Anchor points: Benchmarking models with much fewer examples. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1576–1601, St. Julian's, Malta. Association for Computational Linguistics.

Tomasz Walkowiak and Mateusz Gniewkowski. 2019. Evaluation of vector embedding models in clustering of text documents. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, pages 1304–1311, Varna, Bulgaria. INCOMA Ltd.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. 2024. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Preprint*, arXiv:2406.01574.

Sang Michael Xie, Shibani Santurkar, Tengyu Ma, and Percy Liang. 2023. Data selection for language models via importance resampling. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Abdelrahman Zayed, Prasanna Parthasarathi, Gonçalo Mordido, Hamid Palangi, Samira Shabanian, and Sarath Chandar. 2023. Deep learning on a healthy data diet: finding important examples for fairness. In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'23/IAAI'23/EAAI'23. AAAI Press.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? *Preprint*, arXiv:1905.07830.

Fred Zenker and Kristopher Kyle. 2021. Investigating minimum text lengths for lexical diversity indices. *Assessing Writing*, 47:100505.

A Appendix

A.1 Additional Description and Results of SubMS

A.1.1 Sampling Methods in SubMS

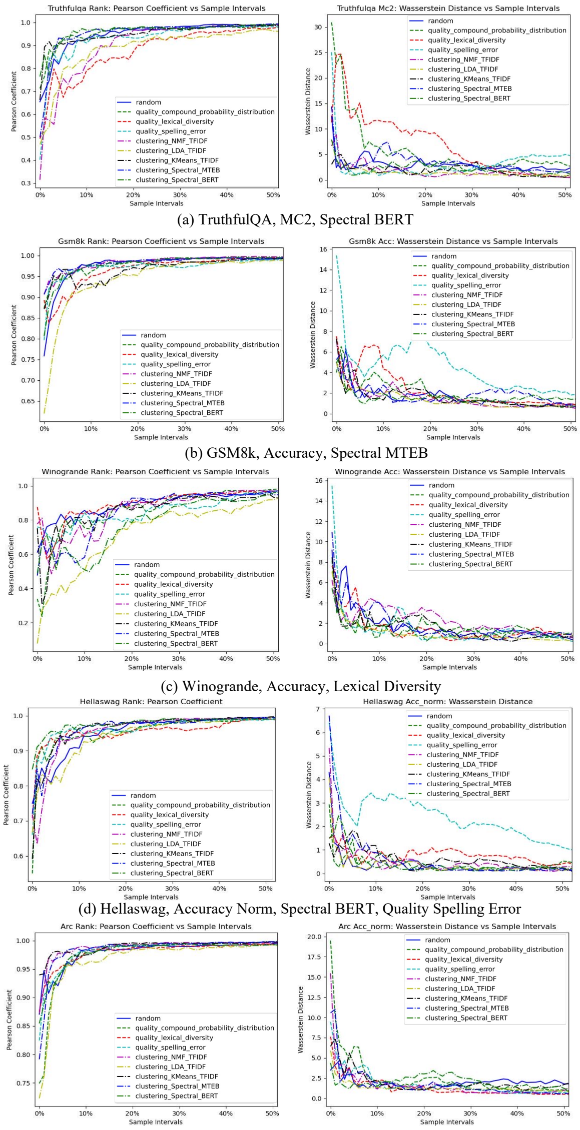
Below are the sampling methods used in SubMS:

Random Sampling serves as the baseline, wherein we select 1% to 100% of the dataset in 1% increments (using fixed random seeds). This straightforward approach offers a simple, unbiased way to compare performance across LLMs and helps calibrate more sophisticated methods.

Clustering-based Sampling including both topic modeling and spectral clustering techniques are explored to ensure diverse coverage of semantic structures. Topic modeling employs Non-Negative Matrix Factorization (NMF) and Latent Dirichlet Allocation (LDA) over TF-IDF representations (Luhn, 1958) to extract topical clusters, from which we stratify samples to capture broad thematic variety. Spectral clustering leverages embeddings such as MTEB (Muennighoff et al., 2022), BERT (Walkowiak and Gniewkowski, 2019), and simple TFIDF embedding based K-means (Hu et al., 2021) to cluster data points in a latent space and sample representatives from each cluster to preserve distributional properties of original benchmark.

Quality-based Sampling A high-quality subset prioritizes instances with clearer and more coherent text, reducing noise and ambiguity. This includes indicators to minimize spelling errors (Hu et al., 2024) for text clarity, maintain an optimal average word length (Bochkarev et al., 2012) to balance complexity with readability, and promote lexical diversity (Zenker and Kyle, 2021) by selecting text segments rich in vocabulary. These methods ensure that only high-quality items are retained, reducing confounding factors and improving the stability of model comparisons.

Difficulty-based Sampling considers input text complexity measures such as *Gunning Fog* and *SMOG*. The *Gunning Fog Index* estimates readability based on sentence length and the proportion of complex words (three or more syllables), indicating how difficult a passage is to comprehend. The *SMOG* grade estimates the education level needed to understand a text. By selecting subsets that span a range of these scores, this approach ensures a balanced distribution of difficulty levels.



(a) TruthfulQA, MC2, Spectral BERT

(b) GSM8k, Accuracy, Spectral MTEB

(c) Winogrande, Accuracy, Lexical Diversity

(d) Hellaswag, Accuracy Norm, Spectral BERT, Quality Spelling Error

(e) ARC, Accuracy Normalized, Clustering NMF, and Spectral MTEB

Figure 6: Rank and Score Preservation of all V1 Benchmarks across multiple sampling techniques

A.1.2 SubMS Results

In this section, we assessed various sampling techniques' effectiveness in reducing the benchmark time while maintaining rankings using a subset of the complete dataset. Using our proposed method outlined in 1, we aim to dynamically pinpoint the best sampling approach for each benchmark.

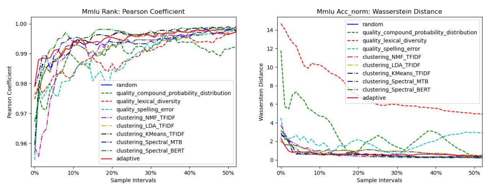


Figure 7: Adaptive Sampling (denoted in Solid Red) achieving stable performance for MMLU Benchmark (V1)

A.2 SubMS: Rank Preservation and Score Distribution Preservation for V1 and V2 Benchmarks

The results for all benchmarks combined is depicted in Figure 6.

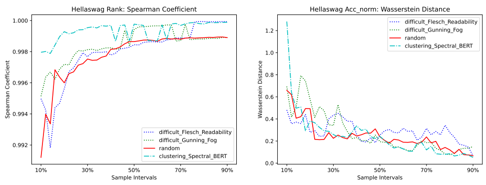


Figure 8: SubMS results on Hellaswag

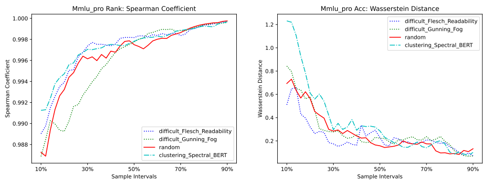


Figure 9: SubMS results on MMLU Pro

A.2.1 Analysis of Rank Preservation and Score Distribution

For SubMS, we evaluated 9 sampling approaches across 50 LLMs on 10 benchmarks listed in A in Table 9. Rank preservation was assessed using the Pearson correlation coefficient, and score distribution discrepancy was measured with the Wasserstein Distance (WD). Figures 7 illustrate these metrics. We selected the best strategies from the results in Quality - Lexical Diversity, Clustering - BERT, and Difficulty - Gunning Fog for our experiments as shown in Figure 8 for Hellaswag and 9 for MMLU Pro.

A.3 SubMS Evaluation with Tiny Benchmarks

We conducted experiments comparing a 20% subset selected by our baseline SubMS (without the Rank Correlation Prediction model) against the TinyBenchmarks baseline. In these experiments, the winner subset was selected from 213 LLMs as the train set, and tested with 100 unseen LLMs to compute the Spearman coefficients for various benchmarks. The results in Table 3 indicate that when sample-level results are available for hundreds of LLMs, TinyBenchmarks generally performs very well—in many cases even outperforming SubMS on several benchmarks (e.g., bbh, gpqa, musr, winogrande). This confirms that TinyBenchmarks leverages extensive evaluation data effectively. However, the key point is that TinyBenchmarks relies on having large-scale sample-level results. In many practical settings, especially during the warm-up phase of a new benchmark, only limited data (typically from 5–20 anchor LLMs) is available. In this low-data regime, our proposed RCP approach (even random baseline) outperforms TinyBenchmarks. Our RCP model is specifically designed to work effectively with limited anchor evaluations, yielding much higher rank correlation and score preservation.

A.4 SubLIME & RCP Model

A.4.1 Describing Benchmark Intrinsic D, Q, DD Metrics

Difficulty D: We use the Gunning Fog Index to quantify textual complexity. In our experiments, this metric provided a well-distributed measure across multiple benchmarks, effectively capturing text complexity (see Table 1). For different benchmark types (multiple-choice vs open-ended), the index is applied directly to the "Question" or "Input Prompt." Once computed, these scores are normalized across benchmarks.

Quality Q: Although Flesch is a traditional readability metric, we interpret it as a proxy for text clarity and coherence—key aspects of quality. In SubLIME, we also considered additional quality metrics (e.g., lexical diversity and duplication rates). We selected the Flesch score because its subset distribution best represented the full benchmark distribution.

Distributional Dispersion DD: The input texts are embedded using a BERT model and clustered to capture data diversity. For each datapoint, we compute the distance from its cluster centroid, which serves as a measure of how well the datapoint represents the overall distribution. As with D and Q, these values are calculated on the "Question" or "Input Prompt" and normalized across benchmarks. These procedures ensure that the RCP model accurately learns the contributions of difficulty, quality, and distributional dispersion to model performance.

Ablation study of D, Q, DD metrics: As suggested the individual contributions of Difficulty (D), Quality (Q), and Distributional Dispersion (DD) is important. We performed metric correlations for GPQA benchmark, shown in Figure 10 which indicates some weak correlations with model performance. The integrated/combined approach of using of all three (D, Q, DD) metrics yields better correlations. Some of the readability not always correlate with model performance, which upon further experimentation we found that, this occurs due to lack of clarity in the benchmark. The weak correlation of Q metric when analysed showed that some question having easier vocabulary was chosen as higher quality but they lacked clarity which might have led to higher model dif-

Method	ARC	BBH	GPQA	GSM8K	HELLASWAG	MATH	MMLU_PRO	MUSR	TRUTHFULQA	WINOGRANDE
SubMS	99.14%	81.64%	31.62%	99.31%	99.18%	96.16%	97.61%	63.39%	98.41%	87.99%
TinyBenchmarks	98.27%	98.11%	43.77%	98.66%	99.76%	98.08%	98.92%	78.51%	96.40%	96.79%

Table 3: Subset Rank correlation of SubMS versus TinyBenchmarks across various benchmarks.

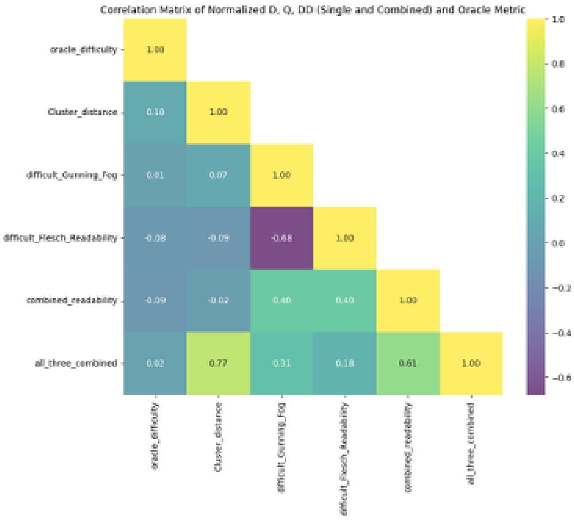


Figure 10: Correlation of D, Q, DD - Invidual and combined metric

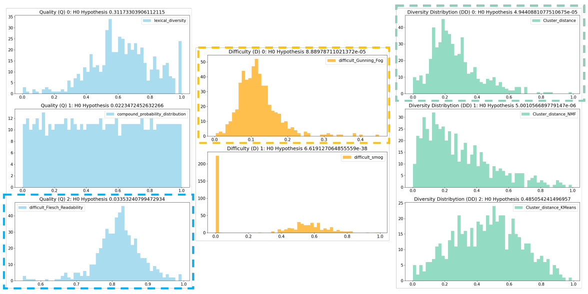


Figure 11: Unified D, Q, DD Selection for GPQA Benchmark based on K-S Test

ficulty. This inspires for additional metrics that could be added to the feature matrix. Which is why we also planning to expand this 3-dimensional intrinsic metrics to a multi-dimensional matrix to provide additional characteristic information indicating the performance, including prompt-based model-based metrics.

A.4.2 RCP model training

Winner subset selection for Oracle-difficulty Metric in 16. These the selected winner subset distributions across different benchmarks, demonstrating preservation of the D, Q, DD metrics with respect to full benchmark data.

Pair-wise distance computed across Benchmark to analyse benchmark diversity:

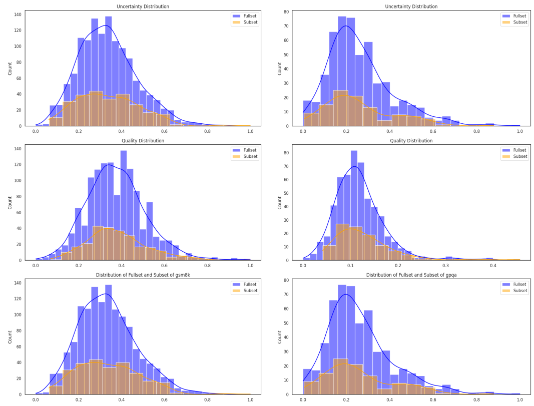


Figure 12: 20% Subset Distribution Preservation on D, Q, DD wrt Fullset - GSM8K (left) and GPQA (right)

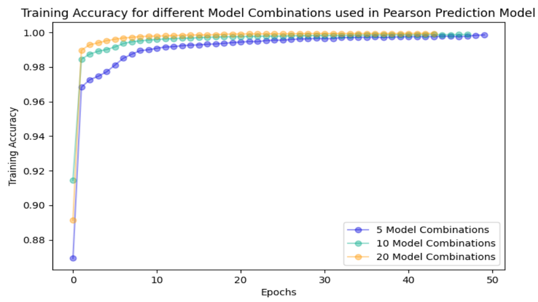


Figure 13: Training Loss for C_5, C_{10}, C_{20} Trained RCP λ Model

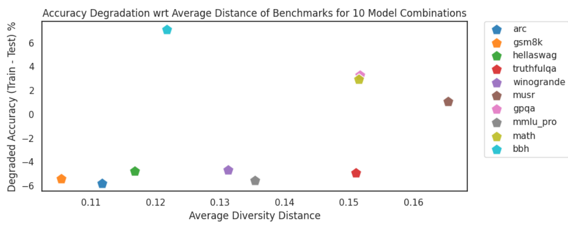


Figure 14: Accuracy Degradation of Benchmark (from Train to Test) for RCP Model Trained with C_{10}

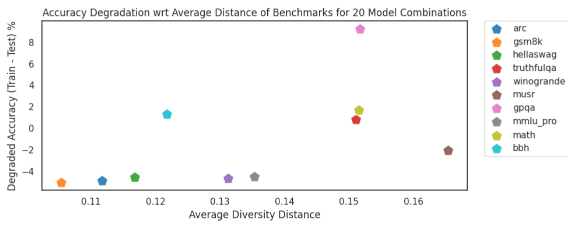


Figure 15: Accuracy Degradation of Benchmark (from Train to Test) for RCP Model Trained with C_{20}

Figure 16: Min vector distance-based D_s filtration of candidate subsets C_s(Red markers) using Oracle-difficulty as "D" - GSM8K(top) and MUSR(bottom) Benchmarks

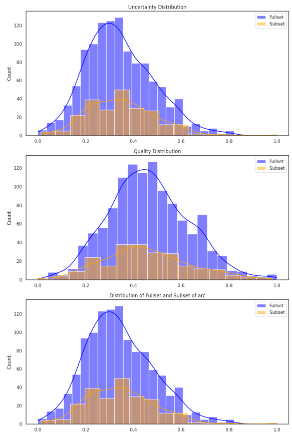


Figure 19: 20% Sub_{Win} D, Q, DD Distribution Preservation wrt Fullset - ARC Benchmark

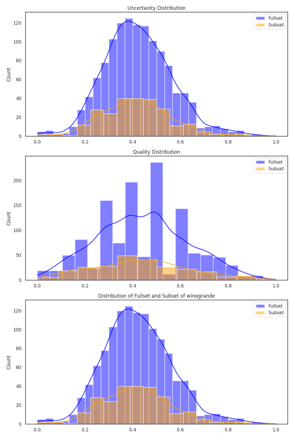


Figure 17: 20% Sub_{Win} D, Q, DD Distribution Preservation wrt Fullset - Winogrande Benchmark

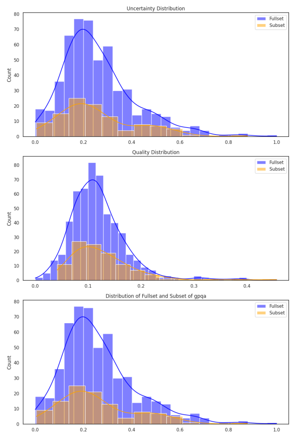


Figure 20: 20% Sub_{Win} D, Q, DD Distribution Preservation wrt Fullset - GPQA Benchmark

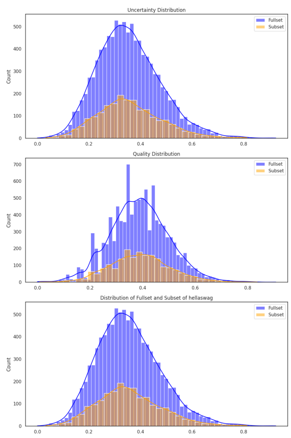


Figure 18: 20% Sub_{Win} D, Q, DD Distribution Preservation wrt Fullset - Hellaswag Benchmark

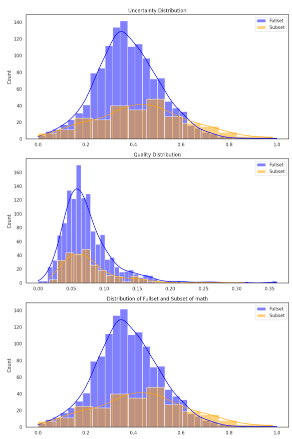


Figure 21: 20% Sub_{Win} D, Q, DD Distribution Preservation wrt Fullset - Math Benchmark

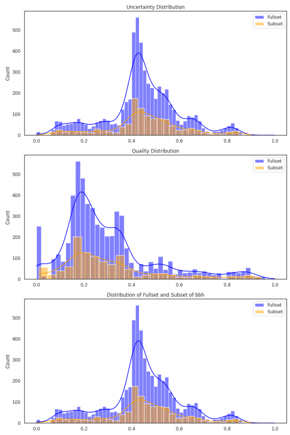


Figure 22: 20% Sub_{Win} D, Q, DD Distribution Preservation wrt Fullset - BBH Benchmark

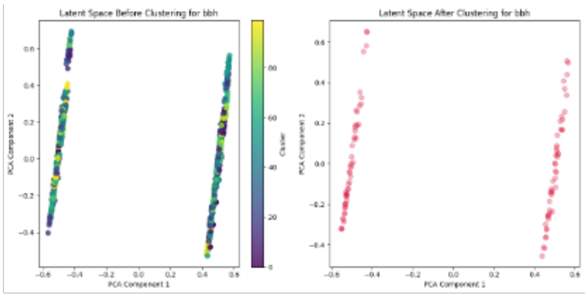


Figure 23: Latent Space Before and After Fixed k Clustering to 100 Clusters for BBH Benchmark

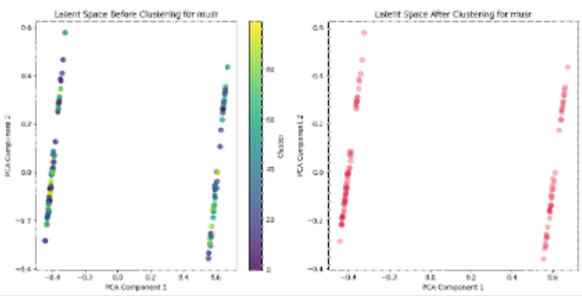


Figure 24: Latent Space Before and After Fixed k Clustering to 100 Clusters for MUSR Benchmark

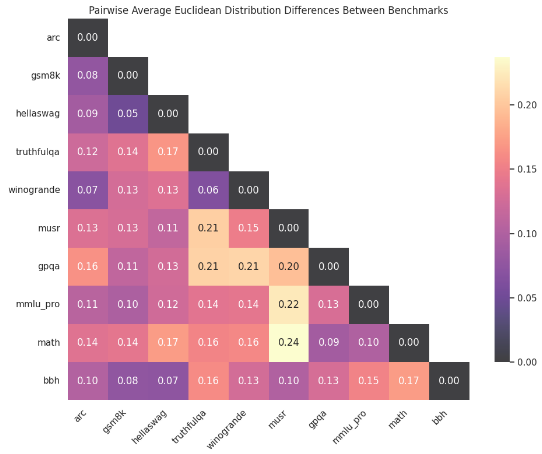


Figure 25: Pair-wise Diversity Distance across Benchmarks

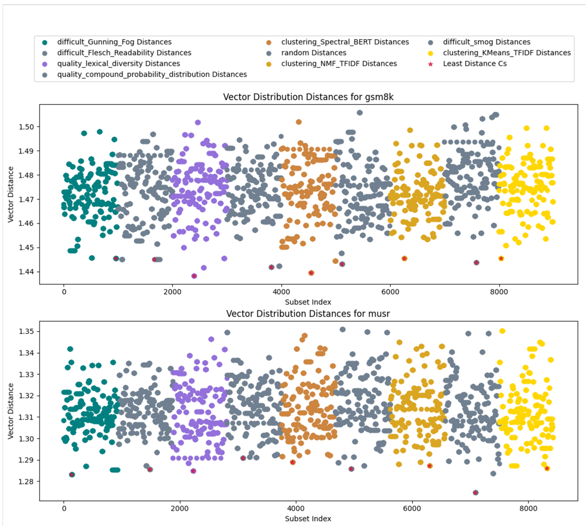


Figure 26: Min vector distance-based D_s filtration of candidate subsets C_s(Red markers) across sampling techniques (9) $s \in S$ - GSM8K(top) and MUSR(bottom) Benchmarks

A.4.3 RCP Model vs Baseline Approaches

A.4.4 SubLIME results

A.5 Model Lists for the experiments

The following models are the overlapping LLMs from the V1 and V2 leaderboard, which were used in our experiments:

A.6 Experiment Setup

LLMs used in Training RCP Model $Mtrain$ - 213 Models:

1. allknowingroger/Neuralmultiverse-7B-slerp
2. microsoft/Phi-3-mini-4k-instruct
3. 01-ai/Yi-1.5-9B-Chat-16K

Table 4: Comparison of SubLIME (RCP) vs. Baseline Methods

Component	SubLIME (RCP)	Baseline Methods
Initialization	5–20 anchor models + D/Q/DD metrics	Requires 100+ model evaluations
Feature Source	Benchmark intrinsic metrics + partial evaluations	Full historical leaderboards
Generalization	Adaptable to new benchmarks and leaderboards	Per-benchmark optimization
Compute	Fewer compute resources as we only train an MLP layer and evaluate partial responses	Comparatively higher as full evaluation and anchor weight training is required

Training		Test		
Benchmarks	**Avg. Train Accuracy**	**Benchmarks**	**SubLIME Test Accuracy**	**Tiny Benchmark Accuracy**
arc, hellaswag, truthfulqa, winogrande, musr, gpqa, mmlu_pro, bbh	92.69%	gsm8k	99.62%	96.81%
		math	91.25%	91.37%
gsm8k, hellaswag, truthfulqa, winogrande, gpqa, mmlu_pro, math, bbh	94.81%	arc	99.48%	95.76%
		musr	92.62%	76.06%
arc, gsm8k, truthfulqa, winogrande, musr, gpqa, math, bbh	92.88%	hellaswag	98.94%	99.39%
		mmlu_pro	99.71%	96.99%
arc, gsm8k, hellaswag, truthfulqa, musr, gpqa, mmlu_pro, math	92.63%	winogrande	98.89%	92.59%
		bbh	87.09%	96.58%
arc, gsm8k, hellaswag, winogrande, musr, mmlu_pro, math, bbh	96.50%	truthfulqa	98.18%	93.55%
		gpqa	89.96%	61.22%

Table 5: Train and Test Accuracy for RCP Model Trained with $M_{train}C_{10}$ Model Combinations

Training		Test		
Benchmarks	**Avg. Train Accuracy**	**Benchmarks**	**SubLIME Test Accuracy**	**Tiny Benchmark Accuracy**
arc, hellaswag, truthfulqa, winogrande, musr, gpqa, mmlu_pro, bbh	91.26%	gsm8k	98.38%	94.67%
		math	92.71%	90.00%
gsm8k, hellaswag, truthfulqa, winogrande, gpqa, mmlu_pro, math, bbh	93.49%	arc	97.84%	93.05%
		musr	89.22%	59.55%
arc, gsm8k, truthfulqa, winogrande, musr, gpqa, math, bbh	91.01%	hellaswag	98.76%	98.85%
		mmlu_pro	99.04%	96.53%
arc, gsm8k, hellaswag, truthfulqa, musr, gpqa, mmlu_pro, math	91.31%	winogrande	96.64%	87.75%
		bbh	76.68%	94.65%
arc, gsm8k, hellaswag, winogrande, musr, mmlu_pro, math, bbh	94.24%	truthfulqa	99.12%	94.45%
		gpqa	87.53%	66.41%

Table 6: Training and Test Benchmark Accuracy for RCP Model Trained with $M_{train}C_5$ Model Combinations

4. SanjiWatsuki/Silicon-Maid-7B
5. shadowml/Mixolar-4x7b

Training		Test		
Benchmarks	**Avg. Train Accuracy**	**Benchmarks**	**SubLIME Test Accuracy**	**Tiny Benchmark Accuracy**
arc, hellaswag, truthfulqa, winogrande, musr, gpqa, mmlu_pro, bbh	93.30%	gsm8k	99.8%	97.66%
		math	93.08%	93.96%
gsm8k, hellaswag, truthfulqa, winogrande, gpqa, mmlu_pro, math, bbh	95.03%	arc	99.21%	94.36%
		musr	96.40%	74.70%
arc, gsm8k, truthfulqa, winogrande, musr, gpqa, math, bbh	93.60%	hellaswag	99.25%	99.51%
		mmlu_pro	99.21%	98.47%
arc, gsm8k, hellaswag, truthfulqa, musr, gpqa, mmlu_pro, math	93.26%	winogrande	99.40%	94.39%
		bbh	93.49%	98.08%
arc, gsm8k, hellaswag, winogrande, musr, mmlu_pro, math, bbh	97.21%	truthfulqa	92.96%	94.46%
		gpqa	84.54%	58.27%

Table 7: Training and Test Benchmark Accuracy for RCP Model Trained with $M_{test}C_{20}$ Model Combinations

6. Ba2han/Llama-Phi-3_DoRA
7. Eric111/CatunaMayo-DPO
8. bigscience/bloom-7b1
9. NousResearch/Hermes-2-Theta-Llama-3-8B
10. EleutherAI/pythia-160m
11. VAGOsolutions/SauerkrautLM-Gemma-2b
12. mlabonne/Daredevil-8B
13. mistralai/Mixtral-8x22B-v0.1
14. vicgalle/Configurable-Yi-1.5-9B-Chat
15. allknowingroger/LimyQstar-7B-slerp
16. fblgit/UNA-ThePitbull-21.4B-v2
17. lmsys/vicuna-7b-v1.3

18. SeaLLMs/SeaLLM-7B-v2.5
19. databricks/dolly-v2-3b
20. OpenBuddy/openbuddy-mixtral-7bx8-v18.1-32k
21. facebook/opt-1.3b
22. NTQAI/Nxcode-CQ-7B-orpo
23. VAGOsolutions/SauerkrautLM-SOLAR-Instruct
24. TencentARC/LLaMA-Pro-8B
25. adamo1139/Yi-34B-200K-AEZAKMI-v2
26. failspy/llama-3-70B-Instruct-abliterated
27. saltlux/luxia-21.4b-alignment-v1.2
28. Felladrin/Minueza-32M-UltraChat
29. uukuguy/speechless-code-mistral-7b-v1.0
30. princeton-nlp/Sheared-LLaMA-2.7B
31. 01-ai/Yi-1.5-6B-Chat
32. cognitivecomputations/dolphin-2.9-llama3-8b
33. Deci/DeciLM-7B
34. 01-ai/Yi-1.5-34B-32K
35. togethercomputer/RedPajama-INCITE-7B-Chat
36. huggyllama/llama-65b
37. automerger/YamshadowExperiment28-7B
38. dreamgen/WizardLM-2-7B
39. Deci/DeciLM-7B-instruct
40. Qwen/Qwen1.5-32B
41. fblgit/UNA-SimpleSmaug-34b-v1beta
42. google/recurrentgemma-2b
43. paloalma/ECE-TW3-JRGL-V1
44. mistralai/Mistral-7B-v0.1
45. Kukedlc/NeuralSynthesis-7B-v0.3
46. gradientai/Llama-3-8B-Instruct-Gradient-1048k
47. google/gemma-1.1-7b-it
48. kevin009/llamaRAGdrama
49. TIGER-Lab/MAmmoTH2-7B-Plus
50. Eric111/CatunaMayo
51. VAGOsolutions/SauerkrautLM-Gemma-7b
52. 0-hero/Matter-0.2-7B-DPO
53. allenai/OLMo-7B-hf
54. EleutherAI/gpt-neo-1.3B
55. mlabonne/NeuralBeagle14-7B
56. EleutherAI/pythia-12b
57. Intel/neural-chat-7b-v3
58. deepseek-ai/deepseek-moe-16b-base
59. Qwen/Qwen1.5-7B
60. euclaise/ReMask-3B
61. SanjiWatsuki/Kunoichi-DPO-v2-7B
62. Kukedlc/NeuralSynthesis-7B-v0.1
63. abhishek/autotrain-llama3-orpo-v2
64. Weyaxi/Einstein-v6.1-Llama3-8B
65. allknowingroger/Strangecoven-7B-slerp
66. 01-ai/Yi-9B-200K
67. allknowingroger/WestlakeMaziyar-7B-slerp
68. HuggingFaceH4/zephyr-7b-gemma-v0.1
69. anakin87/gemma-2b-orpo
70. databricks/dolly-v2-12b
71. bigcode/starcoder2-15b
72. mlabonne/Daredevil-8B-abliterated
73. upstage/SOLAR-10.7B-v1.0
74. pankajmathur/orca_mini_v3_7b
75. CohereForAI/c4ai-command-r-plus
76. argilla/notux-8x7b-v1
77. stabilityai/stablelm-zephyr-3b

78. NousResearch/Nous-Hermes-2-SOLAR-10.7B
79. huggyllama/llama-7b
80. flammenai/Mahou-1.2a-mistral-7B
81. DeepMount00/Llama-3-8b-Ita
82. vicgalle/Configurable-Hermes-2-Pro-Llama-3-8B
83. google/gemma-7b
84. facebook/opt-30b
85. togethercomputer/GPT-NeoXT-Chat-Base-20B
86. iRyanBell/ARC1
87. deepseek-ai/deepseek-llm-7b-chat
88. saishf/Fimbulvetr-Kuro-Lotus-10.7B
89. openchat/openchat_3.5
90. awnr/Mistral-7B-v0.1-signtensors-1-over-2
91. Qwen/Qwen2-0.5B
92. cloudyu/Yi-34Bx2-MoE-60B-DPO
93. jsfs11/MixtureofMerges-MoE-4x7b-v5
94. google/gemma-7b-it
95. meta-llama/Meta-Llama-3-70B
96. 01-ai/Yi-34B-Chat
97. 4season/final_model_test_v2
98. cognitivecomputations/dolphin-2.9.1-yi-1.5-9b
99. microsoft/phi-1_5
100. bigscience/bloom-560m
101. nbeerbower/llama-3-gutenberg-8B
102. Weyaxi/Einstein-v4-7B
103. PygmalionAI/pygmalion-6b
104. togethercomputer/GPT-JT-6B-v1
105. Qwen/Qwen1.5-MoE-A2.7B-Chat
106. 01-ai/Yi-34B
107. 01-ai/Yi-34B-200K
108. EleutherAI/pythia-410m
109. Qwen/Qwen1.5-4B
110. deepseek-ai/deepseek-llm-67b-chat
111. zhengr/MixTAO-7Bx2-MoE-v8.1
112. MaziyarPanahi/Llama-3-8B-Instruct-v0.8
113. tiiuae/falcon-7b-instruct
114. togethercomputer/RedPajama-INCITE-7B-Base
115. BEE-spoke-data/Meta-Llama-3-8Bee
116. CausalLM/34b-beta
117. mistralai/Mixtral-8x7B-Instruct-v0.1
118. argilla/notus-7b-v1
119. stabilityai/stablelm-2-1_6b-chat
120. CohereForAI/c4ai-command-r-v01
121. Weyaxi/SauerkrautLM-UNA-SOLAR-Instruct
122. stabilityai/stablelm-3b-4e1t
123. Intel/neural-chat-7b-v3-1
124. vicgalle/CarbonBeagle-11B-truthy
125. Qwen/Qwen1.5-1.8B-Chat
126. mistral-community/Mistral-7B-v0.2
127. vicgalle/ConfigurableHermes-7B
128. Changgil/K2S3-v0.1
129. AI-Sweden-Models/gpt-sw3-40b
130. awnr/Mistral-7B-v0.1-signtensors-1-over-4
131. shadowml/BeagSake-7B
132. allknowingroger/MultiverseEx26-7B-slerp
133. meta-llama/Llama-2-70b-hf
134. EleutherAI/gpt-neox-20b
135. pankajmathur/orca_mini_v3_13b
136. Sao10K/Fimbulvetr-11B-v2

137. stabilityai/stablelm-2-12b
138. huggyllama/llama-13b
139. tiiuae/falcon-7b
140. Qwen/Qwen1.5-32B-Chat
141. NucleusAI/nucleus-22B-token-500B
142. togethercomputer/Llama-2-7B-32K-Instruct
143. TinyLlama/TinyLlama-1.1B-Chat-v1.0
144. VAGOsolutions/SauerkrautLM-7b-LaserChat
145. Artples/L-MChat-Small
146. Qwen/Qwen2-72B
147. 01-ai/Yi-1.5-34B-Chat
148. Nexusflow/NexusRaven-V2-13B
149. openai-community/gpt2-large
150. MaziyarPanahi/Llama-3-8B-Instruct-v0.9
151. uukuguy/speechless-zephyr-code-functionary-7b
152. pankajmathur/orca_mini_v5_8b_dpo
153. princeton-nlp/Sheared-LLaMA-1.3B
154. 01-ai/Yi-6B
155. teknium/OpenHermes-7B
156. tenyx/Llama3-TenyxChat-70B
157. oobabooga/CodeBooga-34B-v0.1
158. refuelai/Llama-3-Refueled
159. allknowingroger/MultiCalm-7B-slerp
160. lightblue/suzume-llama-3-8B-multilingual
161. openchat/openchat-3.6-8b-20240522
162. Qwen/Qwen1.5-14B
163. abacusai/Llama-3-Smaug-8B
164. ibivibiv/multimaster-7b-v6
165. TencentARC/LLaMA-Pro-8B-Instruct
166. openchat/openchat-3.5-1210
167. Qwen/Qwen1.5-0.5B
168. VAGOsolutions/SauerkrautLM-Mixtral-8x7B-Instruct
169. LLM360/K2
170. fblgit/una-cybertron-7b-v2-bf16
171. vicgalle/CarbonBeagle-11B
172. IDEA-CCNL/Ziya-LLaMA-13B-v1
173. ValiantLabs/Llama3-70B-Fireplace
174. uukuguy/speechless-llama2-hermes-orca-platypus-wizardlm-13b
175. mistralai/Mistral-7B-Instruct-v0.2
176. pankajmathur/orca_mini_v5_8b
177. Artples/L-MChat-7b
178. uukuguy/speechless-mistral-dolphin-orca-platypus-samantha-7b
179. microsoft/Orca-2-7b
180. Qwen/Qwen2-7B
181. allknowingroger/NeuralWestSeverus-7B-slerp
182. Yuma42/KangalKhan-RawRuby-7B
183. 01-ai/Yi-1.5-34B
184. jsfs11/MixtureofMerges-MoE-4x7b-v4
185. mosaicml/mpt-7b
186. 01-ai/Yi-1.5-9B
187. teknium/OpenHermes-2.5-Mistral-7B
188. openchat/openchat-3.5-0106
189. NousResearch/Hermes-2-Pro-Mistral-7B
190. Kquant03/CognitiveFusion2-4x7B-BF16
191. Intel/neural-chat-7b-v3-2
192. spmurrayzzz/Mistral-Syndicate-7B
193. senseable/WestLake-7B-v2
194. yam-peleg/Hebrew-Mistral-7B
195. togethercomputer/RedPajama-INCITE-Instruct-3B-v1

196. google/gemma-2b
197. pankajmathur/orca_mini_v5_8b_orpo
198. Qwen/Qwen1.5-0.5B-Chat
199. Danielbrdz/Barcenas-Llama3-8b-ORPO
200. 01-ai/Yi-1.5-9B-Chat
201. meraGPT/mera-mix-4x7B
202. google/gemma-2b-it
203. togethercomputer/RedPajama-INCITE-Chat-3B-v1
204. mlabonne/AlphaMonarch-7B
205. Open-Orca/Mistral-7B-OpenOrca
206. togethercomputer/RedPajama-INCITE-7B-Instruct
207. ibm/merlinite-7b
208. meta-llama/Meta-Llama-3-8B-Instruct
209. microsoft/DialoGPT-medium
210. EleutherAI/gpt-neo-125m
211. RubielLabarta/LogoS-7Bx2-MoE-13B-v0.2
212. NousResearch/Nous-Hermes-2-Mistral-7B-DPO
213. bigcode/starcoder2-7b

The Original Benchmark size, and Subset size we used in RCP Model Training:

Benchmarks	Fullset Size	20% Subset Size
MUSR	756	151
GPQA	564	112
MMLU_PRO	12032	2406
MATH	1324	264
BBH	5761	1152
arc	1172	234
gsm8k	1319	263
hellaswag	10042	2008
truthfulqa	817	163
winogrande	1267	253

Table 8: Benchmark Sizes with Fullset and 20% Subset

LLMs used in Testing $Mtest$ evaluated on Sub_{Win} subset from RCP Model Inference - 100 Models:

1. meta-llama/Meta-Llama-3-70B-Instruct
2. Qwen/Qwen1.5-110B
3. microsoft/Phi-3-medium-4k-instruct
4. abacusai/Smaug-72B-v0.1
5. MTSAIR/MultiVerse_70B
6. davidkim205/Rhea-72b-v0.5
7. altomek/YiSM-34B-0rn
8. Kukedlc/NeuralLLaMa-3-8b-ORPO-v0.3
9. abacusai/Smaug-34B-v0.1
10. failspy/Phi-3-medium-4k-instruct-abliterated-v3
11. NousResearch/Nous-Hermes-2-Mixtral-8x7B-DPO
12. MaziyarPanahi/Calme-4x7B-MoE-v0.1
13. Kukedlc/NeuralSynthesis-7b-v0.4-slerp
14. MaziyarPanahi/Calme-4x7B-MoE-v0.2
15. dzakwan/dzakwan-MoE-4x7b-Beta
16. mlabonne/Beyonder-4x7B-v3
17. allknowingroger/limyClown-7B-slerp
18. allknowingroger/Neuralcoven-7B-slerp
19. rwitz/go-bruins-v2
20. microsoft/Phi-3-mini-128k-instruct
21. allknowingroger/ROGERphi-7B-slerp
22. 01-ai/Yi-1.5-34B-Chat-16K
23. NousResearch/Hermes-2-Pro-Llama-3-8B
24. 152334H/miqu-1-70b-sf
25. fblgit/UNA-TheBeagle-7b-v1
26. beowolx/CodeNinja-1.0-OpenChat-7B
27. Kukedlc/NeuralExperiment-7b-MagicCoder-v7.5
28. Azure99/blossom-v5.1-34b
29. jeonsworld/CarbonVillain-en-10.7B-v4

30. invalid-coder/Sakura-SOLAR-Instruct-CarbonVillain-en-10.7B-v2-slerp
31. AbacusResearch/Jallabi-34B
32. kekmodel/StopCarbon-10.7B-v5
33. VAGOsolutions/Llama-3-SauerkrautLM-8b-Instruct
34. upstage/SOLAR-10.7B-Instruct-v1.0
35. nlpguy/StarFusion-alpha1
36. vicgalle/ConfigurableBeagle-11B
37. vicgalle/ConfigurableSOLAR-10.7B
38. berkeley-nest/Starling-LM-7B-alpha
39. saltlux/luxia-21.4b-alignment-v1.0
40. ycros/BagelMIsteryTour-v2-8x7B
41. OpenBuddy/openbuddy-llama3-8b-v21.1-8k
42. Intel/neural-chat-7b-v3-3
43. stabilityai/stablelm-2-12b-chat
44. mistralai/Mixtral-8x7B-v0.1
45. rhysjones/phi-2-orange-v2
46. uukuguy/speechless-instruct-mistral-7b-v0.2
47. Qwen/Qwen2-1.5B
48. microsoft/phi-2
49. allknowingroger/Meme-7B-slerp
50. tiiuae/falcon-11B
51. Gryphe/Pantheon-RP-1.0-8b-Llama-3
52. 01-ai/Yi-1.5-6B
53. 01-ai/Yi-9B
54. Josephgflowers/Cinder-Phi-2-V1-F16-gguf
55. mlabonne/OrpoLlama-3-8B
56. meta-llama/Meta-Llama-3-8B
57. CohereForAI/aya-23-8B
58. failspy/Llama-3-8B-Instruct-abliterated
59. microsoft/Orca-2-13b
60. teknium/CollectiveCognition-v1.1-Mistral-7B
61. stabilityai/stablelm-2-zephyr-1_6b
62. mistralai/Mistral-7B-v0.3
63. TencentARC/Mistral_Pro_8B_v0.1
64. Qwen/Qwen1.5-1.8B
65. LeroyDyer/Mixtral_AI_SwahiliTron_7b
66. Qwen/Qwen1.5-14B-Chat
67. Qwen/Qwen1.5-110B-Chat
68. HuggingFaceH4/zephyr-7b-beta
69. meta-llama/Llama-2-13b-hf
70. tiiuae/falcon-40b
71. Josephgflowers/TinyLlama-Cinder-Agent-v1
72. bigcode/starcoder2-3b
73. uukuguy/speechless-coder-ds-6.7b
74. stabilityai/stablelm-2-1_6b
75. Qwen/Qwen1.5-MoE-A2.7B
76. jebcarter/psyonic-cetacean-20B
77. mistralai/Mistral-7B-Instruct-v0.1
78. openchat/openchat_v3.2
79. Qwen/Qwen1.5-7B-Chat
80. openchat/openchat_v3.2_super
81. teknium/OpenHermes-13B
82. google/recurrentgemma-2b-it
83. pszemraj/Mistral-v0.3-6B
84. lmsys/vicuna-7b-v1.5
85. NousResearch/Nous-Hermes-llama-2-7b
86. togethercomputer/LLaMA-2-7B-32K
87. EleutherAI/gpt-j-6b
88. Qwen/Qwen1.5-4B-Chat
89. allenai/OLMo-1B-hf
90. EleutherAI/gpt-neo-2.7B

91. togethercomputer/RedPajama-INCITE-Base-3B-v1
92. databricks/dolly-v2-7b
93. BEE-spoke-data/smol_llama-101M-GQA
94. BEE-spoke-data/smol_llama-220M-GQA
95. openai-community/gpt2
96. OpenAssistant/oasst-sft-1-pythia-12b
97. bigscience/bloom-1b1
98. 01-ai/Yi-1.5-9B-32K
99. Felladrin/Llama-160M-Chat-v1
100. Yash21/TinyYi-7B-Test

Overlapping LLMs from V1 and V2 Leader-board	**Benchmarks from V1 and V2 Leaderboard**
tenyx/Llama3-TenyxChat-70B	arc
mistralai/Mixtral-8x22B-Instruct-v0.1	gsm8k
meta-llama/Meta-Llama-3-70B	hellaswag
SeaLLMs/SeaLLM-7B-v2.5	truthfulqa
mlabonne/Daredevil-8B-abliterated	winogrande
fblgit/UNA-SimpleSmaug-34b-v1beta	musr
openchat/openchat-3.6-8b-20240522	gpqa
jsfs11/MixtureofMerges-MoE-4x7b-v4	mmlu_pro
Kukedlc/NeuralSynthesis-7b-v0.4-slerp	math
dzakwan/dzakwan-MoE-4x7b-Beta	bbh
vicgalle/Configurable-Yi-1.5-9B-Chat	
Eric111/CatunaMayo-DPO	
allknowingroger/Neuralmultiverse-7B-slerp	
Artples/L-MChat-7b	
meta-llama/Meta-Llama-3-8B-Instruct	
Ba2han/Llama-Phi-3_DoRA	
Qwen/Qwen1.5-14B	
4season/final_model_test_v2	
mlabonne/AlphaMonarch-7B	
Weyaxi/Einstein-v6.1-Llama3-8B	
openchat/openchat-3.5-1210	
jeonsworld/CarbonVillain-en-10.7B-v4	
kekmodel/StopCarbon-10.7B-v5	
cognitivecomputations/dolphin-2.9-llama3-8b	
vicgalle/ConfigurableBeagle-11B	
vicgalle/ConfigurableSOLAR-10.7B	
Yuma42/KangalKhan-RawRuby-7B	
lightblue/suzume-llama-3-8B-multilingual	
Intel/neural-chat-7b-v3-3	
NousResearch/Nous-Hermes-2-Mistral-7B-DPO	
CausalLM/34b-beta	
0-hero/Matter-0.2-7B-DPO	
abacusai/Llama-3-Smaug-8B	
microsoft/phi-2	
google/gemma-7b	
01-ai/Yi-1.5-6B	
Deci/DeciLM-7B	
Deci/DeciLM-7B-instruct	
spmurrayzzz/Mistral-Syndicate-7B	
google/gemma-1.1-7b-it	
BEE-spoke-data/Meta-Llama-3-8Bee	
oobabooga/CodeBooga-34B-v0.1	
stabilityai/stablelm-2-zephyr-1_6b	
TencentARC/Mistral_Pro_8B_v0.1	
01-ai/Yi-34B-Chat	
google/gemma-7b-it	
openchat/openchat_3.5	
meta-llama/Llama-2-13b-hf	
Intel/neural-chat-7b-v3-1	
TencentARC/LLaMA-Pro-8B	

Table 9: Overlapping LLMs and Benchmarks from V1 and V2 Leaderboards Used in Testing M_{test} (100 Models)