P3: A Policy-Driven, Pace-Adaptive, and Diversity-Promoted Framework for data pruning in LLM Training

Anonymous ACL submission

Abstract

In the rapidly advancing field of Large Language Models (LLMs), effectively leveraging existing datasets during fine-tuning to maximize the model's potential is of paramount importance. This paper introduces P3, an adaptive framework aimed at optimizing the task-specific fine-tuning process through iterative data pruning. **P3** consists of three key components: (1) **P**olicy-driven Difficulty Measurement, which dynamically assesses data difficulty based on the model's real-time performance, replacing static metrics with adaptable evaluations; (2) **P**ace-Adaptive Selection, leveraging self-paced learning to progressively introduce more challenging data, thereby enhancing model capability; (3) Diversity **P**romotion, incorporating Determinantal Point Process (DPP) to ensure data diversity across epochs, enriching the learning process. We validate **P3** on the reasoning scenarios, APPS and MATH, demonstrating significant improvements over traditional data pruning methods. By advancing dynamic data selection and utilization strategies, **P3** contributes both a theoretical framework and concrete approach to fully exploit existing data for LLMs' performance improvement, offering utility across diverse tasks.

1 Introduction

As Large Language Models (LLMs) continue to evolve rapidly, their ability to handle specialized tasks has become a key focus of research in both academia and industry."(Brown et al., 2020; Mishra et al., 2022; Touvron et al., 2023; OpenAI et al., 2024). This trend underscores the growing importance of LLMs in advancing various technological applications (Naveed et al., 2024; AI4Science and Quantum, 2023; Zhou et al., 2024b). Deploying LLMs within specific domains normally requires a process called fine-tuning, which tailors the model's capabilities to meet the particular needs of different applications. Fine-tuning typically involves adjusting the models using training data that closely reflects specific (application) scenarios (Dong et al., 2023; Taori et al., 2023).

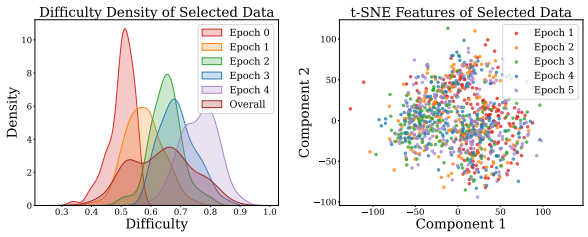

Figure 1: Training Data Analysis Across Epochs. The left graph shows the difficulty distribution of data selected in different epochs, illustrating our progressive training strategy from easier to more challenging tasks. The right graph, using t-SNE visualization, displays the uniform and diverse distribution of selected data.

In the supervised fine-tuning (SFT) of large language models (LLMs), selecting data before or during training is crucial. Research (Albalak et al., 2024; Touvron et al., 2023; Zhou et al., 2024a; Li et al., 2024b; Hase et al., 2024; Li et al., 2024a) shows that LLMs can perform as well as, or even better than, models trained on large datasets by using fewer but higher-quality data points. Both the LLaMA2 and LIMA studies emphasize that "**Quality Is All You Need**" and that "**Less is more for alignment**", demonstrating that a smaller set of well-curated data can yield significant results in SFT of LLMs (Touvron et al., 2023; Zhou et al., 2024a). These findings consistently highlight that the success of fine-tuning LLMs heavily relies on data quality.

Current data pruning methods, including manually defined rules and scoring models to define or evaluate the quality (Li et al., 2024b; Hase et al., 2024; Li et al., 2024a), have notable limitations: (1) They rely on static datasets, failing to adapt to the evolving requirements at different training stages, which often leads to overfitting; (2) Manually defined heuristics and scoring models are not universally applicable, resulting in inaccurate

data valuation when applied across different models or datasets. Furthermore, current data selection methods are usually predetermined before training, however evidence (Xue et al., 2023) suggests that LLMs tend to overfit when trained on static, uniform datasets.

Therefore, our goal is to develop a dynamic and iterative data selection framework during training for LLM that adapts to the changing requirements of each training phase. Previous research (Swayamdipta et al., 2020; Azeemi et al., 2023) has shown that training with appropriately challenging data can improve model performance, highlighting the need for adaptive difficulty. Leveraging self-paced learning, which gradually increases data complexity, allows us to optimize the learning process efficiently, as demonstrated in other domains like semi-supervised learning and noisy label handling. Additionally, maintaining data diversity is essential to promote generalization and reduce overfitting throughout training (Albalak et al., 2024).

Motivated by these insights, we propose a novel adaptive data pruning framework, **P3**, to enhance LLM fine-tuning. As shown in Figure 1, our approach dynamically subsets of training data tailored to different stages of the model's lifecycle, emphasizing higher precision on the same dataset. Specifically, **P3** integrates policy-driven difficulty assessment, self-paced learning (SPL), and Determinantal Point Process (DPP) to progressively fine-tune the model, moving from easier to more challenging tasks while maintaining data diversity. Our key contributions include:

- **Policy-Driven Difficulty Measurement**: A policy-based data evaluation method that directly uses model-derived policies to assess data utility, providing a more accurate and objective alignment with the model's real-time performance.
- **Pace-Adaptive selection for Training**: A stage-wise training strategy using self-paced learning to improve precision with fewer data points. This represents one of the first uses of self-paced learning to optimize LLM fine-tuning for task-specific objectives.
- **Diversity-Promoted Selection via DPP**: An innovative integration of DPP with self-paced learning to balance training data **difficulty** and **diversity**, achieving competitive results in reasoning tasks for LLMs.

Our experiments on the APPS and MATH datasets show that **P3** significantly outperforms traditional methods, demonstrating the effectiveness of our dynamic training strategies in enhancing model performance.

2 Related Works

2.1 Data Pruning

Data pruning, or data selection, has been widely studied to enhance model performance while reducing training dataset size. The goal is to select a subset of data that enables the model to achieve optimal performance (Maharana et al., 2023; Mindermann et al., 2022). Numerous coreset selection techniques have been developed to score data points, indicating their quality for training, with higher scores suggesting greater relevance. However, there is still limited research focused specifically on the application of these methods in the domain of LLMs (Albalak et al., 2024; Li et al., 2024b; Hase et al., 2024). These scoring methods can be categorized into three types: scores based on human expert standards, model-based scores, and scores computed using neural network calculations (Albalak et al., 2024).

Our approach differs from previous work in several key aspects. First, we contend that relying on a fixed standard or another model to prioritize data does not ensure that the selected data will be advantageous for training. Additionally, the difficulty measured by one model might not match the difficulty faced by the training model. Furthermore, we have observed that the difficulty of the same data may decrease as training progresses. This indicates that the data for the training model can vary at different stages, rendering scores assigned prior to training less reliable over time. To tackle these issues, we evaluate the model's actual problem-solving capabilities based on the policies generated during various training stages. A lower success rate at a specific stage suggests that the data presents greater challenges for the model at that point.

2.2 Adaptive Training Strategy

Self-paced learning (SPL) (Kumar et al., 2010; Jiang et al., 2015, 2014; Kong et al., 2021) is well-suited for designing training processes using data of varying difficulties, following a path from simple to complex tasks. SPL allows models to select tasks based on their current learning state, promoting self-regulation. Our research emphasizes

dynamic training, using self-paced learning to tailor our approach. Our framework prioritizes the model's autonomy and adaptability, allowing it to select learning content aligned with its current capabilities, thereby facilitating an adaptive learning pace. We also develop a pioneering DPP-based SPL framework that strategically balances the **difficulty** of tasks with the **diversity** of training data. We believe applying SPL to optimize task-specific model fine-tuning is a pioneering effort in this field.

3 Preliminaries

3.1 Problem Formulation

Consider a domain-specific dataset D consisting of N samples, denoted as $\{(x_i, y_i)\}_{i=1}^{N}$. We utilize a pre-trained model, parameterized by θ, which will be trained across E epochs on the dataset.

The primary objective is to iteratively minimize the expected loss on D by dynamically fine-tuning the pre-trained model θ over the training epochs. This process involves strategically selecting an optimal subset D_e of the training data at the epoch e, determined by the pruning rate α_e:

$$\min_{D_e \subseteq D: \frac{|D_e|}{|D|} \leq \alpha_e} \mathbb{E}_{x,y}\left[L(x, y; \theta_e(D_{e-1}))\right]. \quad (1)$$

Here, L represents the loss function evaluating the model's performance on D_e.

4 Methodology

As shown in Figure 2, our **P3** framework consists of policy-driven difficulty assessment, pace-adaptive training via self-paced learning (SPL), and diversity promotion using Determinantal Point Process (DPP). **Section 4.1** details the difficulty assessment, while **Sections 4.2** and **4.3** outline SPL and DPP. **Section 4.4** describes their integration to optimize data pruning, enhance learning efficiency, and improve generalization through balanced quality and diversity.

4.1 Policy-Driven Difficulty Measurement

This section introduces a policy-driven difficulty metric that evaluates task complexity by analyzing the decisions made by a LLM during execution.

In this work, we use "policy" in an RL-inspired manner, though we do not implement reinforcement learning. Here, "policy" refers to the model's implicit decision-making during training, specifically how it assigns probabilities to different generated sequences, which reflects its assessment of task complexity. Unlike perplexity, this policy-driven difficulty measure captures the model's evolving feedback on task complexity, helping to guide data selection effectively.

To assess the challenge of generating outputs for an instruction x_i, we first segment the corresponding output y_i into discrete sequences a_i:

$$\{a_1, a_2, \ldots, a_n\} = \text{split}(y_i). \quad (2)$$

This segmentation step allows us to analyze complexity at a finer level, identifying which parts of the output contribute most to the task's overall difficulty. It is applicable to various tasks, such as dividing code into lines or breaking down mathematical solutions step-by-step. In essence, generating text can also be viewed as a step-by-step process, making it natural to treat each segment as an action taken by the model.

Next, we compute the generation probability $P(t_i)$ for each sequence using the model's softmax output. For each sequence, we extract the logits corresponding to the label tokens after the prompt length. The generation probability is defined as:

$$P(t_i) = \frac{\exp(\text{logit}_{t_i})}{\sum_j \exp(\text{logit}_{t_j})},$$

where logit_{t_i} represents the logits for the specific token t_i, and the denominator sums the exponentials of all logits for the tokens in the vocabulary. To ensure comparability across different sequences, these probabilities are normalized.

The task difficulty for each sample i at the current epoch e is defined as the complement of the average generation probabilities:

$$d_e(x_i, y_i, \theta) = 1 - \frac{1}{n}\sum_{i=1}^{n} P(a_i), \quad (3)$$

where n is the number of sequences. A higher difficulty score indicates a more challenging task for the model.

As the model trains, the difficulty score may change, making initially hard tasks easier over time. This flexibility enables dynamic training data allocation based on the model's current capabilities.

4.2 Pace-Adaptive Training Methodology

In this section, we introduce a pace-adaptive data pruning method during training aimed at optimizing the training process for LLMs. We implement self-paced learning (SPL) to select training samples that align with the model's capabilities, ensuring

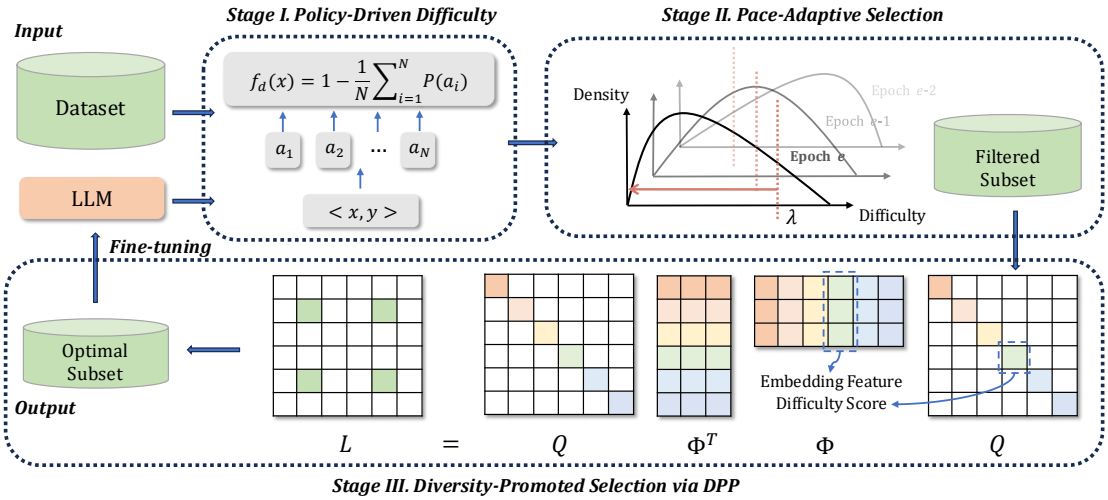

Figure 2: Illustration of the P3 Framework, which embodies a Policy-driven, Pace-adaptive, and Diversity-Promoted approach to optimize Large Language Model training through three strategic stages.

it encounters suitable challenges at each stage to maximize learning efficiency.

We train the model on a domain-specific dataset $D = \{(x_i, y_i), i \in [1, N]\}$. During training, we assign a weight ν to each data example, where $\nu = [\nu_1, \nu_2, \ldots, \nu_N] \in [0, 1]^N$. This weight is adjusted according to the difficulty of the samples, which we quantify using a function $d_e(x_i, y_i, \theta)$ that measures the prediction challenge for each data point.

As training progresses, we iteratively update the difficulty assessment for each example based on the model's policy. This assessment accounts for the model's learning progress, ensuring that the selected training samples reflect the model's current ability. Specifically, we define the adjusted difficulty as:

$$d_{\text{adj},e}(x_i, y_i, \theta) = d_e(x_i, y_i, \theta) + r(\theta) \quad (4)$$

where $r(\theta)$ is a regularization term designed to prevent overfitting:

$$r(\theta) = \alpha \left(f_d(x_i, y_i, \theta_{e-1}) - f_d(x_i, y_i, \theta) \right). \quad (5)$$

In each training epoch, we employ a self-paced function $f(\lambda, \nu_i) = -\lambda \nu_i$ to dynamically focus training efforts. This function determines which samples should be emphasized based on the current difficulty and training progress. We introduce a threshold parameter λ to control the rate at which new samples are included.

$$\nu_i^*(\lambda, l) = \begin{cases} 1, & \text{if } d_e(x_i, y_i, \theta) < \lambda \\ 0, & \text{otherwise} \end{cases}. \quad (6)$$

As training progresses, we define λ as a linear function of the current epoch, specifically a linear function with a slope of 1, increasing from a lower percentile (e.g., 50%) to a higher percentile (e.g., 95%). This approach gradually introducing more challenging samples to encourage continual improvement in the model.

4.3 Diversity-Promoted Selection via DPP

To ensure that our selected training samples are diverse, we use Determinantal Point Processes (DPPs) (Kulesza, 2012). DPP helps us select subsets that are both diverse and representative, which is crucial for training an effective model.

Kernel Matrix for DPP A core part of DPP is the kernel matrix L, which integrates both the quality and similarity of samples.

We start by defining a similarity matrix S that captures how similar each sample is to the others:

$$S_{ij} = \frac{\phi_i^\top \phi_j}{\|\phi_i\| \|\phi_j\|}, \quad (7)$$

where ϕ_i represents the feature vector for sample i. To also consider sample quality, we construct the kernel matrix L as:

$$L_{ij} = q_i S_{ij} q_j, \quad (8)$$

where q_i is the quality score of sample i, which we define as the difficulty score $d_e(x_i, y_i, \theta)$ from Section 4.1. The feature vector ϕ_i is the embedding produced by the model. This kernel matrix combines both the quality and diversity of the samples to guide the selection.

Subset Selection for DPP Given the kernel matrix L, the probability of selecting a subset Y of samples is proportional to the determinant of the corresponding submatrix L_Y:

$$P_L(Y) \propto \det(L_Y). \tag{9}$$

The determinant measures how diverse the selected samples are—higher determinants indicate more diverse selections.

To efficiently select a subset, we use a greedy algorithm to approximate the optimal selection by iteratively choosing samples that maximize the determinant (Chen et al., 2018).

4.4 P3 Architecture

The P3 framework integrates Self-Paced Learning (SPL) with Determinantal Point Processes (DPP) to dynamically adjust data pruning during training process. It increases the difficulty of training samples based on the model's learning capacity while maintaining diversity, which helps prevent overfitting and improve generalization.

In this stage-wise data selection approach, described in Algorithm 1, SPL selects samples that match the model's current capability, while DPP ensures the selected samples are diverse. This combined strategy optimizes the training data selection, enabling the model to gradually learn more challenging tasks effectively.

Algorithm 1 P3 Architecture

Require: training dataset D, model θ, number of epochs E, number of selected samples k, adjustment factor α, self-paced factor λ

```
function DynamicSelect(D, θ, e, k, α, λ)
    d_e ← ComputeDifficulties(θ, D, e)
    S_filtered ← {d ∈ D | d_adj,e ≤ λ(e)}
    K ← ComputeKernelMatrix(S_filtered)
    indices ← DPP(K, k)
    S_e ← {S_filtered[i] | i ∈ indices}
    return S
end function
for epoch e = 1 to E do
    S_e ← DynamicSelect(D, θ, e, k, α, λ)
    θ, loss ← TrainModel(θ, S_e)
end for
```

The P3 architecture effectively balances sample difficulty and diversity throughout training. Self-paced learning adapts the difficulty threshold λ over epochs, allowing the model to gradually take on more challenging data as it learns. Meanwhile, DPP ensures that the training data remains diverse, which improves generalization and helps the model avoid overfitting.

The stage-wise training mechanism follows a straightforward flow: in each epoch, the most appropriate samples are selected based on their difficulty, and DPP is used to promote diversity among the selected samples. This combined approach results in an efficient and adaptive training process, enabling the model to better handle complex tasks over time.

5 Experiments

In this section, we conduct extensive experiments to address the following research questions:

RQ1 How does our framework compare to other methods?

RQ2 How does P3 perform compared to training with the full dataset across different tasks and models?

RQ3 How does policy-based difficulty measurement compare to other metrics, such as model-based and manual assessments?

RQ4 How effectively does DPP contribute to the performance when integrated with SPL?

RQ5 How do the hyperparameters of P3 influence overall model performance?

5.1 Experimental Setups

5.1.1 Datasets

We validate our approach on two well-established LLM datasets: APPS (Hendrycks et al., 2021a) and MATH (Hendrycks et al., 2021b), both focused on reasoning.

APPS (Hendrycks et al., 2021a) is a benchmark for code generation in LLMs. It provides problem descriptions, Python solutions, test cases, function names, and metadata. We use the Introductory level tasks, which include about 3,650 training samples. Due to input length constraints, we randomly select 2-3 solutions per problem, resulting in 9,771 training samples, and evaluate the model on 1,000 test samples.

MATH (Hendrycks et al., 2021b) contains high school math competition problems across multiple subjects, each with a detailed solution in LaTeX. Problems are rated from 1 (easiest) to 5 (hardest).

Table 1: Experimental results on the APPS dataset. The model trains on 1,500 samples per epoch from a 9,771-sample set over 5 epochs. GPTNeo, with fewer parameters, gets a 3,000-sample initial warm-up, not used for other models.

Method		GPTNeo-2.7B		Baichuan2-7B		Llama3-8B		Mistral2-7B	
		ACC	BLEU	ACC	BLEU	ACC	BLEU	ACC	BLEU
Baseline	Without SFT	3.46%	27.41	0.29%	16.07	1.23%	16.84	1.42%	52.48
	Random	8.66%	52.37	9.70%	50.53	27.44%	66.98	25.31%	65.48
	D2Pruning	5.79%	39.36	9.83%	45.91	28.52%	66.39	26.33%	64.96
	RHO-Loss	8.52%	47.50	12.81%	**56.82**	32.18%	68.73	30.63%	67.33
	Cherry	4.99%	22.63	10.33%	41.60	20.18%	49.95	20.14%	61.23
CL	Answer Row	6.47%	41.10	10.13%	43.32	32.78%	69.01	29.54%	**67.82**
	Answer Length	6.68%	34.78	14.33%	47.06	32.56%	67.43	26.06%	64.27
	Question Length	8.62%	46.49	12.63%	52.77	32.04%	**69.93**	27.14%	65.83
Ours	P3	**10.69%**	**54.04**	**14.45%**	56.59	**33.09%**	68.06	**31.04%**	67.47

We use subsets: Algebra (1,744 training, 1,187 test samples) and Counting & Probability (771 training, 474 test samples).

5.1.2 Models

To compare model performance, we fine-tune three LLMs of different sizes: **GPTNeo-2.7B** (Black et al., 2021), **Baichuan2-7B-chat** (Baichuan, 2023), **Llama3-8B-instruct** (AI@Meta, 2024), and **Mistral-7B-Instruct-v0.2** (Jiang et al., 2023) on the APPS and MATH datasets. For MATH, we excluded GPTNeo-2.7B because, despite being trained on GitHub code data, its smaller parameter size makes it unsuitable for the complex reasoning required in MATH.

5.1.3 Evaluation Metrics

For APPS, we assess code generation using BLEU scores and test case accuracy. For MATH, model success is based on final answer accuracy from step-by-step LaTeX solutions. In all metrics, higher scores indicate better performance.

5.1.4 Implementation Details

We use consistent parameters during fine-tuning. The initial learning rate is 0.0004, with a weight decay of 0.0001. To improve hardware efficiency, we use 4-bit quantization and set the computational data type to torch.float16. We apply Low-Rank Adaptation (LoRA) with $r = 64$, lora_alpha $= 16$, and a 0.1 dropout rate for the LoRA layer. APPS experiments are conducted on A6000 48G, while MATH experiments use A100 80G.

5.2 Baselines

We compare our method against several baselines in two categories: state-of-the-art coreset selection techniques (Mindermann et al., 2022; Maharana et al., 2023) and advanced data selection methods in NLP (Li et al., 2024b), as well as curriculum learning approaches that determine training sample difficulty.

The baseline methods are:

- **D2Pruning** (Maharana et al., 2023): Represents the dataset as a graph and selects coresets using message passing, updating difficulty scores based on neighboring examples.
- **RHO-Loss** (Mindermann et al., 2022): Estimates the impact of each data point on reducing loss for unseen data, focusing on non-noisy, non-redundant, and task-relevant examples.
- **Cherry** (Li et al., 2024b): Uses the Instruction-Following Difficulty (IFD) metric to select high-quality samples from open-source datasets.
- **Curriculum Learning**: Involves sorting training data by predefined difficulty and training sequentially, starting with easy examples. For APPS, we use the number of code lines and tokens; for MATH, we use manually assigned difficulty levels and token counts.

5.3 Experimental Results

5.3.1 Evaluation on APPS

Table 1 presents the evaluation of different optimization strategies on the APPS dataset using three language models. Baselines like D2Pruning and RHO-Loss yield varied results in Test Case Accuracy and BLEU scores. Random performs well with GPTNeo-2.7B, while RHO-Loss shows consistently strong performance across all models. Curriculum learning approaches using metrics like Answer Row, Answer Length, and Question Length showed mixed improvements.

Table 2: Experimental results on the MATH dataset. For Algebra, the model is trained on 300 out of a total of 1,744 samples over 5 epochs, with testing on 1,187 samples. For Counting and Probability, the model trains on 150 out of 771 samples per epoch, testing on 474 samples. No warm-up is used.

Method		Algebra			Counting and Probability		
		Baichuan2-7B	**Llama3-8B**	**Mistral-7B**	**Baichuan2-7B**	**Llama3-8B**	**Mistral-7B**
Baseline	Without SFT	2.30%	6.07%	6.49%	1.05%	5.48%	2.74%
	Random	6.32%	11.7%	5.56%	5.49%	8.44%	6.33%
	D2Pruning	4.30%	16.01%	6.74%	2.53%	12.23%	4.64%
	RHO-Loss	6.40%	**19.04%**	6.74%	4.43%	13.08%	7.38%
	Cherry	3.54%	11.8%	6.23%	2.53%	6.96%	4.22%
CL	Level	6.57%	15.2%	8.34%	4.01%	9.49%	5.06%
	Answer Length	5.39%	13.7%	7.58%	3.38%	7.38%	1.89%
	Question Length	8.00%	11.6%	7.58%	4.01%	8.23%	5.91%
Ours	P3	**9.35%**	17.10%	**9.77%**	**7.81%**	**13.29%**	**8.23%**

Table 3: Ablation studies, comparing the results of using DPP for diverse data selection versus not using DPP.

	APPS				Algebra			Counting and Probability		
	GPTNeo	**Baichuan2**	**Llama3**	**Mistral2**	**Baichuan2**	**Llama3**	**Mistral2**	**Baichuan2**	**Llama3**	**Mistral2**
SPL	8.71%	11.97%	32.85%	29.25%	9.10%	15.42%	9.10%	5.06%	8.86%	5.91%
P3	10.69%	14.45%	33.09%	31.04%	9.35%	17.10%	9.77%	7.81%	13.29%	8.23%

Figure 3 compares methods over five epochs in terms of accuracy and BLEU. The P3 method demonstrates rapid learning in early epochs and consistently outperforms baselines, with lower overfitting on Llama3-8B compared to Random and D2Pruning.

5.3.2 Evaluation on MATH

Table 2 shows results for the MATH dataset on Algebra and Counting & Probability tasks using Baichuan2-7B-chat and Llama3-8B-instruct. For Algebra, P3 achieves the highest accuracy of 9.35% on Baichuan2-7B-chat and 15.4% on Llama3-8B-instruct, comparable to the Level-based method. In Counting & Probability, P3 leads with 11.18% accuracy on Llama3-8B-instruct, underscoring its effectiveness for mathematical tasks and optimizing model performance.

5.3.3 In-Depth Analysis

In this section, we compare our framework to existing baselines (RQ1) and assess model performance.

Answer to RQ1:

- On the APPS dataset, our method outperforms RHO-Loss in both Test Case Accuracy and BLEU. Specifically, Test Case Accuracy improves by 25.5% on GPTNeo-2.7B, 12.8% on Baichuan2-7B-chat, and 2.83% on Llama3-8b-ins, consistently achieving better results throughout training.
- On MATH, P3 shows superior performance for both Algebra and Counting & Probability tasks.

5.3.4 Comparison with Full dataset

In this section, we compare P3 to full dataset training across tasks and models.

Table 4: APPS: Full vs. P3 performance.

APPS		Test Acc	BLEU
GPTNeo	**Full (9771*5)**	0.0799	45.17
	P3 (1500*5)	0.1069	54.04
Baichuan2	**Full (9771*5)**	0.1681	57.89
	P3 (1500*5)	0.1445	56.59
Llama3	**Full (9771*5)**	0.3096	65.02
	P3 (1500*5)	0.3309	68.06
Mistral2	**Full (9771*5)**	0.2761	67.01
	P3 (1500*5)	0.3104	67.47

Table 5: MATH: Full vs. P3 performance.

MATH	Algebra		C&P	
	Full (1744*5)	**P3 (300*5)**	**Full (771*5)**	**P3 (150*5)**
Baichuan2	0.0851	0.0935	0.0675	0.0781
Llama3	0.1592	0.1710	0.0992	0.1329
Mistral2	0.0834	0.0977	0.0780	0.0823

Answer to RQ2:

- Tables 4 and 5 show that P3, using fewer samples per epoch, achieves results comparable to full dataset fine-tuning, especially in BLEU scores on APPS. On MATH, P3 performs well in Counting_and_Probability and matches full dataset

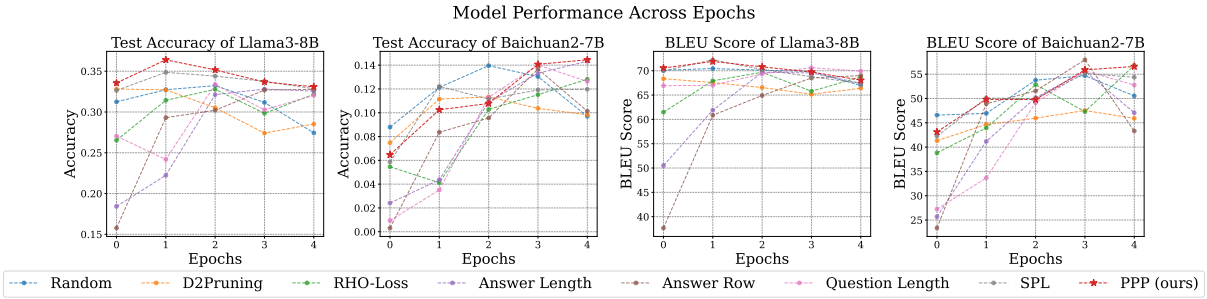

Figure 3: APPS dataset comparison over 5 epochs. Test accuracy and BLEU score show that our P3 method outperforms other baselines with faster learning and stronger results.

training in Algebra, demonstrating that P3 can achieve similar performance to full fine-tuning with significantly reduced data.

5.3.5 Ablation Study

In this section, we examine the effects of the two core components of our data selection framework: difficulty and diversity.

Answer to RQ3: We analyze the impact of different difficulty metrics on the APPS and MATH datasets:

- **Manual Metrics**: Metrics based on human annotations (level or answer rows) show limited effectiveness in guiding the model's learning process.
- **Model-Based Metrics**: Metrics using token counts (question/answer length) perform slightly better but fail to fully capture task complexity.
- **Policy-Based Metrics (P3)**: Policy-driven metrics dynamically assess task difficulty based on model actions. As shown in Table 3, P3 achieves superior results, with a Test Case Accuracy of 33.09% on APPS and 13.29% on MATH using Llama3-8b-ins. This suggests that policy-based metrics effectively capture task complexity and adapt well to the model's learning, outperforming manual and model-based approaches.

Answer to RQ4:

- **Impact of Diversity (DPP)**: The t-SNE visualization in Figure 1 shows diverse and uniformly distributed data across epochs, indicating effective data variety. Table 3 confirms that incorporating DPP improves model accuracy over SPL on both APPS and MATH datasets, validating DPP's role in enhancing performance through diverse data selection.

5.3.6 Hyper-Parameter Analysis

In this section, we analyze how the number of selected samples and epochs affect the performance of the Baichuan-7B-chat model on the Algebra dataset.

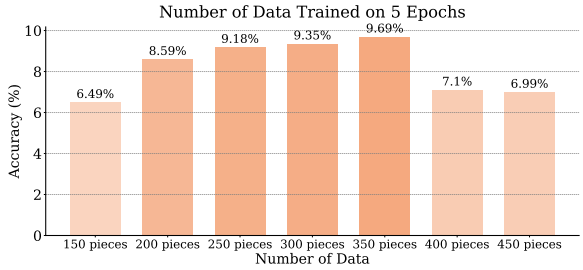

Figure 4: Effect of data size on acc over 5 epochs.

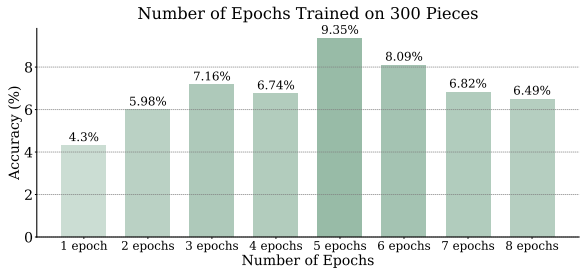

Figure 5: Effect of epochs on acc with 300 samples.

Answer to RQ5:

- Figure 4 shows that accuracy improves as more training data is added, but gains diminish beyond a certain point, indicating limited benefit from excessive data.
- Figure 5 demonstrates that performance initially improves with more epochs but declines after a certain point, suggesting an optimal stopping point is needed to avoid overfitting.

6 Conclusion

In this paper, we present P3, a novel framework for data pruning during training to fine-tune LLM. Experiments on APPS and MATH datasets demonstrate that P3 outperforms existing methods, offering an adaptive strategy for optimizing LLM performance. This work highlights the importance of adaptive data selection for efficient training and robust model performance.

7 Limitations

The performance of the P3 framework was primarily validated on reasoning scenarios, specifically the APPS and MATH datasets. While the results indicate significant improvements over traditional methods, the generalizability of the approach to other domains or datasets remains untested. In the future, we will explore the effectiveness of P3 across a broader array of NLP tasks.

References

Microsoft Research AI4Science and Microsoft Azure Quantum. 2023. The impact of large language models on scientific discovery: a preliminary study using gpt-4. *Preprint*, arXiv:2311.07361.

AI@Meta. 2024. Llama 3 model card. *Llama 3 Model Card*.

Alon Albalak, Yanai Elazar, Sang Michael Xie, Shayne Longpre, Nathan Lambert, Xinyi Wang, Niklas Muennighoff, Bairu Hou, Liangming Pan, Haewon Jeong, Colin Raffel, Shiyu Chang, Tatsunori Hashimoto, and William Yang Wang. 2024. A survey on data selection for language models. *Preprint*, arXiv:2402.16827.

Abdul Azeemi, Ihsan Qazi, and Agha Raza. 2023. Data pruning for efficient model pruning in neural machine translation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 236–246, Singapore. Association for Computational Linguistics.

Baichuan. 2023. Baichuan 2: Open large-scale language models. *arXiv preprint arXiv:2309.10305*.

Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. 2021. Gpt-neo: Large scale autoregressive language modeling with mesh-tensorflow. *Zenodo*. If you use this software, please cite it using these metadata.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *Preprint*, arXiv:2005.14165.

Laming Chen, Guoxin Zhang, and Eric Zhou. 2018. Fast greedy map inference for determinantal point process to improve recommendation diversity. In *NeurIPS*, pages 5627–5638.

Guanting Dong, Hongyi Yuan, Keming Lu, Chengpeng Li, Mingfeng Xue, Dayiheng Liu, Wei Wang, Zheng Yuan, Chang Zhou, and Jingren Zhou. 2023. How abilities in large language models are affected by supervised fine-tuning data composition. *ArXiv*, abs/2310.05492.

Peter Hase, Mohit Bansal, Peter Clark, and Sarah Wiegreffe. 2024. The unreasonable effectiveness of easy training data for hard tasks. *Preprint*, arXiv:2401.06751.

Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir Puranik, Horace He, Dawn Song, and Jacob Steinhardt. 2021a. Measuring coding challenge competence with apps. *NeurIPS*.

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

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Lu Jiang, Deyu Meng, Shoou-I Yu, Zhenzhong Lan, Shiguang Shan, and Alexander Hauptmann. 2014. Self-paced learning with diversity. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

Lu Jiang, Deyu Meng, Qian Zhao, Shiguang Shan, and Alexander Hauptmann. 2015. Self-paced curriculum learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29.

Yajing Kong, Liu Liu, Jun Wang, and Dacheng Tao. 2021. Adaptive curriculum learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5067–5076.

Alex Kulesza. 2012. Determinantal point processes for machine learning. *Foundations and Trends® in Machine Learning*, 5(2–3):123–286.

M. Kumar, Benjamin Packer, and Daphne Koller. 2010. Self-paced learning for latent variable models. In *Advances in Neural Information Processing Systems*, volume 23. Curran Associates, Inc.

Ming Li, Yong Zhang, Shwai He, Zhitao Li, Hongyu Zhao, Jianzong Wang, Ning Cheng, and Tianyi Zhou. 2024a. Superfiltering: Weak-to-strong data filtering for fast instruction-tuning. *arXiv preprint arXiv:2402.00530*.

Ming Li, Yong Zhang, Zhitao Li, Jiuhai Chen, Lichang Chen, Ning Cheng, Jianzong Wang, Tianyi Zhou, and Jing Xiao. 2024b. From quantity to quality: Boosting

llm performance with self-guided data selection for instruction tuning. *Preprint*, arXiv:2308.12032.

Adyasha Maharana, Prateek Yadav, and Mohit Bansal. 2023. D2 pruning: Message passing for balancing diversity and difficulty in data pruning. *arXiv preprint arXiv:2310.07931*.

Sören Mindermann, Jan M Brauner, Muhammed T Razzak, Mrinank Sharma, Andreas Kirsch, Winnie Xu, Benedikt Höltgen, Aidan N Gomez, Adrien Morisot, Sebastian Farquhar, and Yarin Gal. 2022. Prioritized training on points that are learnable, worth learning, and not yet learnt. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 15630–15649. PMLR.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2022. Cross-task generalization via natural language crowdsourcing instructions. *Preprint*, arXiv:2104.08773.

Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian. 2024. A comprehensive overview of large language models. *Preprint*, arXiv:2307.06435.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, et al. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Swabha Swayamdipta, Roy Schwartz, Nicholas Lourie, Yizhong Wang, Hannaneh Hajishirzi, Noah A. Smith, and Yejin Choi. 2020. Dataset cartography: Mapping and diagnosing datasets with training dynamics. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9275–9293, Online. Association for Computational Linguistics.

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

Hugo Touvron, Louis Martin, Kevin Stone, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Fuzhao Xue, Yao Fu, Wangchunshu Zhou, Zangwei Zheng, and Yang You. 2023. To repeat or not to repeat: Insights from scaling llm under token-crisis. In *Advances in Neural Information Processing Systems*, volume 36, pages 59304–59322. Curran Associates, Inc.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. 2024a. Lima: Less is more for alignment. *Advances in Neural Information Processing Systems*, 36.

Hongjian Zhou, Fenglin Liu, Boyang Gu, Xinyu Zou, Jinfa Huang, Jinge Wu, Yiru Li, Sam S. Chen, Peilin Zhou, Junling Liu, Yining Hua, Chengfeng Mao, Chenyu You, Xian Wu, Yefeng Zheng, Lei Clifton, Zheng Li, Jiebo Luo, and David A. Clifton. 2024b. A survey of large language models in medicine: Progress, application, and challenge. *Preprint*, arXiv:2311.05112.