Extended Abstract

Motivation

Large language models (LLMs) are typically fine-tuned on large task-specific datasets. However, such datasets often contain noisy or redundant samples that not only slow down training but may also degrade downstream performance. While data selection methods have proven effective for large-scale pretraining, they remain impractical for fine-tuning due to their computational cost and poor alignment with task-specific semantics. In this work, we address the challenge of efficient data selection for fine-tuning by leveraging reinforcement learning (RL) to identify high-quality training subsets that improve model generalization while significantly reducing training cost.

Method

We formulate the data selection task as a **`MDP`** over cluster subsets derived from the training data. States correspond to partial subsets of clusters, actions correspond to adding a new cluster, and the episode ends when the size of the partial subset reaches a fraction δ of the total number of clusters $|C|$. Rewards are defined via proxy-model-based signals, such as improvement in validation accuracy or reduction in training or validation loss. We evaluate several policies for navigating this state space, including **`DQN`**, **`PPO`**, and reward model-based search strategies **`CLIMB`** and its variants.

Implementation

We cluster the training data using KMeans or stratified KMeans over sentence embeddings and encode states using binary masks or mean-variance of sub-sampled representative points. Reward functions are derived from smaller proxy models (MobileLLM-125M or 600M) trained on sampled data points from each state. We implement **`DQN`** with a 5-layer MLP and **`PPO`** with actor-critic MLPs. **`CLIMB`** and **`CLIMB-Diffusion`** use reward model with random and diffusion-guided sampling over binary cluster selection masks respectively. All target model evaluations are conducted using MobileLLM-1.5B on held-out test dataset.

Results

We evaluate our methods on four classification tasks: **ANLI**, **Emotion**, **MetaHate**, and **MMLU**. Our RL-guided data selection strategies consistently outperform random and heuristic baselines. Notably, we find that training on a 5% subset selected by our methods can match or exceed full-dataset training:

MetaHate: **`CLIMB`** achieves **94.01%**, outperforming full-data (83.20%) by **+11.0** points.
Emotion: **`CLIMB`** reaches **68.40%**, exceeding full-data performance (68.10%) by **+0.30** points.
ANLI: **`DQN`** achieves **57.60%**, a gain of **+3.4** points over baseline selection, but below the full-data performance (64.76%).
MMLU: **`PPO + RND`** reaches **45.68%**, just **2.3** points below full-data (49.38%).

Discussion

Our results reveal that RL-based agents not only outperform random sampling and heuristics (e.g., top-loss, bandits), but also beat reward-approximation-based random search, indicating effective learning of informative subset policies. **`CLIMB`** variants are especially effective for noisy tasks like MetaHate. **`PPO`** performs best on MMLU and shows strong performance even with minimal warm-start. **`DQN`** is competitive across domains. Surprisingly, in some tasks, our policies outperform full training, suggesting that datasets may introduce harmful noise or redundancy that our policies learn to avoid.

Conclusion

This work presents a RL-based framework for data selection during fine-tuning. Our methods are effective against full training and baselines on all datasets. However, the best method varies across datasets and performance is highly susceptible to minor changes. These results show that by intelligently selecting the most informative training examples, we can reduce computational costs, energy consumption and carbon footprint while maintaining or enhancing model performance.

RL-Guided Data Selection for Language Model Finetuning

Harshit Gupta (gharshit) * **Animesh Jha (animjha)** * **Rohan Garg (rohang73)** *

Abstract

While data selection methods have proven valuable for large-scale language model pre-training, they remain computationally prohibitive for fine-tuning scenarios with smaller-scale datasets, which may also be plagued with noise and data redundancies. In this work, we introduce a principled reinforcement learning framework that efficiently identifies high-quality, non-redundant training samples for LLM fine-tuning. Our approaches combine clustering and semantic embedding-based state representations with efficient proxy model-based rewards and RL-based training approaches such as Deep Q-Networks and Proximal Policy Optimization to optimize for training subsets that maximize performance while minimizing computational costs. Experiments on text classification tasks demonstrate that training on data subsets selected by our approaches (as small as 5% of the original dataset) can achieve comparable or superior performance to training on the full dataset. Notably, we surpass full-data performance by upto 11% points in downstream task accuracy across datasets, with small computational overhead. These findings highlight the inefficiencies of existing large-scale data selection methods in small-data regimes and pave the way for scalable, task-specific fine-tuning strategies.

1. Introduction

Large Language Models (LLMs) are commonly adapted to specific tasks through *fine-tuning*, where model parameters are further trained on task-specific data. However, real-world datasets present two significant challenges: they often contain substantial noise that can lead to overfitting, and they include redundant data points where training on just one representative example would achieve the same performance as training on the entire group. Strategic data selection methods can address both issues by filtering out low-quality data and eliminating redundancies, improving model performance while reducing training time and costs.

However, existing data selection methods have primarily been designed for large-scale pretraining with terabyte-scale datasets. These approaches typically use computationally expensive techniques like gradient-based influence scoring (Xia et al., 2024) or domain-level optimization (Xie et al., 2024), which become impractical for smaller datasets. For instance, Xia et al., 2024 takes over two days to compute gradient stores for a medium-sized dataset of around 160,000 examples. Furthermore, these pretraining-oriented methods often perform poorly when applied to fine-tuning tasks, as they tend to focus on surface-level features rather than semantic content relevant to specific downstream tasks (Gu et al., 2024).

Reinforcement Learning (RL) represents an underexplored yet promising approach for data selection. RL can learn adaptive selection strategies tailored to specific models and tasks through weak supervision, enabling the estimation of data importance based on non-differentiable, complex reward functions such as the downstream performance of proxy models. Therefore, in this work, we focus on developing RL-guided approaches for training data selection, specifically for fine-tuning LLMs on medium-scale datasets ranging from 100,000 to 1 million data points.

We begin by clustering the training dataset, with the powerset of these clusters forming the state space of a Markov Decision Process (MDP). Actions involve adding new clusters to the current state, and rewards are defined based on validation accuracies and losses of smaller proxy models trained on data from the clusters in the resulting state. We explore various state encoding schemes and subsampling methods to efficiently compute rewards, and develop a family of RL-guided approaches based on Deep Q-Networks, Proximal Policy Optimization, Linear Bandits, and Reward Approximation for data subset optimization, given the MDP and reward function.

Experiments across four text classification datasets demonstrate that our RL-guided approaches consistently outperform heuristic-based data selection methods by up to 8.3 percentage points in downstream task accuracy for the same data selection fractions. Notably, for the MetaHate dataset, training on a 5% data subset selected by our RL-guided approach surpasses full-dataset training performance by 11 percentage points. Our findings indicate that different RL-guided approaches excel with different datasets and proxy models, and we also perform ablation studies on design con-

siderations for the DQN, PPO, reward model based methods. By varying data selection fractions, we show that RL-guided approaches achieve a good balance between downstream performance and training efficiency, demonstrating substantial potential for RL-guided methods for data subset selection in LLM fine-tuning.

2. Related Work

2.1. Statistical Foundations of Data Selection

Signficant recent research focuses on establishing a rigorous statistical framework for data selection. Under this paradigm, the goal is to identify a subset of the training data that preserves (or even improves) generalization by reducing noise and redundancy. In Kolossov et al., 2023, the authors develop a statistical theory for subsampling under weak supervision across a variety of model classes. Their analysis provides conditions under which substantial data reduction is achievable without sacrificing—and sometimes even enhancing—generalization performance. This line of thinking has been extended to frame data selection as an information-theoretic problem (Deb et al., 2025), selecting examples that maximize the information gain, measured by the determinant of the Fisher information matrix. In addition, the DSDM framework (Engstrom et al., 2024) introduces a model-aware approach which analyse individual data points' influence on specific target samples, providing granular insights into data utility to guide the selection process. Similarly, Influence Distillation (Nikdan et al., 2025) formulates influence using second-order information and uses a landmark-based approximation to make selection tractable. Recent work by Gu et al., 2024 reformulates data selection as an optimal control problem solvable via Pontryagin's Maximum Principle.

2.2. Data Selection for Language Models

In the context of large language models (LLMs), data selection plays a critical role given the enormous and ever-growing training datasets (Albalak et al., 2024). One influential approach is the LESS framework (Xia et al., 2024), which estimates the influence of individual samples by constructing gradient stores and quantifying their contributions to model convergence. However, as noted previously (Yin & Rush, 2025; Liu et al., 2025), the high computational cost of such gradient-based methods can make them suboptimal in practice; cheaper methods may yield better overall performance when both selection and training costs are considered. Complementary to LESS, methods such as DSIR (Xie et al., 2023) utilize importance resampling to select examples that are statistically most beneficial for pre-training. Recent advances also include techniques such as DoReMi (Xie et al., 2024) which optimize data mixtures to accelerate language model pretraining. Other strategies include data pruning (Marion et al., 2023) and deduplication methods like D4 (Tirumala et al., 2023) and SemDeDup (Abbas et al., 2023) that aim to improve training efficiency and reduce redundancy. More recently, CLiMB (Diao et al., 2025) iteratively samples random data mixtures, evaluates them, and trains a predictor that guides subsequent mixture selection. Another class of methods are based on adaptive curricula, Progressive Data Dropout (S et al., 2025) gradually skips an increasing fraction of training subset each epoch. Its utility in LLM fine-tuning remains an open question because language models typically observe each token once or only a few times. For VLM finetuning, PROGRESS (Chandhok et al., 2025) dynamically ranks training data clusters by the model's learning progress and selects those with the greatest headroom of improvement.

2.3. Reinforcement Learning for Feature Selection

A long line of work exists for RL-based techniques for optimal feature subset. Early work framed feature selection as a one-player game (Gaudel & Sebag, 2010), where an RL agent explores the powerset of features using methods such as a variant of Upper Confidence Trees (Kocsis & Szepesvári, 2006). Although originally devised for selecting features, these RL-based approaches provide natural baselines for more general data subset selection problems. The formulation of data selection as a sequential decision-making process allows the agent to iteratively refine the subset based on feedback from model performance.

3. Methodology

We formulate the data selection task as a reinforcement learning (RL) problem. Formally, given training dataset of size $|D|$, we aim to select the data subset of size $\delta|D|$ (where δ is the selection fraction) that maximizes downstream performance when used to train a *target* model.

3.1. MDP Setup: States and Actions

Given a training set $\mathcal{D}$, we cluster it into C clusters of data points based on their semantic embeddings via K-means clustering. (**`Kmeans`**) To induce label information in the clusters, we also try a variant where we enforce a cluster to have data points corresponding to only one label (henceforth called **`Stratified-Kmeans`**).

The state space $\mathcal{S} = \mathcal{P}(C)$ is then the set of all possible subsets of C. At any step we maintain the state $s_t \subseteq C$ which denotes the set of clusters selected in the current episode so far as candidates for the target model's training dataset; an episode is terminated when the selected subset reaches size $|s_H| = \delta|C|$. The action space $\mathcal{A}$ consists of selecting a new cluster $a_t \in C \setminus s_t$ and adding it to the current set of clusters to get the new state $s_{t+1} = s_t \cup \{a_t\}$, see Figure 1. While this formulation allows for the

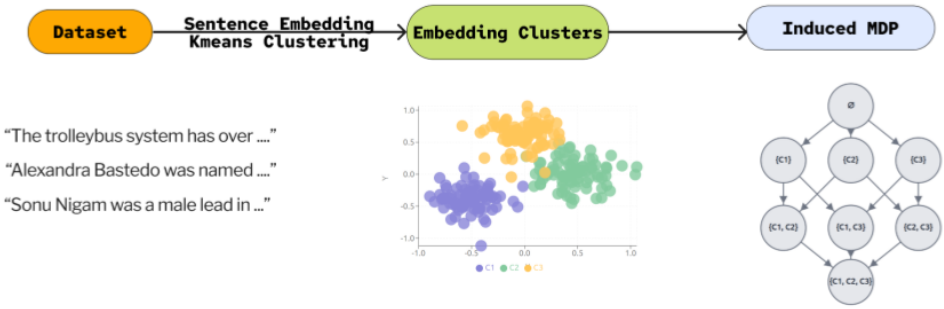


Figure 1. MDP induced by a dataset

removal of clusters during the episode as well, we only focus on additions in this work. For a given state s_t, we explore different ways of computing a state encoding $\phi(s_t)$. The simplest encoding, denoted by **`Binary-Mask`**, is $|C|$-length binary vector with $\phi_i(s_t) = 1 \Leftrightarrow C_i \in s_t$. In another case (**`Mean-Std`**), we use:

$$\phi(s_t) = [\mu(s_t), \sigma^2(s_t)],$$

where $\mu(\cdot)$ and $\sigma^2(\cdot)$ are the mean and variance of the cluster-centroid embeddings in the currently selected set. Another variant (**`Concat`**) involves concatenating embeddings of representative samples from each cluster. We explore two approaches for selecting these representative samples — choosing them at random from the cluster (**`Random`**) or choosing the furthest points from the cluster centroid (**`Furthest`**), capturing the spread of the cluster.

3.2. Reward Approximation

We test three reward functions computed using a *proxy* model, which is a smaller model than the *target* model, usually from the same model family. Let Val-Acc$(\cdot)$ be the validation accuracy of the proxy model and $\mathcal{L}(\mathbf{D_v}|\mathbf{D_t})$ be the loss value for dataset $\mathbf{D_v}$ after training on $\mathbf{D_t}$. (for clearness, We omit $\mathbf{D_t}$ if it is same as $\mathbf{D_v}$)

Accuracy-based Reward (R_{acc}): This reward function computes the improvement in validation accuracy when adding a new cluster to the selected data, thus capturing its impact on the downstream performance of the proxy model:

$$R_{\text{acc}}(s_t, a_t) = \text{Val-Acc}(s_t \cup \{a_t\}) \; - \; \text{Val-Acc}(s_t). \quad (1)$$

Although effective, measuring changes in validation accuracy entails retraining the proxy model from scratch after each action for a substantial number of training steps and performing evaluation, which is extremely expensive.

Training Loss-based Reward ($R_{\text{loss}}^{\text{train}}$): This reward function makes two assumptions — training losses on the same batches of data are correlated for the *target* and *proxy* model, and training loss for a model is negatively correlated with downstream performance. Then, the reward function measures changes in the proxy model's training loss when the new cluster is added to the current state:

$$f(x) = 5 \; - \; 2\ln(2x) \quad (2)$$

$$R_{\text{loss}}(s_t, a_t) \; = \; f\big(\mathcal{L}(\xi(s_t) \cup \xi(\{a_t\})\big) \; - \; f\big(\mathcal{L}(\xi(s_t))\big). \quad (3)$$

where $\ln(\cdot)$ is the natural logarithm, and a subsampling function $\xi(\cdot)$ is used to select a fixed number of data points (set as a hyperparameter) from each cluster to estimate the training loss from the *proxy* model at the end of multiple epochs of training. The logarithmic transformation $f(\cdot)$ serves a dual purpose: it establishes a baseline of $f(\mathcal{L}(\emptyset)) = 0$ while also magnifying subtle loss variations in the low-loss regime of training on larger subsets of data. $R_{\textbf{loss}}$ is much faster than $R_{\textbf{acc}}$, taking 2 seconds and 90 seconds to compute, respectively, for a state containing 4 clusters, which makes training more efficient.

Validation loss-based Reward ($R_{\text{loss}}^{\text{val}}$): This reward function is similar to $R_{\text{loss}}^{\text{train}}$, except for using validation-set loss instead of training loss. Formally,

$$R_{\text{loss}}^{\text{val}}(s_t, a_t) \; = \; f\big(\mathcal{L}(\xi_{val}(\mathbf{D}_{val})|\xi(s_t) \cup \xi(\{a_t\})\big) \; - \; f\big(\mathcal{L}(\xi_{val}(\mathbf{D}_{val})|\xi(s_t))\big). \quad (4)$$

where the subsampling function $\xi_{val}(\cdot)$ is used to select a fixed number of data points (set as a hyperparameter) from the validation set, keeping the label proportion constant. f serves a similar purpose to that in $R_{\textbf{loss}}$. While $R_{\textbf{loss}}^{\textbf{val}}$ is slower than $R_{\textbf{loss}}^{\textbf{train}}$) (5 seconds v/s 2 seconds), it is much better correlated with downstream performance.

Random Network Distillation (RND): For each of the reward approximations described above, Random Network Distillation (Burda et al., 2018) can be added to improve exploration of the policy. RND is implemented using a 4-layer MLP with MSE loss between the target and predictor

network as intrinsic reward. The state and rewards are normalized using a running average to stabilize the intrinsic rewards.

3.3. Policies

We implement three different types of RL-based approaches for the data cluster selection task: (1) a Deep Q-Network-based (DQN) (Mnih et al., 2013) based policy, (2) a Proximal Policy Optimization (PPO) (Schulman et al., 2017) based policy, (3) Reward Model Based approaches (Diao et al., 2025; Sutton, 1991). All approaches build on the state/action definitions and reward functions provided earlier, but differ in how they learn the policy and approximate the value function.

DQN: At each state s_t, we compute an embedding $\phi(s_t)$ using one of the state encoding methods. We then feed $\phi(s_t)$ into a function approximator $f_\theta(\cdot)$, either an **`MLP`** or a small **`Transformer`**, which outputs an $|\mathcal{A}|-$dimensional vector where each component represents the estimated Q-value (or "goodness") of taking action $a \in \mathcal{A}$ in the current state s_t. We then mask out actions corresponding to the clusters already in s_t and choose the action with the highest Q-value via ϵ-greedy sampling. The network parameters θ are then optimized through experience replay updates.

PPO: We adopt a variant of PPO that supports the masking of invalid actions (Huang & Ontañón, 2022). Both the actor and critic networks are 3-layer MLPs; for each state s_t, the actor outputs a probability distribution over available cluster actions, while the critic estimates the value of s_t. We investigate two variants of PPO as well. We first try training PPO from **`Scratch`**, initializing the actor and critic randomly. Next, we try to give PPO a **`Warm Start`**. We pre-train the critic using a regression task on rewards for "single-cluster" states. Specifically, for each cluster $c_i \in \mathcal{A}$, we compute the average reward obtained when taking action c_i on the state containing the empty set to reach state s_i. We then regress the critic network on the (s_0,c_i,s_i,r_i) tuples, where $s_0 = \emptyset$ and r_i corresponds to the average reward for each single-cluster addition. This setup encourages the critic to produce, for the start state, outputs that rank clusters in proportion to their individual expected returns.

3.4. Reward Model Based Strategies

These strategies approximate the true reward function in order to accelerate policy learning by generating additional, "synthetic" rollouts. Concretely, we train a proxy reward model $\hat{r}_\phi(s,a)$ on true reward signals $r(s,a)$ and then use $\hat{r}_\phi$ to label transitions sampled under the current policy, and mix these synthetic transitions with real ones when updating the agent. Real rollouts are given higher weight. Based on the agent, we have two strategies: **`DynaDQN`** and **`CLIMB`**.

DynaDQN: The proxy reward $\hat{r}_\phi$ is implemented as an ensemble of four independently initialized, 5-layer MLPs. Each ensemble member is trained on real transitions using mean-squared error (`MSE`) loss with ℓ_2 regularization. **`MLP`** variant of DQN is used as the policy. At each environment step, we sample a batch of 32 state–action pairs, compute their proxy rewards by averaging the ensemble outputs, and then only insert those synthetic transitions into the replay buffer if the ensemble-standard deviation falls below a fixed threshold $\sigma_{\max}$. Synthetic transitions are retained for at most four episodes, and during learning, they are weighted by an importance factor of 0.5 relative to real transitions.

CLIMB: Drawing inspiration from (Diao et al., 2025), we implemented **`CLIMB`** for discrete states. For this strategy, the reward function $r(s)$ is the absolute value instead of the increment from the previous state. The proxy reward model $\hat{r}_\phi(s)$ is a single 3-layer MLP trained with `MSE` loss. In each iteration, we uniformly sample M previously unseen states, rank them by their estimated reward $\hat{r}_\phi$, then query the environment for the true reward of the top-K states and use these K new labels to update $\hat{r}_\phi$. After T epochs, we re-evaluate all seen states under $\hat{r}_\phi$ and select the highest-scoring one as the final best state.

To improve the exploration in **`CLIMB`**, ensembling (**`CLIMB-ensemble`**) and diffusion (**`CLIMB-diffusion`**) are used. In **`CLIMB-ensemble`**, we implemented an ensemble of reward models, shifting from a purely exploitation-driven strategy to one that incorporates UCB-based exploration rewards. For selecting the top K states, we used the standard deviation in the forward pass of the ensemble networks as our exploration term.

$$\text{UCB}(s) = \frac{1}{E}\sum_{i=1}^{E}\hat{r}_i(s) + \beta \cdot \sqrt{\frac{1}{E}\sum_{i=1}^{E}(\hat{r}_i(s) - \mu(s))^2} \quad (5)$$

where $\hat{r}_i$ is the i^{th} reward network out of total E and β is a hyperparameter to tune the exploration. Higher values of β encourages the algorithm to learn the actual reward for the masks where it has more uncertainty.

`CLIMB-diffusion`: Instead of randomly sampling in 2^M states we sample through a reverse-diffusion process. Specifically, we introduce a Denoising Diffusion Probabilistic Model (DDPM) (Austin et al., 2023; Dhariwal & Nichol, 2021) to learn the distribution of high-reward states encountered so far. The reverse-diffusion process runs for τ timesteps. At each timestep t, the score for a noisy state s_t is guided by the gradient of the proxy reward model $\hat{r}_\phi(s)$ and modified as follows:

$$\text{score}(s_t) = \text{score}_{\text{diffusion}}(s_t, t) + \gamma \cdot \nabla_{s_t}\hat{r}_\phi(s_t) \quad (6)$$

where γ is a guidance scale hyperparameter that balances between the learned distribution and reward maximization. After computing (6) we add Gumbel noise and project to the top indices to get a binary mask with the correct number of clusters selected.

3.5. Linear Bandits Formulation

We also explore an alternative linear bandit approach, where each cluster is treated as a separate "arm" in a multi-armed bandit framework. Although the reward for selecting a cluster is inherently noisy — due to subsampling before computing the proxy loss and the corresponding variances during reward function computation — we can approximate an upper-confidence bound (UCB) or adopt an $\epsilon-$greedy strategy to identify the most promising arms. We design a policy π parameterized by a 2-layer MLP that, given an action encoding which is the same as the feature encoding of a single cluster state, outputs a predicted "value" for that arm. The MLP thus extrapolates across similar clusters, enabling far more efficient learning than a tabular bandit strategy, which would assign a separate value to each cluster without sharing information. We use the MLP to estimate both a mean reward and its uncertainty; for the final subset selection we pick the top-k clusters/arms with the highest upper-confidence bounds. While this method scales to a large number of clusters allowing finer granualarity, it can not capture the effect of the interaction of different clusters during training.

Dataset	Task	Train Size	Test Size	# Labels
ANLI	Natural Language Inference	162,400	3,200	3
MetaHate	Hate Speech Detection	1,051,165	25,000	2
GooglePlay	Sentiment Classification	98,836	5,000	5
MMLU	MCQ Answering	99,842	14042	4

Table 1. Summary of text classification datasets with their respective tasks, training sizes, test sizes, and number of labels.

4. Experimental Setup

4.1. Downstream Tasks

We focus on text classification tasks training the LLM to output a single-token class label in response to a text input and a fixed prompt. We chose this task because the single-logit losses can be compared more consistently across different sequence lengths and models. Table 1 contains information about the datasets used. The MetaHate and GooglePlay datasets do not have an explicit test split, so we randomly sample 25K and 5K samples respectively to create one.

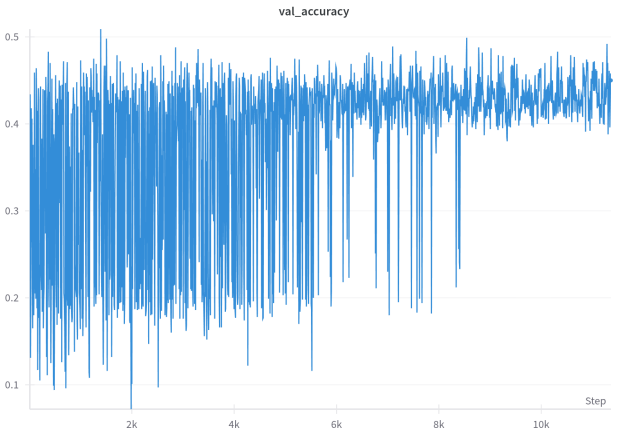


Figure 2. Reward (Validation Accuracy) of sampled states for Climb Diffusion on Emotion with the 600M proxy model.

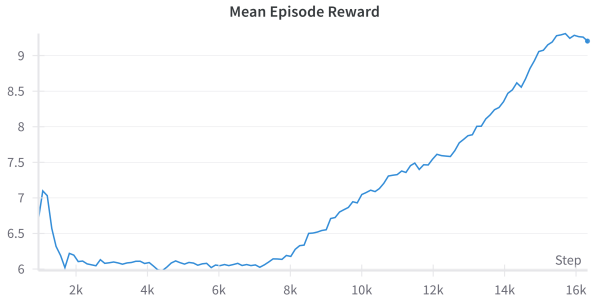


Figure 3. Reward (Train loss) vs episode for PPO on MetaHate with the 125M proxy model.

4.2. Models

In our experiments, we utilize the MobileLLM (at Meta) model family, that provides pre-trained language models with the same architecture at varying scales. Specifically, we use MobileLLM-600M and MobileLLM-125M as proxy models to estimate the influence of training data points on downstream performance, while MobileLLM-1.5B is used as the target model for training and evaluation.

4.3. Baselines

For our results, we fix the data selection percentage to 5% unless otherwise mentioned. We compare our approaches to the following baselines:

1. **Full:** The target model is trained on the entire dataset (100% of the data).

2. **Random-5%:** A random 5% subset is used as training data for the target model. This serves as a natural baseline for data efficiency.

3. **Top-5%:** Using the proxy model, we compute the loss of each data point in the training dataset. We then select the top 5% high-loss or harder examples.

4. **Bottom-5%** Similarly, we select the bottom 5% examples, hypothesizing that high-loss examples may have been noisy or mislabelled. **Top-5%** and **Bottom-5%** check whether focusing on "hard" or "easy" points during training yields improvements.

5. **Random-Search-5%** Similar to **Random-5%:**, but multiple subsets are evaluated and the one with best proxy-reward is used as the final sample.

6. **LESS and DSIR:** These are additional state-of-the-art data selection methods from recent literature (Xia et al., 2024; Xie et al., 2023). However, in our early experiments, these methods performed worse than random sampling when applied to our smaller-scale datasets, and LESS is prohibitively expensive to run in terms of computation, requiring two days to construct gradient stores for ANLI. Therefore, we omit these baselines from our reported results.

4.4. Evaluation

We report accuracy on a held-out test set for each dataset, for a target model trained on the data subsets selected by the different approaches. We also perform experiments (Section 6) over varying selection percentages to capture different trade-offs between dataset size, training time, and model accuracy.

5. Results

Hyperparameters. `BAAI/bge-small-en-v1.5` is used to obtain semantic embeddings for the training datasets and K-Means or stratified K-Means clustering is used to cluster the resulting embeddings into 64 (or 128) clusters. We use a batch size of 16 with 4 gradient accumulation steps to train the proxy model for 2 epochs with a learning rate of 1e-5. For each cluster, 64 data points are sampled for proxy-model training.

For the DQN, we use a 5-layer **`MLP`** of size 256 to learn the Q-function, with **`Mean-Std`** state encodings and **`Furthest`** subsampling. We use $\gamma = 0.99$ and decaying ϵ starting from 1 with a decay of 0.99 per episode and a minimum of 0.01. A replay buffer is used and steps are sampled in batches of 32 to train the model. A learning rate of 1e-4 is used to train the DQN network and the target network is updated every 10 steps. The DQN is trained for 500 episodes. PPO is trained with a learning rate of 3e-4, for 500 episodes. For the linear bandits approach, we train for 1000 steps with a UCB coefficient of 2 and learning rate of 1e-4. In DynaDQN, the reward model has a hidden dimension of 256, and the same configuration as DQN is used for policy. Learning rate of $5e-4$ is used with no training for first 5 episodes. CLIMB is trained with 50 iterations, sampling 128 states and selecting top 32 states finally at each step. The hidden dimension is set to 128, and learning rate of 1e-4 is used with the reward model trained for 2 epochs per iteration.

We train the target model for 4 epochs on the selected data subsets, with a batch size of 4 and 8 gradient accumulation steps, and use a cosine annealing schedule for the learning rate from 1e-5 to 1e-6 and linear warmup for the first 5% of training steps. Checkpoints are chosen based on highest validation accuracy for all settings to compute downstream performance.

Algorithm	ANLI	Emotion	Hate	MMLU
Full	64.76	68.10	83.20	49.38
Random	54.20	58.30	72.60	40.90
Top 5%	57.40	21.90	84.00	37.34
Lowest 5%	57.10	22.60	77.80	22.96
Random Search	55.61	59.30	72.60	43.71
Top-k linear bandits	52.33	50.30	83.10	45.57
DQN	**57.60**	65.60	69.40	44.27
DQN + RND	35.30	63.76	70.91	44.18
DynaDQN	52.96	61.94	50.50	45.11
PPO	54.20	62.32	60.85	44.80
PPO Warm Start	56.24	60.24	87.95	44.19
PPO + RND	55.80	56.52	59.50	**45.68**
CLIMB	53.83	**68.40**	**94.01**	41.73
CLIMB + Diffusion	33.30	46.76	**94.01**	44.51
CLIMB + Ensemble	–	55.54	–	–

Table 2. Performance of MobileLLM-1.5B when trained on data selected using various strategies. All strategies are discussed in Section 3. Numbers for the baseline trained with the full training dataset are provided in the first row, while the best numbers for the data selection approaches are highlighted. CLiMB + Ensemble does not finish within time.

Discussion. We report results with MobileLLM-1.5B as the target model in Table 2. Since ANLI has three splits, we report macro-averaged accuracies over each of the three splits.

Contrary to expectations, training on the full dataset results in the best performance in only 2 out of the 4 datasets. Using a randomly selected 5% subset results in a performance drop of approximately 10 points in all cases. The `Bottom 5%` baseline, which selects examples with the lowest proxy model loss, performs poorly across the board, suggesting that such examples are not informative for continued training for the target model. The `Top 5%` heuristic performs well on ANLI and Hate, but fails on Emotion and MMLU, indicating its limited generalizability. In contrast, reward ap-

proximation based `Random Search` improves upon random selection in all settings except Hate, where it maintains parity. RL-based agents consistently outperform both random selection and heuristic baselines, including reward-based random search, suggesting that the agents are learning policies that capture more than just immediate reward approximation. They achieve the best performance on all 4 datasets, even exceeding training on the full dataset in two cases (MetaHate and Emotion). On MetaHate, both CLIMB and its variant outperform the full-data upper bound, showing a striking 11-point gain. PPO also surpasses the full-data baseline by 4.8 points. On Emotion, CLIMB slightly outperforms the full model, and DQN comes within 3 points of full-data performance. On ANLI, DQN performs best among selection-based methods, improving over the random baseline by over 3 points, however, a significant gap to the full dataset remains, which we analyze further in Section 6.6. Finally, on MMLU, PPO achieves the highest performance, coming within 3 points of full-data performance.

We also present reward graphs for Climb Diffusion on Emotion (Figure 2) and PPO on MetaHate (Figure 3) over the course of training. The graphs reveal how for Climb Diffusion the reward model becomes better at filtering out bad sampled states and the sampling itself moves towards high reward states due to the diffusion prior. PPO demonstrates a clear upward trend in rewards throughout training, which aligns with its strong performance on MetaHate, as shown in Table 2.

6. Analysis

6.1. Varying Number of clusters

We evaluate **`Random-Search`** algorithm over a range of cluster counts $C \in \{64, 256, 1024, 4096\}$, with results shown in Figure 4. As C increases, we observe a consistent improvement in the downstream performance. However, the total runtime grows approximately quadratically in C, since both the number of episodes and the number of proxy sub-samples per reward evaluation increase with the cluster count. Balancing this trade-off between solution quality and computational cost, we fix $C = 64$ and proxy subsamples to 64.

6.2. Clustering Strategy

The **`Stratified-kmeans`** method exhibits suboptimal performance when the number of clusters is small and the number of class labels is large. This is primarily due to its inability to ensure representation of all labels in the selected subset, which leads to label imbalance. However, as the number of clusters increases, its performance improves, as shown in Figure 4. This improvement is attributed to the greater flexibility in selecting samples with more diverse

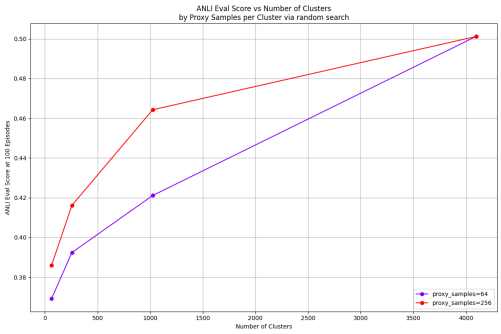


Figure 4. Downstream performance vs number of clusters for ANLI with **`Random-Search`** and **`stratified-kmeans`** for different proxy subsamples.

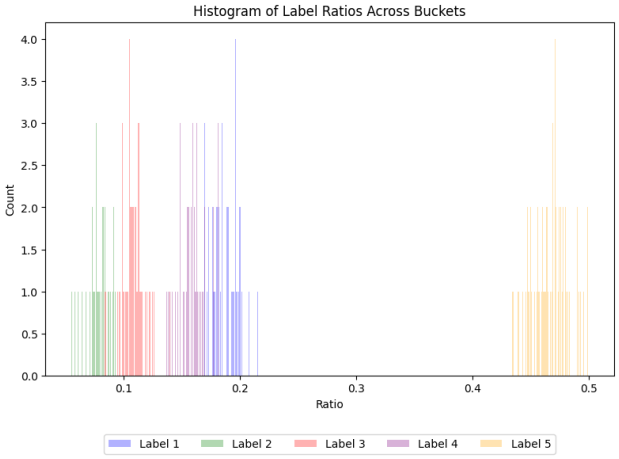


Figure 5. Histogram of label ratios across clusters using **`K-means`** in the Emotion dataset. Each color corresponds to a different label, and the x-axis indicates the proportion of samples within a cluster that belong to each label.

label distributions across an increased number of clusters.

In contrast, **`K-means`** tends to preserve the overall label distribution more consistently, making it more effective when the number of clusters is limited. This distinction is illustrated in Figure 5, which presents the distribution of label proportions across clusters for the Emotion dataset. The figure demonstrates how label representation varies between the two methods and supports the superior performance of K-means in scenarios with fewer clusters.

6.3. Comparison for Different State Encoders

`DQN` : We present results for DQN methods with various state encoding methods, subsampling strategies, and DQN models across three datasets and two proxy models in Table 3. Our findings indicate that the **`Furthest`** subsampling strategy outperforms the **`Random`** strategy in nearly all cases, except for the 125M proxy model on Google-

Dataset	State Representation	Subsampling	DQN Model	600M Proxy Accuracy (↑)	125M Proxy Accuracy (↑)
ANLI	`Mean-Std`	`Furthest`	`MLP`	**57.6**	**57.2**
	`Mean-Std`	`Random`	`MLP`	52.9	54.6
	`Concat`	`Furthest`	`Transformer`	56.0	56.6
	`Concat`	`Random`	`Transformer`	54.9	53.2
MetaHate	`Mean-Std`	`Furthest`	`MLP`	**69.4**	63.4
	`Mean-Std`	`Random`	`MLP`	67.0	36.0
	`Concat`	`Furthest`	`Transformer`	60.9	61.6
	`Concat`	`Random`	`Transformer`	67.4	**66.0**
GooglePlay	`Mean-Std`	`Furthest`	`MLP`	**65.6**	60.6
	`Mean-Std`	`Random`	`MLP`	65.1	62.3
	`Concat`	`Furthest`	`Transformer`	61.8	59.4
	`Concat`	`Random`	`Transformer`	63.3	**64.9**

Table 3. Performance of MobileLLM-1.5B when trained on data selected using various DQN variants and two different proxy models. All strategies are discussed in Section 3. The best numbers for the data selection approaches are highlighted.

Clustering Type	# clusters/subsamples	Proxy Model	State encoder	R_{acc}	$R_{\text{loss}}^{\text{train}}$	$R_{\text{loss}}^{\text{val}}$
Kmeans	64/64	125M	`Binary-Mask`	**65.50%**	62.12%	65.36%
			`Mean-Std`	65.04%	64.40%	61.88%
Kmeans	64/64	600M	`Binary-Mask`	**68.40%**	64.40%	65.58
			`Mean-Std`	62.62%	63.84%	59.42%
Stratified Kmeans	128/32	125M	`Binary-Mask`	**61.38%**	46.90%	46.16%
			`Mean-Std`	55.36%	56.28%	46.76%

Table 4. Performance of MobileLLM-1.5B for GooglePlay dataset when trained on $1/16$ data selected using CLIMB with different state encodings and different reward functions. All strategies are discussed in Section 3. The best numbers for each configuration of environment is highlighted.

Play and the `Transformer`-based DQNs on MetaHate and GooglePlay. Notably the additional expressive power provided by the `Transformer` does not generally lead to better performance compared to the `MLP`-based approach, except for the 125M proxy model on MetaHate and GooglePlay. Overall, using the 600M proxy model tends to yield better results for DQN-based approaches across all datasets. While there are no clear winners, using the `Mean-Std` state encoding with `Furthest` sampling and a `MLP`-based DQN results in generally strong performance across datasets.

`CLIMB` : We present the results for running CLIMB for multiple configurations of environments with `Furthest` subsampling in Table 4. Note that `Stratified-Kmeans` is run with 128/32 to allow for representation of all (5) labels in chosen clusters. From the numbers, we find that R_{acc} with `Binary-Mask` performs the best in all configurations and 600M performs better than 125M. Also, $R_{\text{loss}}^{\text{train}}$ performs better with `Mean-Std`, while $R_{\text{loss}}^{\text{val}}$ performs better with `Binary-Mask`. These results suggest that the semantic information presented in state by `Mean-Std` is not meaningful in case of validation set based rewards. Given the much higher time taken by R_{acc}, $R_{\text{loss}}^{\text{val}}$ with `Binary-Mask` is the most suitable choice.

6.4. Strategy Specific Comparisons

PPO Warm Start We present results for PPO with and without the `Warm Start` in Table 5 for all four datasets and two proxy models. The `Warm Start` is beneficial to the performance of PPO for both ANLI and MetaHate, but worsens performance slightly on GooglePlay and MMLU. Notably, the `Warm Start` nearly doubles downstream performance for MetaHate with the 125M proxy model.

RND: We evaluate the performance of the `RND` environment using $R_{\text{loss}}^{\text{val}}$ as the base reward signal with `DQN-MLP` and `PPO` policies. The corresponding results are presented in Table 6. It indicates that `RND` yields only marginal improvements in performance for the MetaHate task with `DQN-MLP` and the MMLU task with `PPO`, while substantially degrading performance across all other task–algorithm combinations. These results suggest that RND does not provide meaningful benefits for this MDP.

Reward Model Based Strategies: Comparing the performance of various reward model based strategies in table 2, we find that `CLIMB` demonstrates consistently strong performance, outperforming all other strategies for GooglePlay and MetaHate. In contrast, while DynaDQN slightly

Dataset	Variant	600M Proxy Accuracy (↑)	125M Proxy Accuracy (↑)
ANLI	`Scratch`	54.2	53.7
	`Warm Start`	**55.8**	**54.9**
MetaHate	`Scratch`	60.9	45.9
	`Warm Start`	**73.1**	**88.0**
GooglePlay	`Scratch`	**62.3**	**61.7**
	`Warm Start`	55.8	60.2
MMLU	`Scratch`	**44.8**	-
	`Warm Start`	44.19	-

Table 5. Performance of MobileLLM-1.5B when trained on data selected using PPO with and without warm starts and two different proxy models. The best numbers are highlighted.

Dataset	Variant	DQN Accuracy (↑)	PPO Accuracy (↑)
ANLI	`Val-Loss`	**57.6**	**56.24**
	`RND`	35.3	55.8
MetaHate	`Val-Loss`	69.4	**87.95**
	`RND`	**70.91**	59.5
GooglePlay (Emotion)	`Val-Loss`	**65.6**	**60.24**
	`RND`	63.76	56.52
MMLU	`Val-Loss`	**44.27**	44.19
	`RND`	44.18	**45.68**

Table 6. Performance of MobileLLM-1.5B when trained on data selected using PPO and DQN with and without RND exploration reward. The best numbers are highlighted.

surpasses DQN on MMLU, it underperforms significantly on ANLI, GooglePlay, and MetaHate. This suggests that the synthetic rollouts generated by reward model are not helpful, possibly due to inaccurate reward model leading to noisy rewards. Similarly, diffusion yields markedly lower performance on ANLI and GooglePlay, with only marginal improvements observed on MMLU. Ensembling is impractical for this MDP requiring over five times the training time compared to full dataset training.

6.5. Varying Selection Fractions

To obtain a better estimate of the trade-offs between training time and performance improvements, we vary the selection fraction in $[\frac{1}{32}, \frac{1}{16}, \frac{1}{8}]$ and present results for two DQN configurations with the 125M proxy model: (1) DQN with **`Mean-Std`** state encodings, **`Furthest`** subsampling, and an **`MLP`** (**DQN (F)**), and (2) DQN with **`Concat`** state encodings, **`Furthest`** subsampling, and a **`Transformer`** (**DQN-T (F)**) in Figure 6. For comparison, we also include results for the **Random** and **Full** baselines. The reported wall-clock times account for the combined duration of training the DQN and subsequently training the target model on the selected data subsets, while the wall-clock times for the random baseline include only the target model's training time.

Our results show that with a $\frac{1}{32}$ selection fraction, the DQN-based approaches do not outperform the random baseline and take longer to run. However, for selection fractions $\frac{1}{16}$ and $\frac{1}{8}$, the DQN-based approaches outperform the random baseline, with an additional hour of training time. Although training on the full dataset yields the best performance, it requires more than twice the time needed for the DQN-based approaches with a $\frac{1}{8}$ selection fraction. Notably, while **`Transformer`**-based DQNs take slightly longer to train, they outperform **`MLP`**-based DQNs for the $\frac{1}{8}$ selection fraction. Overall, we conclude that DQN-based approaches provide a favorable balance between downstream performance and training time.

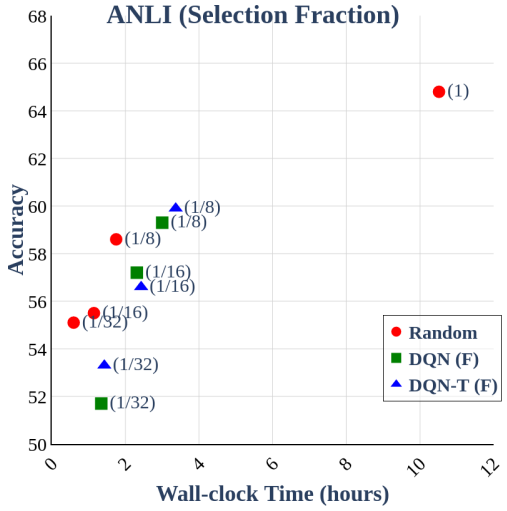


Figure 6. Downstream Performance vs Training Times for the Random and Full baselines, along with two DQN-based approaches. The selection fraction for each data point is enclosed in brackets.

6.6. Qualitative Analysis of Clusters

Reward signal diversity We visualize the diversity and reward spread of explored states across four datasets in Table 8. Emotion and MetaHate (b, d) exhibit high reward variance, correlating with the strong performance of RL agents on these tasks. In contrast, MMLU (c) shows moderate variance, and ANLI (a) has the lowest reward variability—corresponding to the largest performance gap with full-data training. These trends suggest that reward signal diversity may be predictive of success.

Label distribution analysis: To better understand the characteristics of the data subsets selected by our approaches across different datasets, we compare the label distribution in the original dataset with the subset selected using **`Binary-Mask`** state encodings and **`Furthest`** subsampling with the 600M proxy model, as shown in Figure 7. We use **`CLIMB`** for MetaHate since it performs substantially better and **`DQN-MLP`** for the rest.

For the ANLI dataset, we observe a partial rebalancing between the entailment and neutral labels following subset

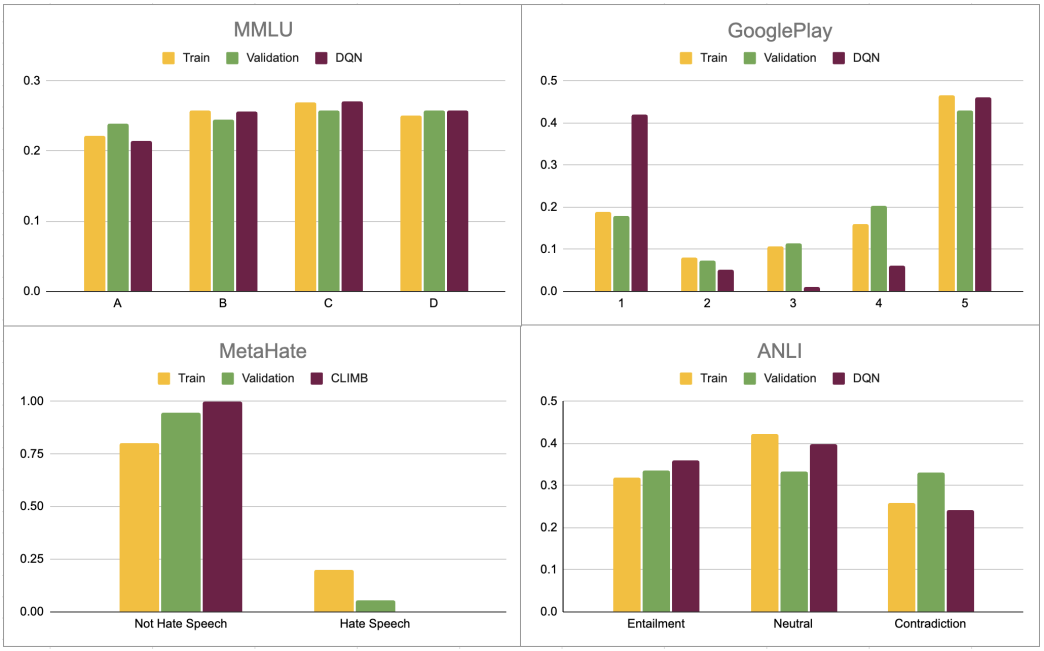
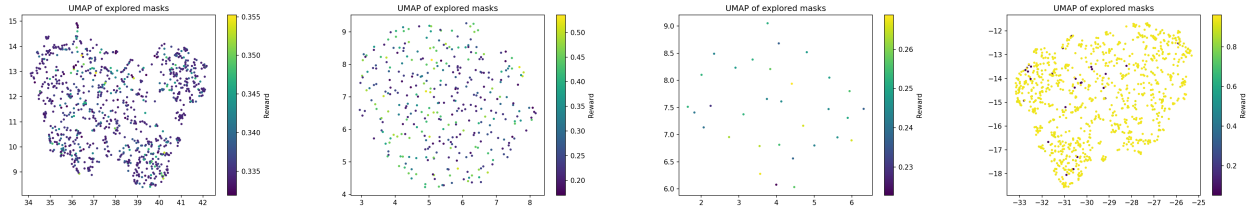


Figure 7. Label distribution of data subsets selected by the DQN against the full training dataset for all three datasets.



(a) ANLI (b) Emotion (c) MMLU (d) MetaHate

Figure 8. UMAP projections of explored cluster-selection masks during CLIMB-Diffusion sampling, colored by validation accuracy.

selection; however, the contradiction label remains notably underrepresented. A similar trend is seen in MMLU, where minor rebalancing occurs, but the overall label distribution remains largely unchanged. In the case of GooglePlay, the DQN-selected subset disproportionately oversamples label 1, reducing representation of the less frequent labels 2, 3, and 4. Since the selected subset outperforms the random baseline (which is expected to have the same label distribution as the full dataset) for these datasets, we hypothesize that the target model's performance may already be strong for the underrepresented labels, leading to their under-sampling during DQN-based selection. A comparable pattern is observed in the MetaHate dataset using **`CLIMB`**, where oversampling leads to significant increase in performance. Notably, the algorithm predominantly only selects examples labeled as "Not Hate Speech." We attribute this behavior to the highly skewed validation set, in which over 90% of the instances are labeled as "Not Hate Speech," thereby biasing the selection strategy.

7. Discussion

Limitations: The effectiveness of our approach depends critically on the ability of proxy models and the reward signal to accurately reflect the target model's response to selected training data. Our ablation studies further highlight the sensitivity of performance to hyperparameters, including type of state encoding, RL policy configuration and quality of validation set. This sensitivity necessitates careful tuning for the specific dataset-at-hand, which may not always be feasible in practical or resource-constrained environments. Additionally, the method's advantage is most pronounced in datasets with sufficient diversity and label balance; performance gains are diminished for highly homogeneous or low-variance datasets, as observed in the ANLI experiments.

Broader Impacts and Ethical Considerations: By intelligently selecting the most informative training examples, we can reduce computational costs, energy consumption, and carbon footprint while maintaining or enhancing model performance. In practice, our method democratize access to high-performance models and allows organization with limited compute to annotate a much smaller high-quality validation set to subsample a high-quality training set from raw data. However, automated data selection also introduces risks since it may propagate existing biases in the training set, potentially exacerbating fairness and equity concerns. Thus, it is critical that such systems be complemented by robust evaluation and interpretability tools, as well as ongoing audits for bias and representation.

Reflection: Through extensive experimentation, we found that RL methods not only outperform simple heuristics but, in some cases, even surpass full-data training by filtering out noisy or redundant examples. Nonetheless, negative results from several reward model variants (e.g., Random Network Distillation or diffusion-based sampling) emphasize that success is not uniform and depends on careful alignment between RL strategies, reward signals, and dataset characteristics. Given the high computational cost of experiments and number of hyper-parameters, proper tuning has been a major challenge, restricting the number of runs we could perform. Since there is not much literature on data selection for fine-tuning, we were severely limited on benchmarks and datasets to evaluate our methods and it took us substantial time to finalize our experiments.

8. Conclusion

This work demonstrates that data selection can be effectively formulated as a reinforcement learning problem, resulting in significant finetuning efficiency gains. Our experiments across ANLI, MetaHate, GooglePlay, and MMLU datasets show that models trained on fractions as small as 5% of the original data can achieve comparable or better performance than full dataset training while reducing wall-clock time from 10-12 hours to approximately 4 hours. Notably, we outperform full dataset training by **11.0 points** on Hate and **0.3 points** on Emotion. Different RL methods exhibit varying effectiveness across datasets. DQN and CLIMB variants performed consistently well, while PPO excelled with the MetaHate and MMLU datasets. Notably, our approach sometimes outperformed full dataset training, particularly with MetaHate, indicating effective filtering of noisy examples. The proxy model validation loss-based reward function proved to be both an effective and computationally efficient reward signal for guiding selection. These findings show promising directions for efficient LLM training in resource-constrained environments.

Future work could explore combining our RL methods with other efficiency techniques and investigating generalization to larger models and more diverse datasets. Our framework relies on discrete, binary selection of cluster subsets, extending the action space to continuous mixtures would enable more expressive and fine-grained selection policies. Task specific reward approximation remains an open problem. While we found proxy model-based validation loss to be a generally effective reward signal, different datasets and tasks may benefit from alternative reward formulations. A principled approach to selecting or learning the most suitable reward approximator per task could lead to more robust generalization across domains. Our current formulation is agnostic to fairness or distributional concerns, we can extend our framework by penalizing states that disproportionately exclude minority class clusters. The reward landscape over the state space is relatively smooth, small perturbations in the selected clusters often lead to small changes in reward. We can exploit this structure via algorithmic stability techniques (Hardt et al., 2016), which quantify how similar model outputs are when trained on neighboring datasets. Stability aware exploration may lead to more efficient search policies and tighter performance guarantees.

9. Contributions

This project is a continuation of Animesh and Harshit's CS234 Project which Rohan subsequently joined. The project involved substantial contributions from all authors building on top of previous work. Harshit worked on implementing MMLU, RND, MDP + random search, and performing analysis over different configurations and selected subsets. Animesh worked on implementing reward based models (DynaDQN, Climb and Climb Diffusion) and Neural Bandits, validation loss reward approximation, and new clustering methods. Rohan worked on adding uncertainity quanitifcation to existing methods such as CLimb Ensemble, he also identified binary mask state encoding. All three authors ran experiments and contributed writing portions of the report.

Changes from Proposal: Our hypothesis remains the same as proposal and we validate it using various methods. However, we did not end up experimenting with single player game formulations (Gaudel & Sebag, 2010) as we planned. Also, we tried various reward model based strategies like CLIMB that turned out to give good performance.

References

Abbas, A., Tirumala, K., Simig, D., Ganguli, S., and Morcos, A. S. Semdedup: Data-efficient learning at web-scale through semantic deduplication. *arXiv preprint arXiv:2303.09540*, 2023.

Albalak, A., Elazar, Y., Xie, S. M., Longpre, S., Lambert, N., Wang, X., Muennighoff, N., Hou, B., Pan, L., Jeong, H., Raffel, C., Chang, S., Hashimoto, T., and Wang, W. Y. A survey on data selection for language models. *arXiv preprint arXiv:2402.16827*, 2024. https://arxiv.org/abs/2402.16827.

at Meta, A. Mobile llm. URL https://huggingface.co/collections/facebook/mobilellm-6722be18cb86c20ebe113e95.

Austin, J., Johnson, D. D., Ho, J., Tarlow, D., and van den Berg, R. Structured denoising diffusion models in discrete state-spaces, 2023. URL https://arxiv.org/abs/2107.03006.

Burda, Y., Edwards, H., Storkey, A., and Klimov, O. Exploration by random network distillation, 2018. URL https://arxiv.org/abs/1810.12894.

Chandhok, S., Yang, Q., Manas, O., Jain, K., Sigal, L., and Agrawal, A. Learning what matters: Prioritized concept learning via relative error-driven sample selection, 2025. URL https://arxiv.org/abs/2506.01085.

Deb, R., Thekumparampil, K., Kalantari, K., Hiranandani, G., Sabach, S., and Kveton, B. Fishersft: Data-efficient supervised fine-tuning of language models using information gain, 2025. URL https://arxiv.org/abs/2505.14826.

Dhariwal, P. and Nichol, A. Diffusion models beat gans on image synthesis, 2021. URL https://arxiv.org/abs/2105.05233.

Diao, S., Yang, Y., Fu, Y., Dong, X., Su, D., Kliegl, M., Chen, Z., Belcak, P., Suhara, Y., Yin, H., Patwary, M., Yingyan, Lin, Kautz, J., and Molchanov, P. Climb: Clustering-based iterative data mixture bootstrapping for language model pre-training, 2025. URL https://arxiv.org/abs/2504.13161.

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

Gaudel, R. and Sebag, M. Feature selection as a one-player game. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pp. 359–366, Madison, WI, USA, 2010. Omnipress. ISBN 9781605589077.

Gu, Y., Dong, L., Wang, H., Hao, Y., Dong, Q., Wei, F., and Huang, M. Data selection via optimal control for language models. *ArXiv*, abs/2410.07064, 2024. URL https://api.semanticscholar.org/CorpusID:273228851.

Hardt, M., Recht, B., and Singer, Y. Train faster, generalize better: Stability of stochastic gradient descent, 2016. URL https://arxiv.org/abs/1509.01240.

Huang, S. and Ontañón, S. A closer look at invalid action masking in policy gradient algorithms. *The International FLAIRS Conference Proceedings*, 35, May 2022. ISSN 2334-0762. doi: 10.32473/flairs.v35i.130584. URL http://dx.doi.org/10.32473/flairs.v35i.130584.

Kocsis, L. and Szepesvári, C. Bandit based monte-carlo planning. In *Proceedings of the 17th European Conference on Machine Learning*, ECML'06, pp. 282–293, Berlin, Heidelberg, 2006. Springer-Verlag. ISBN 354045375X. doi: 10.1007/11871842_29. URL https://doi.org/10.1007/11871842_29.

Kolossov, G., Montanari, A., and Tandon, P. Towards a statistical theory of data selection under weak supervision. *arXiv preprint arXiv:2309.14563*, 2023.

Liu, Z., Ke, R., Liu, Y., Jiang, F., and Li, H. Take the essence and discard the dross: A rethinking on data selection for fine-tuning large language models, 2025. URL https://arxiv.org/abs/2406.14115.

Marion, M., Üstün, A., Pozzobon, L., Wang, A., Fadaee, M., and Hooker, S. When less is more: Investigating data pruning for pretraining llms at scale, 2023. URL https://arxiv.org/abs/2309.04564.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing atari with deep reinforcement learning, 2013. URL https://arxiv.org/abs/1312.5602.

Nikdan, M., Cohen-Addad, V., Alistarh, D., and Mirrokni, V. Efficient data selection at scale via influence distillation, 2025. URL https://arxiv.org/abs/2505.19051.

S, S. M., Hao, X., Hou, S., Lu, Y., Sevilla-Lara, L., Arnab, A., and Gowda, S. N. Progressive data dropout: An embarrassingly simple approach to faster training, 2025. URL https://arxiv.org/abs/2505.22342.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Sutton, R. S. Dyna, an integrated architecture for learning, planning, and reacting. *SIGART Bull.*, 2(4): 160–163, July 1991. ISSN 0163-5719. doi: 10.1145/122344.122377. URL https://doi.org/10.1145/122344.122377.

Tirumala, K., Simig, D., Aghajanyan, A., and Morcos, A. S. D4: Improving llm pretraining via document de-duplication and diversification, 2023. URL https://arxiv.org/abs/2308.12284.

Xia, M., Malladi, S., Gururangan, S., Arora, S., and Chen, D. Less: Selecting influential data for targeted instruction tuning. *arXiv preprint arXiv:2402.04333*, 2024.

Xie, S. M., Santurkar, S., Ma, T., and Liang, P. S. Data selection for language models via importance resampling. *Advances in Neural Information Processing Systems*, 36: 34201–34227, 2023.

Xie, S. M., Pham, H., Dong, X., Du, N., Liu, H., Lu, Y., Liang, P. S., Le, Q. V., Ma, T., and Yu, A. W. Doremi: Optimizing data mixtures speeds up language model pretraining. *Advances in Neural Information Processing Systems*, 36, 2024.

Yin, J. O. and Rush, A. M. Compute-constrained data selection, 2025. URL https://arxiv.org/abs/2410.16208.