# Data Mixing Agent: Learning to Re-weight Domains for Continual Pre-training

**Kailai Yang**[1*] **Xiao Liu**[2†] **Lei Ji**[2] **Hao Li**[3] **Xiao Liang**[4] **Zhiwei Liu**[1]
**Yeyun Gong**[2†] **Peng Cheng**[2] **Mao Yang**[2]
[1] The University of Manchester
[2] Microsoft Research
[3] Imperial College London
[4] University of California, Los Angeles

## Abstract

Continual pre-training on small-scale task-specific data is an effective method for improving large language models in new target fields, yet it risks catastrophic forgetting of their original capabilities. A common solution is to re-weight training data mixtures from source and target fields on a domain space to achieve balanced performance. Previous domain reweighting strategies rely on manual designation with certain heuristics based on human intuition or empirical results. In this work, we prove that more general heuristics can be parameterized by proposing **Data Mixing Agent**, the first model-based, end-to-end framework that learns to re-weight domains. The agent learns generalizable heuristics through reinforcement learning on large quantities of data mixing trajectories with corresponding feedback from an evaluation environment. Experiments in continual pre-training on math reasoning show that Data Mixing Agent outperforms strong baselines in achieving balanced performance across source and target field benchmarks. Furthermore, it generalizes well across unseen source fields, target models, and domain spaces without retraining. Direct application to the code generation field also indicates its adaptability across target domains. Further analysis shows the agents' well-aligned heuristics with human intuitions and efficiency in achieving superior model performance with less source-field data.

## 1 Introduction

Large Language Models (LLMs) (Yang et al., 2025; Liu et al., 2024a), though pre-trained to obtain generalization capabilities, often require further enhancement in knowledge-intensive fields (Yang et al., 2024; Guo et al., 2024) via continual pre-training in the target field. However, directly adapting to the target field data can lead to catastrophic forgetting of source data and collapse on existing model capabilities (Hui et al., 2024; Lin et al., 2025), due to the significant distribution shift between source and target fields.

A popular solution is to curate data mixtures of the source and target fields to achieve a balanced performance (Shi et al., 2024). Existing methods mainly organize data mixtures defined by meta-attributes such as data sources and focus, known as domains (Du et al., 2022; Luo et al., 2024). During training, the data mixture is allocated through a distribution in the domain space, which reflects the ratio of data allocated from each domain. The distribution can be adjusted after several training steps if necessary, leading to a data mixing trajectory (Luo et al., 2024; Xia et al., 2023) along the domain reweighting steps. Data mixing trajectories significantly influence model performance (OLMo et al., 2024; Grattafiori et al., 2024; Li et al., 2024), and previous works have explored data mixing algorithms to determine optimal trajectories for different tasks (Liu et al., 2024b; Ye et al., 2024; Xie et al., 2023; Xia et al., 2023; Luo et al., 2024).

A commonality of these data mixing methods is that their designs are based on certain general heuristics, such as: *data mixtures that provide balanced evaluation loss lead to desired downstream performance*. Another indication of these heuristics is the various empirical conclusions drawn from training practices. For example, Wettig et al. (2025) concluded that *Data from the 'Science' domain heavily promote model performance on MMLU (Hendrycks et al., 2020), while the 'Home' domain improves HellaSwag (Zellers et al., 2019) performance*. In Fig. 1, we provide an average of distributions along 20 randomly generated data mixing trajectories (each with 80 domain reweighting steps), separated into four categories. The categories are defined by whether they increase or decrease the target model's performance on the MMLU/MATH (Hendrycks et al., 2021) bench-

[*] Work done during his internship at Microsoft Research.
[†] Corresponding authors.



*Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9447–9473

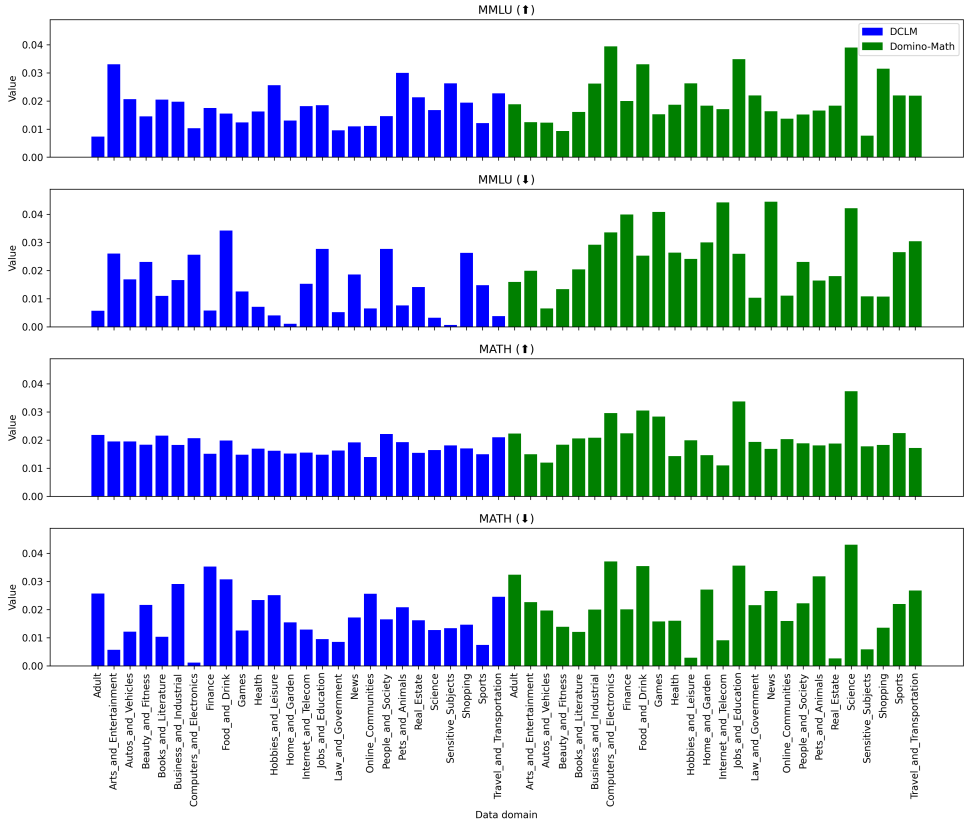


Figure 1: Four averaged distributions drawn from 20 randomly generated data mixing trajectories. Each distribution in the trajectories is first categorized by whether it increases/decreases the performance of a 50M target model on the MMLU or MATH benchmarks within one re-weighting step, leading to four categories. The distributions in each category are then averaged within the domain space to represent their features, as shown in the figure. The models are trained on a 52-dimensional space, mixing the general domain data from DCLM (Li et al., 2024) and the math reasoning data from the math split of the Dolmino-mix-1124 (OLMo et al., 2024) dataset.

marks. The data mixing domains are defined and classified by the Nvidia domain classifier [1]. According to the results, there are explicit differences between data distributions that increase/decrease model performance. For example, in MMLU (the first and second rows), higher ratios of DCLM data from the *Science* and *Home&Garden* domains significantly improves benchmark performance. In MATH, increasing data mixtures from the *Hobbies&Leisure* and *Real estate* domains of the Dolmino-math data while keeping a balanced mix of DCLM data is likely to boost benchmark performance. The above results unveil more general heuristics that can enhance model performance in continual pretraining, yet have not been discovered or utilized by existing works. This negligence motivates further efforts to efficiently mine such heuristics and leverage them for data mixing in different scenarios.

These observations reveal a rich heuristic space for domain reweighting. We believe these model- and data-agnostic heuristics can be unified into a small agent model to guide the data mixing trajectories in an end-to-end manner. Based on this intuition, we propose **Data Mixing Agent**, the first model-based method that learns to re-weight domains for continual pre-training. We start by randomly sampling large quantities of data mixing trajectories, each with fixed domain re-weighting steps. We then train small proxy models on all trajectories, obtaining model checkpoints on each re-weighting step. All checkpoints are evaluated in a light-weight evaluation environment to assess the target capabilities. The trajectories and corresponding environment feedback are expected to empirically enclose a wide range of heuristics. Data Mixing Agent is then optimized on these collected data with the Conservative Q-Learning (CQL) reinforcement learning algorithm (Kumar et al., 2020). During continual pre-training on the target model,

[1] https://huggingface.co/nvidia/domain-classifier

the agent directly predicts the domain distribution for the next domain reweighting step on the fly, considering previous states in the data mixing trajectory and the environment feedback.

We apply Data Mixing Agent on the math reasoning target field while preserving performance in the general field. Evaluation on in-distribution source field data shows that our method significantly outperforms state-of-the-art static (Liu et al., 2024b) and dynamic (Xia et al., 2023) domain reweighting methods, achieving an average improvement of 3.02% across 8 general benchmarks and 4 math reasoning benchmarks. The agent's generalization ability is demonstrated with balanced performance across 3 unseen source fields, 4 target models, and 2 domain spaces, all without retraining. We also directly apply agents trained on math reasoning to guide training on the unseen code generation field, proving their generalization across target domains. Additional analysis confirms that these heuristics align well with human intuitions, and Data Mixing Agent can achieve superior continual pre-training performance with less source-field data.

In summary, we make the following contributions:

- We propose Data Mixing Agent, the first end-to-end, lightweight domain reweighting method for continual pre-training;
- Data Mixing Agent significantly outperforms state-of-the-art baselines in extensive experiments, leading to balanced performance across model capabilities and high efficiency in data usage;
- Data Mixing Agent learns heuristics that generalize across source and target fields, target models, and domain spaces.

## 2 Domain Re-weighting as MDP

In this section, we formally state domain reweighting as a Markov Decision Process (MDP), defined as a tuple $(\mathcal{S}, \mathcal{A}, f, r, \rho_s, \rho_e)$. We describe each element as follows:

**State Space** The state space $\mathcal{S}$ is a continuous space consisting of all data distributions from previous domain reweighting steps. Specifically, for step $t$, the state $s_t \in \mathcal{S}$ has dimension $s_t \in \mathbb{R}^{t \times N}$, where $N$ is the dimension of the action space, determined by the definition of the domains.

**Action Space** The action space $\mathcal{A}$ is a continuous space denoting the data distribution in the current domain reweighting step. At step $t$, the action $a_t \in \mathcal{A}$ is a *probability distribution* over the domain space: $a_t \in \mathbb{R}^N$ and $\sum_{i=1}^{N} a_t^i = 1, a_t^i \geq 0$.

**Policy and Reward** The policy function $f$ denotes Data Mixing Agent that determines the action at the current step. The feedback is modeled by the reward function $r$, determined by the target fields and the environment. With domain re-weighting modeled as an MDP, the policy function can be optimized via reinforcement learning.

**Start and Terminate State** The start state $\rho_s$ depends on the target model, reflecting the distribution of its pre-training data. The terminate state $\rho_e$ depends on manual setting or data scale, as the training process can often end with predefined token budgets or the exhaustion of target field data.

## 3 Data Mixing Agent

In this section, we introduce the methodology of Data Mixing Agent. An overview of the pipeline is shown in Fig. 2. **The full implementation details of all procedures are described in Appn. C.**

### 3.1 Trajectory Sampling

**Action Space Definition** We start by defining the action space $\mathcal{A}$, which is essential for trajectory sampling. While most methods define the space via data sources (Xia et al., 2023; Luo et al., 2024; OLMo et al., 2024), recent work has emphasized the drawbacks of data overlap across domains (Xi et al., 2025) and the unstructured nature (Wettig et al., 2025) of source-based data clustering. Inspired by Wettig et al. (2025), we construct domains with the Nvidia domain classifier, which classifies the data from the source and target fields, each into 26 domains, leading to a 54-dimensional data distribution space. The definition of the 26 domains are shown in Fig. 1.

**Start State Estimation** The start state $\rho_s$ can easily be determined when data from the source field are available. We randomly sample 1B tokens from the training data and utilize the same domain classifier to organize the data into the defined domains. The start state is estimated as a normalization over sample numbers in each domain. When data from the source field is unavailable (Grattafiori et al., 2024; Liu et al., 2024a), we explore using synthetic data from the target model to estimate the start state.

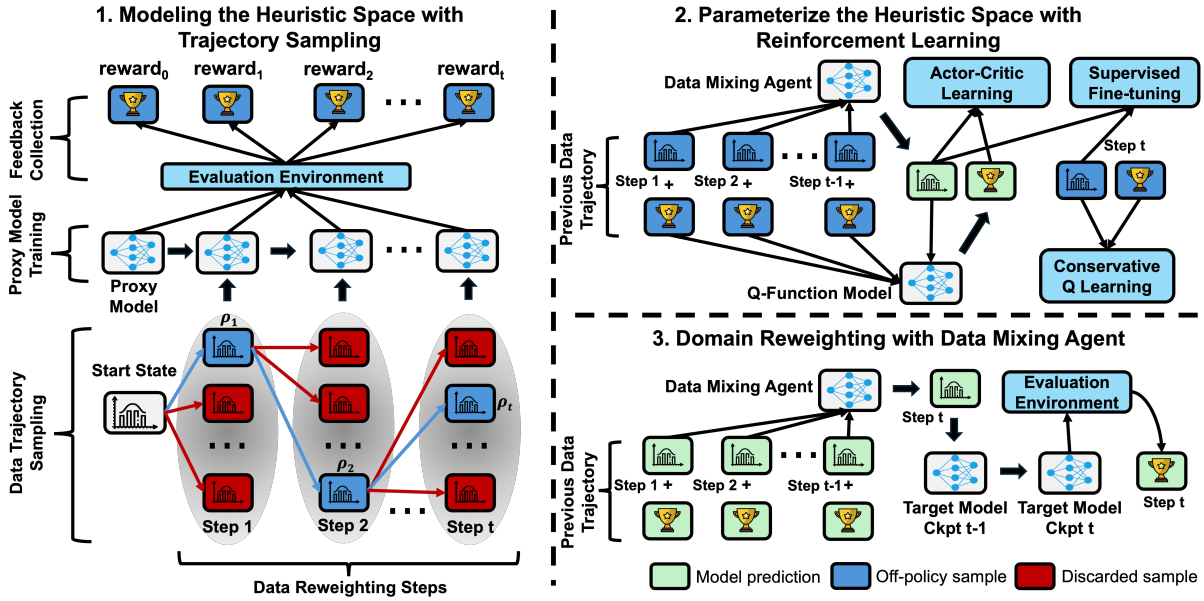


Figure 2: An overview of Data Mixing Agent. We first sample data mixing trajectories and train small proxy models on them. Each model checkpoint obtains feedback from the evaluation environment. Secondly, the agent is optimized on these trajectories and feedback via CQL. When guiding continual pretraining, the agent determines the distribution for the next step on the fly.

We experiment on five Pythia models (Biderman et al., 2023) by estimating the distributions from their generated data and calculating their KL divergence with the ground-truth distribution. The results prove the effectiveness of using random samples from the target model as estimates for their start states. More details about the experiments and results are shown in Appn. B.

**Data Mixing Trajectory Sampling** We randomly sample data mixing trajectories as training data for modeling the heuristic space. The random sampling process is based on the following principle:

> The data mixing trajectories should be well-distributed across the action space, ensuring coverage of actions that enhance and degrade model performance.

To ensure this principle, we design an inductive scoring algorithm to rate each sampled distribution, which is designed based on the following inductive biases: 1) The distribution at the current step should not deviate significantly from that of the previous step; 2) The distribution at each re-weighting step should align more closely with the target distribution; 3) The distribution at the current step should differ from sampled trajectories to encourage diversity; 4) The target distribution encourages reducing reliance on source-field data. During implementation, we use random tokens from the DCLM (Li et al., 2024) as the source field data and the math split of the Dolmino-mix-1124 (OLMo et al., 2024) dataset as the target field data, obtaining 384 trajectories with various qualities. This algorithm is formally described in Appn. C.1.

**Environment Design and Feedback Collection** The evaluation environment is curated to assess model checkpoints. It is lightweight yet accurately reflects target capabilities, providing effective supervision while minimizing computational overhead. Specifically, we select a small high-quality evaluation set $\mathcal{D}_i$ that well represents the $i$-th target field: $\{q_j, r_j\}_{j=1}^{|\mathcal{D}_i|}$. For the model checkpoint $\mathcal{M}$, we compute the average per-token log probability on all question-answer pairs to reflect model performance on the $i$-th target field:

$$\mathcal{S}(\mathcal{M}, \mathcal{D}_i) = \frac{1}{|\mathcal{D}_i|} \sum_{(q_j, r_j) \in \mathcal{D}_i} \frac{1}{|r_j|} \log P_{\mathcal{M}}(r_j \mid q_j) \quad (1)$$

The final environment feedback returns a vector-style assessment for model $\mathcal{M}$:

$$\mathcal{R}(\mathcal{M}) = \left[\mathcal{S}(\mathcal{M}, \mathcal{D}_1), \mathcal{S}(\mathcal{M}, \mathcal{D}_2), ..., \mathcal{S}(\mathcal{M}, \mathcal{D}_{|\mathcal{D}|})\right] \quad (2)$$

During implementation, the environment assesses the general capability via the validation set of the MMLU (Hendrycks et al., 2020) dataset and

the math reasoning capability via a subset of the MATH (Hendrycks et al., 2021) dataset. Leveraging this evaluation environment, we collect feedback data by training a small proxy model $\mathcal{M}_p$ on each sampled trajectory from scratch. The model checkpoint is evaluated on this environment at each data reweighting step. For the $i$-th data mixing distribution $\rho_i \in \tau$, we obtain a tuple $(\rho_i, \mathcal{R}(\mathcal{M}_p^i))$, where $\mathcal{M}_p^i$ denotes the model checkpoint after training on the $i$-th step.

### 3.2 Modeling the Heuristic Space with RL

We expect the sampled data mixing trajectories and the feedback to well represent the heuristic space for domain reweighting. We parameterize these heuristics by training an agent model on these trajectories in a reinforcement learning-based paradigm.

**Agent Model Structure** We determine the model structure for the data mixing agent with the following principles: 1) effectively model temporal sequences and support long-range interactions between distributions; 2) be lightweight to prevent unacceptable latency and computation overhead.

We utilize the Transformer (Vaswani et al., 2017) decoder architecture, which is widely used in time series forecasting (Zhang et al., 2024; Li et al., 2025) and facilitates long-range interactions between data points with dot-product attention. We stack two Transformer layers followed by a linear layer and Softmax to project the representations into the action space, with merely 2.1M parameters. Formally, at data reweighting step $t$, the agent $f$ predicts the domain distribution with the previous trajectory and environment feedback as follows:

$$\begin{aligned} \rho_t &= f(\tilde{\rho}_0, \tilde{\rho}_1, ..., \tilde{\rho}_{t-1}) \\ \tilde{\rho}_i &= [\rho_i; \mathcal{R}(\mathcal{M}_i)], i = 0, ..., t-1 \end{aligned} \tag{3}$$

where ; denotes concatenation, $\tilde{\rho}_i \in \mathbb{R}^{N+|\mathcal{D}|}$ denotes the input feature in the data reweighting step $i$, and $\rho_t$ denotes the agent's output action at step $t$.

**Off-policy Optimization with Conservative Q-Learning** We first perform a warm-up on the randomly initialized agent model with Supervised Fine-Tuning (SFT) to reduce the later parameter searching space. Details about the SFT process is shown in Appn. C.3. Based on the warmed-up model, we parameterize the heuristic space via reinforcement learning, where the algorithm is selected based on the following two principles: 1) The training process is offline and off-policy; 2) The agent's actions are sampled from a continuous domain space.

We select Conservative Q-Learning (CQL) (Kumar et al., 2020) as the optimization algorithm. CQL prevents overestimation of Q-values for out-of-distribution actions by introducing a conservative penalty for the Q-function optimization process. The agent model is trained in an actor-critic (Sutton et al., 1999) structure until convergence, where the agent acts as the actor model, and the critic model (Q-function) is initialized from scratch with another neural network. Details about the CQL training process are shown in Appn. C.4.

### 3.3 Domain Reweighting with Data Mixing Agent

The system pipeline of continual pretraining with Data Mixing Agent is described in Algorithm 2. The agent directly determines the distribution for the next domain re-weighting step on the fly, considering the previous states in the data mixing trajectory and the corresponding environment feedback. This MDP continues until the target data is fully leveraged or a predetermined computation budget is reached. We expect the agent to optimally curate the training recipe by balancing performance across all target fields while minimizing the use of source-field data tokens to reduce computational cost. **We also expect the agent's learned heuristics to generalize to unseen target models, data mixtures, and even target domains.** This generalization is crucial to avoid repeated trajectory sampling and agent retraining when adapting to new continual pre-training scenarios, thereby significantly reducing overall computational cost.

## 4 Experiments

### 4.1 Experimental Settings

**Full justifications for implementation and experimental settings are described in Appn. D.**

**Target Models** Since most open-source LLMs have already been optimized on math reasoning or code generation, we pre-train 3 models from scratch, with the same LLaMA model architecture (Grattafiori et al., 2024) with 3B parameters, on 100B random tokens from the DCLM (Li et al., 2024), Fineweb-Edu (Penedo et al., 2024), and Nemotron-CC (Su et al., 2024) dataset, resulting in 3 target models: **LLaMA-DCLM**, **LLaMA-FWE**,

| Method | GPU Hrs | Avg.↑ | Var.↓ | General Benchmarks | | | | | | | | | Math Benchmarks | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | MMLU | Hella. | OBQA | Wino. | ARC-C | PiQA | SciQ | LogiQA | Avg. | GSM8K | Minerva | MATH | MathQA | Avg. |
| **LLaMA-DCLM** | | | | | | | | | | | | | | | | | |
| Base Model | – | 38.15 | 780.62 | 34.5 | **64.5** | 37.0 | 61.56 | 36.69 | **75.84** | 84.2 | 28.11 | 52.8 | 2.55 | 4.1 | 4.22 | 24.52 | 8.85 |
| Naive Training | 874.88 | 38.51 | – | 27.11 | 37.0 | 28.2 | 54.22 | 28.58 | 60.28 | 68.7 | 26.88 | 41.37 | 59.21 | **16.16** | **22.85** | 32.96 | 32.80 |
| RegMix | 1987.84 | 44.01 | 559.83 | 30.42 | 59.72 | 36.6 | 61.72 | 34.73 | 73.88 | 85.1 | 28.73 | 51.36 | 55.87 | 11.5 | 17.7 | 32.16 | 29.31 |
| DBL | 1906.8 | 43.50 | 575.91 | 29.0 | 55.66 | 35.0 | **63.64** | 32.11 | 72.5 | **88.42** | 28.89 | 50.65 | 56.22 | 12.2 | 18.24 | 30.14 | 29.2 |
| DataAgent$_{SFT}$ | 1880.32 | 44.95 | 563.27 | 33.81 | 60.23 | 34.2 | 60.43 | 36.26 | 73.28 | 87.3 | 29.33 | 51.86 | 57.84 | 12.7 | 21.3 | 32.73 | 31.14 |
| DataAgent$_{RL}$ | 1891.84 | **47.03** | **547.66** | **34.06** | 63.38 | **42.14** | 62.35 | **36.92** | 74.85 | 87.89 | **30.29** | **54.04** | **59.24** | 14.8 | 22.75 | **35.3** | **33.02** |
| **LLaMA-FWE** | | | | | | | | | | | | | | | | | |
| Base Model | – | 37.65 | 781.34 | 34.47 | 60.52 | 37.8 | 57.77 | 40.53 | 74.21 | 85.4 | **28.26** | 52.37 | 2.5 | 2.52 | 4.06 | 23.72 | 8.2 |
| Naive Training | 852.1 | 38.51 | – | 27.25 | 37.03 | 28.4 | 53.51 | 26.96 | 61.1 | 69.6 | 28.11 | 41.5 | **58.91** | 12.58 | **24.7** | **33.94** | **32.53** |
| RegMix | 1948.26 | 43.83 | 557.18 | 32.83 | 60.2 | 36.45 | 54.51 | 37.17 | 71.72 | 86.5 | 28.0 | 50.92 | 55.9 | 12.2 | 20.35 | 30.1 | 29.64 |
| DBL | 1883.58 | 43.92 | 566.47 | 31.27 | 56.58 | **40.7** | 56.27 | 38.32 | 71.18 | **88.26** | 26.0 | 51.07 | 54.2 | 12.05 | 20.72 | 31.52 | 29.62 |
| DataAgent$_{SFT}$ | 1887.82 | 45.23 | 552.61 | **34.65** | **60.83** | 38.28 | 59.3 | **40.8** | **74.6** | 85.6 | 26.96 | **52.63** | 56.26 | 12.19 | 21.92 | 31.32 | 30.42 |
| DataAgent$_{RL}$ | 1838.2 | **45.48** | **540.83** | 33.78 | 60.44 | 38.8 | **59.59** | 38.89 | 73.12 | 84.9 | 27.49 | 52.13 | 58.07 | **13.46** | 23.96 | 33.28 | 32.19 |
| **LLaMA-Nemo** | | | | | | | | | | | | | | | | | |
| Base Model | – | 38.22 | 782.95 | 34.22 | 64.51 | 37.6 | 59.12 | 36.26 | **75.57** | **88.4** | 26.73 | 52.8 | 2.5 | 4.85 | 5.3 | 23.62 | 9.07 |
| Naive Training | 870.4 | 37.86 | – | 27.06 | 37.4 | 28.0 | 52.88 | 27.05 | 59.74 | 68.6 | 25.19 | 40.74 | **59.05** | **12.3** | **24.52** | **32.53** | **32.1** |
| RegMix | 1925.23 | 44.13 | 546.24 | **35.03** | **65.15** | 35.9 | 59.83 | 36.45 | 72.85 | 86.2 | 28.88 | 52.54 | 49.39 | 10.7 | 19.23 | 30.0 | 27.33 |
| DBL | 1920.21 | 42.63 | 522.88 | 33.2 | 64.82 | 34.0 | **62.06** | **39.23** | 70.79 | 78.11 | 24.0 | 50.77 | 45.91 | 10.74 | 20.01 | 28.74 | 26.35 |
| DataAgent$_{SFT}$ | 1887.25 | 44.12 | 538.44 | 34.06 | 64.25 | 39.04 | 60.4 | 38.1 | 74.17 | 86.75 | 29.16 | 53.24 | 47.81 | 9.07 | 17.8 | 28.86 | 25.89 |
| DataAgent$_{RL}$ | 1812.31 | **45.8** | **520.71** | 34.27 | 63.95 | **39.8** | 61.58 | 38.74 | 74.49 | 86.9 | **29.95** | **53.71** | 54.28 | 10.83 | 22.94 | 31.85 | 29.98 |

(a) Model performances on the 2-dimensional data reweighting space based on data sources.

| Method | GPU Hrs | Avg.↑ | Var.↓ | General Benchmarks | | | | | | | | | Math Benchmarks | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | MMLU | Hella. | OBQA | Wino. | ARC-C | PiQA | SciQ | LogiQA | Avg. | GSM8K | Minerva | MATH | MathQA | Avg. |
| **LLaMA-DCLM** | | | | | | | | | | | | | | | | | |
| Base Model | – | 38.15 | 780.62 | **34.5** | **64.5** | 37.0 | 61.56 | 36.69 | **75.84** | 84.2 | 28.11 | 52.8 | 2.55 | 4.1 | 4.22 | 24.52 | 8.85 |
| Naive Training | 874.88 | 38.51 | – | 27.11 | 37.0 | 28.2 | 54.22 | 28.58 | 60.28 | 68.7 | 26.88 | 41.37 | **59.21** | 16.16 | 22.85 | **32.96** | **32.80** |
| RegMix | 2306.14 | 44.67 | 571.39 | 34.38 | 62.17 | 38.2 | 61.93 | **36.95** | 74.97 | 87.5 | 29.49 | 53.19 | 55.78 | 10.36 | 15.75 | 28.67 | 27.64 |
| DataAgent$_{SFT}$ | 2174.26 | 45.75 | 560.74 | 34.47 | 63.36 | 40.6 | 62.35 | 35.87 | 74.32 | **89.2** | **29.96** | 53.77 | 56.77 | 11.12 | 18.76 | 32.3 | 29.74 |
| DataAgent$_{RL}$ | 2202.88 | **46.84** | **552.11** | 32.99 | 62.64 | **41.6** | **63.98** | 36.64 | 73.5 | 89.1 | 31.5 | **53.99** | 59.04 | **16.48** | **22.9** | 31.72 | 32.54 |
| **LLaMA-FWE** | | | | | | | | | | | | | | | | | |
| Base Model | – | 37.65 | 781.34 | 34.47 | 60.52 | 37.8 | 57.77 | **40.53** | 74.21 | 85.4 | 28.26 | **52.37** | 2.5 | 2.52 | 4.06 | 23.72 | 8.2 |
| Naive Training | 852.1 | 38.51 | – | 27.25 | 37.03 | 28.4 | 53.51 | 26.96 | 61.1 | 69.6 | 28.11 | 41.5 | 58.91 | 12.58 | **24.7** | **33.94** | 32.53 |
| RegMix | 2276.97 | 43.78 | 553.76 | **35.64** | 57.64 | **39.4** | 57.56 | 38.2 | 67.4 | 85.03 | 29.34 | 51.28 | 53.68 | 10.84 | 20.35 | 30.0 | 28.72 |
| DataAgent$_{SFT}$ | 2102.56 | 45.21 | 541.08 | 34.27 | 61.2 | 37.95 | 58.57 | 40.28 | **75.17** | 85.8 | 28.23 | 52.68 | 54.5 | 13.27 | 22.17 | 31.1 | 30.26 |
| DataAgent$_{RL}$ | 2162.57 | **45.55** | **532.47** | 32.55 | 58.64 | 35.8 | **59.42** | 39.76 | 73.34 | **87.2** | **29.35** | 52.01 | **58.96** | **14.68** | 23.62 | 33.32 | **32.65** |
| **LLaMA-Nemo** | | | | | | | | | | | | | | | | | |
| Base Model | – | 38.22 | 782.95 | 34.22 | 64.51 | 37.6 | 59.12 | 36.26 | **75.57** | 88.4 | 26.73 | 52.8 | 2.5 | 4.85 | 5.3 | 23.62 | 9.07 |
| Naive Training | 870.4 | 37.86 | – | 27.06 | 37.4 | 28.0 | 52.88 | 27.05 | 59.74 | 68.6 | 25.19 | 40.74 | **59.05** | **12.3** | **24.52** | **32.53** | **32.1** |
| RegMix | 2276.17 | 44.86 | 559.63 | **35.68** | **66.4** | 41.4 | **61.56** | 35.82 | 72.4 | **87.53** | 29.34 | **53.77** | 48.17 | 10.91 | 19.03 | 30.04 | 27.04 |
| DataAgent$_{SFT}$ | 2144.15 | 45.16 | 544.92 | 34.8 | 64.01 | 38.2 | 60.73 | 36.9 | 73.92 | 87.3 | **29.37** | 53.15 | 52.5 | 10.7 | 20.77 | 32.71 | 29.17 |
| DataAgent$_{RL}$ | 2160.44 | **45.9** | **536.18** | 33.89 | 62.7 | **42.6** | 59.82 | **40.0** | 74.86 | 84.34 | 28.29 | 53.31 | 56.78 | 11.82 | 23.61 | 32.03 | 31.06 |

(b) Model performances on the 52-dimensional data reweighting space based on the Nvidia domain classifier.

Table 1: Results of math reasoning on 12 benchmarks. We apply DBL only on the 2D domain space, due to the lack of evaluation sets for all domains in the 52D space. "GPU Hrs" denotes training time measured in GPU hours. "Avg." and "Var." denote the average and variance of each method on all benchmarks.

and **LLaMA-Nemo**. We also include the **Pythia-1.4B** model (Biderman et al., 2023) to evaluate performance on existing open-source models.

**Baseline Methods** We compare Data Mixing Agent (**DataAgent**$_{RL}$) with the following baseline methods: **Base Model**: direct evaluation of the target models; **Naive Training**: continual training solely on target field data; **RegMix** (Liu et al., 2024b): trains small proxy models on multiple distributions and fit a regression model to determine the optimal recipe; **Dynamic Batch Loading (DBL)** (Xia et al., 2023): dynamic domain reweighting based on excess losses across different domains; **DataAgent**$_{SFT}$: the data mixing agent model without the reinforcement learning process. DBL can only be applied to the data source-based experiments since it requires an evaluation set for each domain.

**Target Model Training Data** For source field data, the pretrained LLaMA models utilize their corresponding pre-training data, each with 100B tokens. We use synthetic data from the Pythia-1.4B with quality filters as the source field data for itself. For math reasoning, we select the math split (10B tokens) of the Dolmino-mix-1124 (OLMo et al., 2024). For code generation, we select the GitHub training split of SlimPajama-DC (Shen et al., 2023) with 30B tokens.

**Evaluation Benchmarks** We evaluate general capabilities on the MMLU (Hendrycks et al., 2020), HellaSwag (Zellers et al., 2019) (Hella.), OpenBookQA (Mihaylov et al., 2018) (OBQA), Winogrande (Sakaguchi et al., 2021) (Wino.), ARC-Challenge (Clark et al., 2018) (ARC-C),

PiQA (Bisk et al., 2020), SciQ (Welbl et al., 2017), and LogiQA (Liu et al., 2020) benchmarks. We evaluate math reasoning capabilities on the GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021), Minerva (Lewkowycz et al., 2022), and MathQA (Amini et al., 2019) benchmarks. We evaluate code generation capabilities on the HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021) benchmarks.

**Target Model Training Setting** The agent is trained on a 52-dimensional domain reweighting space. To evaluate in different domain spaces, we further employ the agent on the data source-based action space (source and target). We use the GitHub validation split of the SlimPajama-DC dataset when building the environment for code generation.

### 4.2 Evaluation Results on Math Reasoning

Evaluation results on the math reasoning target field are shown in Table 1. We have the following observations:

**Data Mixing Agent significantly outperforms other methods in balanced performance across fields.** In Table 1a, for the LLaMA-DCLM target model, DataAgent$_{RL}$ achieves the best average performance 54.04% and 33.02% on general/math benchmarks, outperforming the base model in general ability and the naively trained model on math reasoning. Overall, DataAgent$_{RL}$ achieves 47.03% on average, surpassing RegMix by 3.02%, DBL by 3.53%, and the base model by 8.88%. DataAgent$_{RL}$ also outperforms DataAgent$_{SFT}$ by a large margin of 2.08%, proving reinforcement learning with CQL as a crucial step for effective heuristics compared to the inductive biases in the SFT stage. All domain reweighting methods significantly reduce performance variance by over 200 compared to the base model. Data mixing agent further achieves the best results in variance, further demonstrating its advantage in stable performance across fields.

**The capabilities of data mixing agents can generalize across target models, source-field data, and domain spaces without retraining.** Though the agent is trained on the 52-dimensional data reweighting space with trajectories sampled with the DCLM data, it effectively guides domain reweighting for three target models across 2 domain definitions. On the 2-dimensional domain space (Table 1a), DataAgent$_{RL}$ outperforms DBL by an average of 2.37% on unseen target models LLaMA-FWE and LLaMA-Nemo. On the 52-dimensional space (Table 1b), DataAgent$_{RL}$ outperforms RegMix by an average of 1.41%. These results reflect data- and model-agnostic heuristics, enabling the agent to generalize well without retraining.

Full results and analysis on the Pythia-1.4B model and synthetic source field data are presented in Appn. E.1.

### 4.3 Evaluation Results on Code Generation

We directly utilize the agent trained on math reasoning to guide domain reweighting for code generation. The results are shown in Table 2. We have the following observations:

**Data Mixing Agent can generalize across target fields without retraining.** DataAgent$_{RL}$ achieves the best average performance of 46.3%, significantly outperforming RegMix by 1.45% and DBL by 2.78%, which are directly optimized on code generation. It shows that heuristics can be directly transferred without modifying the weights of the agent. In code generation, DataAgent$_{RL}$ outperforms naive training by 6.22%, while in Table 3a, the advantage is 8.52%, mainly due to the fact that applying DataAgent$_{RL}$ to code generation leads to worse performance on general benchmarks compared to math reasoning, which indicates the existence of heuristics that are dependent on the target field and the potential misalignment when converting them to a new target field.

Full results and analysis on the Pythia-1.4B model and synthetic source field data are presented in Appn. E.2.

### 4.4 Comparisons on Training Efficiency

In Table 1 and 2, we provide the training time of the main experiments measured in GPU hours to compare their efficiency.

**Data Mixing Agent requires fewer GPU hours for training than baseline methods, demonstrating higher efficiency despite the addition of agent models.** For instance, DataAgent$_{RL}$ saves an average of 106.33 GPU hours compared to RegMix in math reasoning, and 264.29 GPU hours in code generation. This efficiency primarily stems from two factors: (1) the agent model is extremely lightweight, adding negligible computational overhead and latency during inference; and (2) it is trained to reduce dependence on source-field data, thereby lowering the overall token budget by incorporating less source-field data. Further discussion

| Method | GPU Hrs | Avg. | General Benchmarks | | | | | | | | | Code Benchmarks | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | MMLU | Hella. | OBQA | Wino. | ARC-C | PiQA | SciQ | LogiQA | Avg. | HumanEval | MBPP | Avg. |
| Base Model | – | 44.52 | **34.5** | **64.5** | 37.0 | **61.56** | **36.69** | 75.84 | **84.2** | 28.11 | **52.8** | 8.6 | 14.2 | 11.4 |
| Naive Training | 3390.77 | 40.08 | 27.6 | 42.96 | 29.37 | 53.1 | 24.76 | 70.5 | 62.95 | 24.46 | 41.96 | **27.3** | **37.8** | **32.55** |
| RegMix | 5726.11 | 44.85 | 31.22 | 57.13 | 33.4 | 57.43 | 29.05 | **76.1** | 82.09 | 29.33 | 49.47 | 21.1 | 31.6 | 26.35 |
| DBL | 5480.74 | 43.52 | 31.7 | 55.7 | 35.2 | 53.8 | 31.28 | 68.58 | 78.26 | **29.8** | 48.04 | 20.6 | 30.3 | 25.45 |
| DataAgent$_{SFT}$ | 5318.37 | 45.07 | 32.69 | 63.43 | 35.0 | 49.88 | 33.46 | 72.85 | 78.66 | 29.61 | 49.45 | 22.4 | 32.8 | 27.6 |
| DataAgent$_{RL}$ | 5461.82 | **46.3** | 33.84 | 63.79 | **37.8** | 57.3 | 35.05 | 73.2 | 78.57 | 27.34 | 50.86 | 22.0 | 34.1 | 28.05 |

Table 2: Evaluation results on the code generation target field, reflected on 10 benchmarks. The data is reweighted on the 2-dimensional data source-based domain space and the LLaMA-DCLM target model.

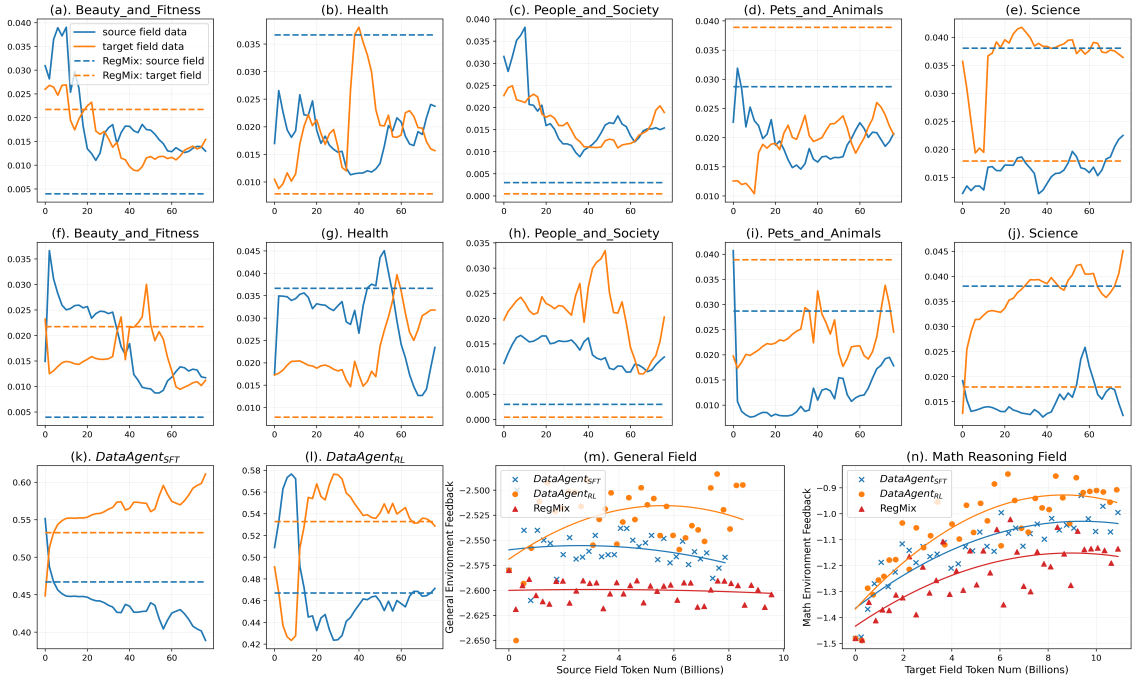


Figure 3: Training on LLaMA-DCLM and math reasoning, (a)-(e) presents DataAgent$_{RL}$'s output trajectories on 5 domains within the 52-dimensional space. (f)-(j) presents DataAgent$_{SFT}$'s trajectories. (k)-(l) presents the two agents' trajectories on the 2-dimensional domain space. (a)-(l) share the legend within (a). Dashed lines show RegMix outputs. (m)-(n) present the dynamics of model performance on the evaluation environment with increasing training data in the corresponding field.

of these savings is provided in Sec. 4.6.

Training agent models involves additional computational cost from trajectory sampling, where training proxy models on 384 trajectories and their evaluation require 1996.08 GPU hours in total. In contrast, training the lightweight agent model with SFT and CQL finishes within 10 minutes. **Importantly, the agent training process is needed only once, and our experiments demonstrate that the Data Mixing Agent generalizes across source and target fields, target models, and domain spaces without re-training.**

### 4.5 Analysis on Reweighting Trajectories

We show the data mixing trajectories guided by the agent to provide more intuition on its learned heuristics. The 2-dimensional results and part of 52-dimensional results are shown in Fig. 3(a)-(l). The full trajectories of all domains and a more detailed analysis are presented in Appn. E.3.

**Data Mixing Agent follows a less-to-more trend for target field data along, but DataAgent$_{RL}$ adopts a more fine-grained approach.** In Fig. 3(k)-(l), both the DataAgent$_{RL}$ and DataAgent$_{SFT}$ models increase target field data, where DataAgent$_{SFT}$ shows a radical increasing trend for math data, almost monotonically from 45% to over 60%. DataAgent$_{RL}$ adopts a fine-grained three-stage strategy: (1) **Early warm-up** to prioritize source field data; (2) **Mid training** to rapidly increase target field data; (3) **Final stage** to gradually reintroduce more source field data, stabilizing around the optimal RegMix weights. The superior performance of

DataAgent$_{RL}$ proves the effectiveness of optimization on various trajectories via reinforcement learning. In Fig. 3(a)-(j), the 52-dimensional results further strengthen the above arguments. In all domains, DataAgent$_{SFT}$ organizes the target field data from 60% of the trajectories to increase monotonically, while DataAgent$_{RL}$ introduces more subtle strategies in 80% of the trajectories.

**Data Mixing Agent learns heuristics that correspond to human intuition.** Wettig et al. (2025) summarized the top-3 domains that benefit the MMLU performance: $Science\&Tech.$, $Health$, and $Politics$. In Fig. 3(a)-(e), we observe a significant uplift of data in the corresponding domains compared to RegMix distributions: $Science$, $Health$, and $People\&Society$. Wettig et al. (2025) also enumerated domains that hurt performance, such as $Fashion\&Beauty$. where DataAgent$_{RL}$ also conveys an explicit downsampling process. These observations further encourage the discovery of new general heuristics. For example, DataAgent$_{RL}$ continuously reduces data from both source and target fields in the $Pets\&Animals$, indicating its less importance in general or math reasoning capabilities.

### 4.6 Data Efficiency

We explore how efficiently the agents leverage the source and target field data to improve or preserve model capabilities in the corresponding fields. We record the performance dynamics of LLaMA-DCLM with increasing training data. The results are shown in Fig. 3(m)-(n). A more detailed analysis is presented in Appn. E.4.

**Data Mixing Agent leverages general field data more efficiently than RegMix, better preserving model capabilities in the source field.** As shown in Fig. 3(m), the agent obtains higher general feedback values from the environment at most token budgets for the source field. DataAgent$_{RL}$ further outperforms DataAgent$_{SFT}$, with feedback values fluctuating around -2.525. Notably, DataAgent$_{RL}$ shows higher variance along the domain reweighting trajectory, reflecting its active strategies in adjusting domain reweighting distributions to improve source field capabilities. Its superior performance on general benchmarks (Table 1a) indicates the effectiveness of such strategies.

**Data Mixing Agent leverages data from the math reasoning field more efficiently than RegMix, achieving higher performance in the target field.** As shown in Fig. 3(n), though all methods show logarithmic-scale improvements on math reasoning feedback, the agent methods show a faster momentum in increasing feedback values. RegMix performance stabilizes around -1.2 while both data mixing agent methods achieve performance over -1.1. DataAgent$_{RL}$ further outperforms DataAgent$_{SFT}$ with the optimized feedback values over -1.0. Overall, DataAgent$_{RL}$ can coordinate the source and target field data to improve performance on multiple target capabilities.

**Data Mixing Agent achieves balanced continual pre-training performance with less reliance on data from the source field.**. While we set a total training budget of 21B tokens, DataAgent$_{RL}$ triggers an early stopping at 19.92B tokens, and DataAgent$_{SFT}$ triggers an early stopping at 18.86B tokens, due to the exhaustion of the target field data. These results show that the agent can achieve superior performance than RegMix in both the general and math reasoning fields while saving up to a training token budget of 2.14B.

## 5 Related Work

Continual pre-training has been widely applied in domain-specific LLM adaptation (Shao et al., 2024; Guo et al., 2024; Hui et al., 2024; Xie et al., 2024; Lin et al., 2025; Tu et al., 2024), often facing the catastrophic forgetting problem (Hui et al., 2024; Lin et al., 2025; Luo et al., 2023; Yang et al., 2024). To mitigate this, data mixing strategies have merged (Xie et al., 2023; Liu et al., 2024b; Xia et al., 2023; Luo et al., 2024) to balance model performance across fields. Recent studies also highlight the role of domain space definitions in improving re-weighting performance (Wettig et al., 2025; Rukhovich et al., 2025; Diao et al., 2025; Xi et al., 2025). More details in Appn. F.

## 6 Conclusion

We propose Data Mixing Agent, the first model-based domain reweighting method for continual pre-training. By learning general heuristics through reinforcement learning on sampled mixing trajectories with evaluation feedback, it consistently outperforms strong baselines on 12 general and math reasoning benchmarks. The learned strategies generalize across source data, models, domain spaces, and new fields without retraining, while aligning with human intuitions and achieving superior performance with less data.

## Limitations

The limitations of this work are threefold. Firstly, due to the high cost of continual pre-training and limited computational resources, we evaluated Data Mixing Agents on only two domain reweighting spaces, four target models, and two target fields (math reasoning and code generation). Future work will extend the evaluation to a broader range of settings, particularly to test generalization across more target fields and domain spaces. Secondly, our evaluation focused exclusively on continual pre-training scenarios. An important direction for future work is to explore the applicability of Data Mixing Agents in large-scale pre-training. Thirdly, although Data Mixing Agents generalize well across related fields, they may underperform when the target field exhibits a significant distribution shift from the source field (e.g., from math reasoning to the medical domain). We leave a detailed analysis of such cases, along with potential optimizations, to future work.

## References


Aida Amini, Saadia Gabriel, Peter Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. *arXiv preprint arXiv:1905.13319*.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR.

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

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

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

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

Shizhe Diao, Yu Yang, Yonggan Fu, Xin Dong, Dan Su, Markus Kliegl, Zijia Chen, Peter Belcak, Yoshi Suhara, Hongxu Yin, et al. 2025. Climb: Clustering-based iterative data mixture bootstrapping for language model pre-training. *arXiv preprint arXiv:2504.13161*.

Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, et al. 2022. Glam: Efficient scaling of language models with mixture-of-experts. In *International conference on machine learning*, pages 5547–5569. PMLR.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. 2020. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. 2024. The llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407.

Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Yu Wu, YK Li, et al. 2024. Deepseek-coder: When the large language model meets programming–the rise of code intelligence. *arXiv preprint arXiv:2401.14196*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. 2022. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*.

Jian Hu, Xibin Wu, Zilin Zhu, Weixun Wang, Dehao Zhang, Yu Cao, et al. 2024. Openrlhf: An easy-to-use, scalable and high-performance rlhf framework. *arXiv preprint arXiv:2405.11143*.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, et al. 2024. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*.

Hamish Ivison, Yizhong Wang, Jiacheng Liu, Zeqiu Wu, Valentina Pyatkin, Nathan Lambert, Noah A Smith, Yejin Choi, and Hanna Hajishirzi. 2024. Unpacking dpo and ppo: Disentangling best practices for learning from preference feedback. *Advances in neural information processing systems*, 37:36602–36633.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. 2020. Conservative q-learning for offline reinforcement learning. *Advances in neural information processing systems*, 33:1179–1191.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. 2022. Solving quantitative reasoning problems with language models. *Advances in Neural Information Processing Systems*, 35:3843–3857.

Hao Li, Bowen Deng, Chang Xu, Zhiyuan Feng, Viktor Schlegel, Yu-Hao Huang, Yizheng Sun, Jingyuan Sun, Kailai Yang, Yiyao Yu, et al. 2025. Mira: Medical time series foundation model for real-world health data. *arXiv preprint arXiv:2506.07584*.

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

Zhenghao Lin, Zihao Tang, Xiao Liu, Yeyun Gong, Yi Cheng, Qi Chen, Hang Li, Ying Xin, Ziyue Yang, Kailai Yang, et al. 2025. Sigma: Differential rescaling of query, key and value for efficient language models. *arXiv preprint arXiv:2501.13629*.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. 2024a. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. 2020. Logiqa: A challenge dataset for machine reading comprehension with logical reasoning. *arXiv preprint arXiv:2007.08124*.

Qian Liu, Xiaosen Zheng, Niklas Muennighoff, Guangtao Zeng, Longxu Dou, Tianyu Pang, Jing Jiang, and Min Lin. 2024b. Regmix: Data mixture as regression for language model pre-training. *arXiv preprint arXiv:2407.01492*.

Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. 2023. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *arXiv preprint arXiv:2308.08747*.

Zheheng Luo, Xin Zhang, Xiao Liu, Haoling Li, Yeyun Gong, Chen Qi, and Peng Cheng. 2024. Velocitune: A velocity-based dynamic domain reweighting method for continual pre-training. *arXiv preprint arXiv:2411.14318*.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*.

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

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

Alexey Rukhovich, Alexander Podolskiy, and Irina Piontkovskaya. 2025. Commute your domains: Trajectory optimality criterion for multi-domain learning. *arXiv preprint arXiv:2501.15556*.

Shiori Sagawa, Pang Wei Koh, Tatsunori B Hashimoto, and Percy Liang. 2019. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. *arXiv preprint arXiv:1911.08731*.

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

Takuma Seno and Michita Imai. 2022. d3rlpy: An offline deep reinforcement learning library. *Journal of Machine Learning Research*, 23(315):1–20.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Zhiqiang Shen, Tianhua Tao, Liqun Ma, Willie Neiswanger, Zhengzhong Liu, Hongyi Wang, Bowen Tan, Joel Hestness, Natalia Vassilieva, Daria Soboleva, et al. 2023. Slimpajama-dc: Understanding data combinations for llm training. *arXiv preprint arXiv:2309.10818*.

Haizhou Shi, Zihao Xu, Hengyi Wang, Weiyi Qin, Wenyuan Wang, Yibin Wang, Zifeng Wang, Sayna Ebrahimi, and Hao Wang. 2024. Continual learning of large language models: A comprehensive survey. *ACM Computing Surveys*.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*.

Dan Su, Kezhi Kong, Ying Lin, Joseph Jennings, Brandon Norick, Markus Kliegl, Mostofa Patwary, Mohammad Shoeybi, and Bryan Catanzaro. 2024. Nemotron-cc: Transforming common crawl into a refined long-horizon pretraining dataset. *arXiv preprint arXiv:2412.02595*.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. 1999. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12.

Tao Tu, Shekoofeh Azizi, Danny Driess, Mike Schaekermann, Mohamed Amin, Pi-Chuan Chang, Andrew Carroll, Charles Lau, Ryutaro Tanno, Ira Ktena, et al. 2024. Towards generalist biomedical ai. *Nejm Ai*, 1(3):AIoa2300138.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Ke Wang, Houxing Ren, Aojun Zhou, Zimu Lu, Sichun Luo, Weikang Shi, Renrui Zhang, Linqi Song, Mingjie Zhan, and Hongsheng Li. 2023. Mathcoder: Seamless code integration in llms for enhanced mathematical reasoning. *arXiv preprint arXiv:2310.03731*.

Johannes Welbl, Nelson F Liu, and Matt Gardner. 2017. Crowdsourcing multiple choice science questions. *arXiv preprint arXiv:1707.06209*.

Alexander Wettig, Kyle Lo, Sewon Min, Hannaneh Hajishirzi, Danqi Chen, and Luca Soldaini. 2025. Organize the web: Constructing domains enhances pre-training data curation. *arXiv preprint arXiv:2502.10341*.

Xiangyu Xi, Deyang Kong, Jian Yang, Jiawei Yang, Zhengyu Chen, Wei Wang, Jingang Wang, Xunliang Cai, Shikun Zhang, and Wei Ye. 2025. Samplemix: A sample-wise pre-training data mixing strategey by coordinating data quality and diversity. *arXiv preprint arXiv:2503.01506*.

Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. 2023. Sheared llama: Accelerating language model pre-training via structured pruning. *arXiv preprint arXiv:2310.06694*.

Qianqian Xie, Weiguang Han, Zhengyu Chen, Ruoyu Xiang, Xiao Zhang, Yueru He, Mengxi Xiao, Dong Li, Yongfu Dai, Duanyu Feng, et al. 2024. Finben: A holistic financial benchmark for large language models. *Advances in Neural Information Processing Systems*, 37:95716–95743.

Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifeng Lu, Percy S Liang, Quoc V Le, Tengyu Ma, and Adams Wei Yu. 2023. Doremi: Optimizing data mixtures speeds up language model pretraining. *Advances in Neural Information Processing Systems*, 36:69798–69818.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, et al. 2024. Qwen2. 5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*.

Jiasheng Ye, Peiju Liu, Tianxiang Sun, Jun Zhan, Yunhua Zhou, and Xipeng Qiu. 2024. Data mixing laws: Optimizing data mixtures by predicting language modeling performance. *arXiv preprint arXiv:2403.16952*.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2023. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*.

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

Xiyuan Zhang, Ranak Roy Chowdhury, Rajesh K Gupta, and Jingbo Shang. 2024. Large language models for time series: A survey. *arXiv preprint arXiv:2402.01801*.

## A Agent Model

To prove that data mixing agent is qualified as an agent model, we'd like to start with Google's definition of the term AI agent:

> AI agents are software systems that use AI to pursue goals and complete tasks on behalf of users. They show reasoning, planning, and memory and have a level of autonomy to make decisions, learn, and adapt.

Based on this definition, we prove data mixing agent can be qualified as an agent for the following reasons:

**Software systems**: Data mixing agent is a Transformer-based machine learning system. Goals and Tasks: Data mixing agent takes the previous training history as inputs and output the optimal data mixing recipe for the next domain reweighting step.

**Reasoning, planning, and memory**: Data mixing agent memorizes previous data mixing trajectories and reasons over them to plan for the future training recipe.

**Learn, and adapt**: Our experiments prove that data mixing agent can generalize well across fields, including math reasoning and code generation.

According to the definition, an AI agent is not necessarily a language model. The system calls the Transformer-based data mixing agent. The system directly calls the data mixing agent at each data re-weighting step to predict the next mixing weight.

## B Start State Estimation with Synthetic Data

In scenarios where the data from the source field are unavailable (Grattafiori et al., 2024; Liu et al., 2024a), we explore randomly sampled data from the target model as estimates for the start state. To prove the viability of this method, we experiment on five Pythia models (Biderman et al., 2023), as they are trained on the same open source Pile dataset (Gao et al., 2020), where the ground-truth start state can be calculated using the same method as in the paper. Specifically, we first randomly sample tokens from the target model simply by prepending the start-of-sentence token to start generation with a default temperature of 1.0. The generated data are then passed through the same Nvidia domain classifier to obtain a domain label. The estimated start state is calculated by normalizing sample numbers on each domain in the generated data.

To prove the viability of such estimation, we calculate the KL divergence between the estimated start state and the ground-truth distribution obtained from the Pile dataset, and the results are presented in Fig. 4. As shown, sampling over 2,000 random samples already leads to a KL divergence of less than 0.1 on 4 out of 5 Pythia models, showing a fast convergence rate of the proposed estimation method. The estimated distribution also converges on most models with over 4,000 samples. In addition, larger models such as Pythia-6.9B also show more accurate estimates and a faster convergence rate, possibly due to the higher quality of their generated data, leading to more accurate domain classification. The above results prove the effectiveness of using random samples from the target model as estimates for their start states. This method is further utilized in experiments on the Pythia-1.4B model.

## C Implementation Details

In this section, we provide more technical implementation details of Data Mixing Agent, including the trajectory sampling algorithm, the evaluation environment feedback design, the data mixing agent training process, and our evaluation process on various settings.

### C.1 Trajectory Sampling

**Algorithm** The detailed algorithm for the trajectory sampling process is provided in Algorithm 1. The function $CalculateInductiveScores$ describes the scoring algorithm. This function is designed based on three inductive biases that denote a potentially good distribution:

- The data re-weighting distribution at the current step should not deviate significantly from that of the previous step;
- As the data re-weighting progresses, the distribution at each step should gradually align more closely with the target distribution;
- The distribution at the current step should differ from those at the same step in previously sampled trajectories to encourage diversity.

The target distribution is defined as the complement of the start state: probabilities for source-field domains are set to zero, while those for target-field domains are estimated based on their empirical distribution in the target field data. **This target distribution serves as a soft constraint that encourages trajectories to gradually reduce reliance on source-field data and increase the coverage of target-field data**, thereby accelerating the continual pre-training process and saving computation costs.

**Implementation** During implementation, we use 100B random tokens from the DCLM (Li et al., 2024) as the source field data $S$, and the math split (about 10B tokens) of the Dolmino-mix-1124 (OLMo et al., 2024) dataset as the target field data $T$. The max data reweighting steps $M = 80$, and the reweighting sample number per step $R$ is set to 8K. To ensure inclusion of both high-quality and low-quality trajectories, we run Algorithm 1 four times, each with the path sampling number

**Algorithm 1:** Data Mixing Trajectory Sampling with Top-K Inductive Biases

**Input:** Source field Data $S$, Target field data $T$, Path sampling number $P$, Max data reweighting steps $M$, Reweighting sample number per step $R$, Inductive threshold $K$

**Output:** Sampled trajectories $\mathcal{T}$

$D \leftarrow$ GetDomainConfig() ; // Load the domain space based on definitions.

$\rho_s \leftarrow$ GetStartState($S$, $D$) ; // Estimate start state from source data $S$.

$\rho_t \leftarrow$ GetTargetState($T$, $D$) ; // Estimate target state from source data $S$.

$T_M \leftarrow R \cdot M$ ; // Max data samples for each trajectory.

$\mathcal{T} \leftarrow [\,]$ ; // Initialize empty trajectory list.

**for** $p \leftarrow 1$ **to** $P$ **do**

  $d \leftarrow 0$, $c \leftarrow 0$, $\rho \leftarrow \rho_s$, $\tau \leftarrow [\rho_s]$ ; // Initialize the current trajectory.

  **while** $d < M$ **do**

    $\mathcal{C} \leftarrow [\,]$ ; // Reset candidate list.

    **for** $i \leftarrow 1$ **to** 20000*;* // Repeat the sampling 20,000 times.

    **do**

      $\rho' \leftarrow$ RandomProbability($|D|$) ; // Randomly sample a distribution.

      $s \leftarrow$ CalculateInductiveScores($d$, $\rho'$, $\mathcal{T}$, $\tau$, $\rho$, $\rho_t$) ;

      Append $(\rho', s)$ to $\mathcal{C}$ ;

    $\hat{\rho} \leftarrow$ RandomTopK($\mathcal{C}$, $K$) ; // Randomly select from top-$K$ candidates with lowest inductive scores.

    Append $\hat{\rho}$ to $\tau$ ; // Update current trajectory.

    $\rho \leftarrow \hat{\rho}$, $d \leftarrow d + 1$ ;

    $c \leftarrow c+$ TargetSamplesCovered($\hat{\rho}$, $R$) ; // Track covered target sample number.

    **if** $c \geq |T|$ **then**

      **break** ; // Early stopping if target data is fully covered.

  Append $\tau$ to $\mathcal{T}$ ; // Store the current trajectory.

**Function** CalculateInductiveScores($d$, $\rho'$, $\mathcal{T}$, $\tau$, $\rho$, $\rho_t$)**:**

  $s_c \leftarrow \mathrm{KL}(\rho \,||\, \rho')$ ; // KL divergence between the current and last action.

  $s_t \leftarrow \mathrm{KL}(\rho_t \,||\, \rho')$ ; // KL divergence between the current action and target state.

  $s_d \leftarrow 0$ ;

  **if** $|\mathcal{T}| > 0$ **then**

    $S \leftarrow [\,]$ ; // A set to store the similarities.

    **foreach** $\tau' \in \mathcal{T}$ **do**

      **if** $d < |\tau'|$ **then**

        $S \leftarrow S \cup \{\mathrm{KL}(\tau'[d] \,||\, \rho')\}$ ; // Calculate similarities to states in previous trajectories.

    **if** $|S| > 0$ **then**

      $s_d \leftarrow \frac{1}{|S|} \sum_{x \in S} x$ ; // Average similarity to previous states

  **return** $\alpha \cdot s_c + \beta \cdot \sigma\left(\frac{d}{5}\right) \cdot s_t - \gamma \cdot s_d$ ; // Final inductive score

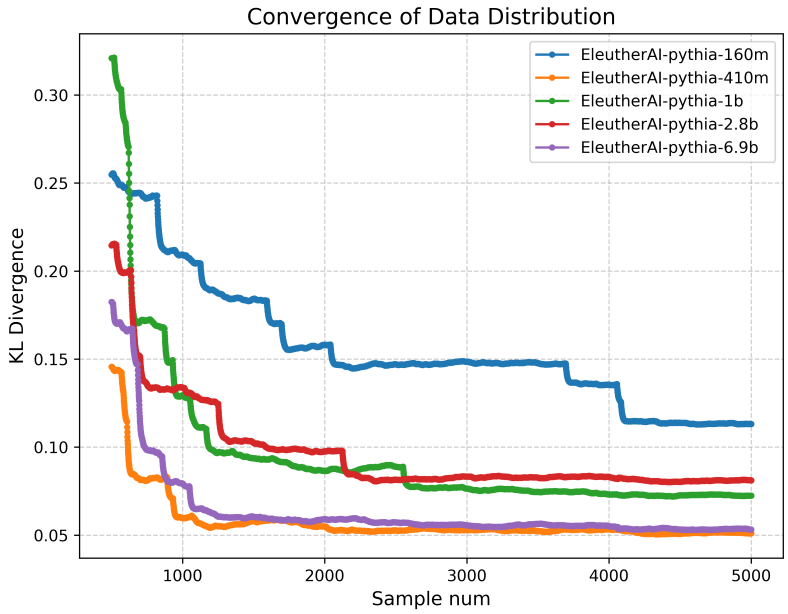


Figure 4: The KL divergence between the estimated start state by sampled data from the target model and the ground-truth distribution obtained from the Pile dataset. The results are averages of 5 random runs.

**Algorithm 2:** Continal Pre-training with Data Mixing Agent

**Input:** $N$ domains of source data $\{S_1, S_2, ..., S_N\}$ and target data $\{T_1, T_2, ..., T_N\}$, the agent $f$, Max data reweighting steps $M_{tgt}$, Reweighting sample number per step $R_{tgt}$, the target model $\mathcal{M}_{tgt}$, the evluation environment $\mathcal{E}$.

**Output:** The continually pretrained target model checkpoint $\hat{\mathcal{M}}_{tgt}$

$D \leftarrow$ GetDomainConfig() ; // Load the domain space based on definitions.

$\rho_s \leftarrow$ GetStartState($S$, $D$) ; // Estimate start state from source data $S$.

$\hat{\mathcal{M}}_{tgt} \leftarrow \mathcal{M}_{tgt}$ ; // Initialize the current model checkpoint.

$\mathcal{T}_{tgt} \leftarrow [\rho_s]$ ; // Initialize trajectory list.

$reward_{tgt} \leftarrow \varnothing$ ; // Initialize feedback list.

c$\leftarrow 0$

**for** $t \leftarrow 1$ **to** $M_{tgt}$ **do**

  $reward(\hat{\mathcal{M}}_{tgt}) \leftarrow$ GetEnvFeedback($\hat{\mathcal{M}}_{tgt}, \mathcal{E}$) ; // Get environment feedback.

  Append $reward(\hat{\mathcal{M}}_{tgt})$ to $reward_{tgt}$ ; // Update current feedback list.

  $reward^s_{tgt} \leftarrow$ std($reward_{tgt}$); // Standardize the current feedback list.

  $\{\tilde{\rho}\} \leftarrow$ concat($\mathcal{T}_{tgt}, reward^s_{tgt}$); // Concat trajectories with the corresponding feedback.

  $\rho_t \leftarrow f(\tilde{\rho}_1, \tilde{\rho}_2, ..., \tilde{\rho}_{t-1})$; // Obtain domain reweighting distribution from the agent.

  $\mathcal{B}_t \leftarrow$ sample($\{S_i\}_1^N$,$\{T_i\}_1^N$,$\rho_t$); // Sample domain data with the current distribution.

  Update weights for $\hat{\mathcal{M}}_{tgt}$ with the training loss $\mathcal{L}(\hat{\mathcal{M}}_{tgt}, \mathcal{B}_t)$;

  Append $\rho_t$ to $\mathcal{T}_{tgt}$;

  $c \leftarrow c+$ TargetSamplesCovered($\hat{\rho}$, $R$) ; // Track covered target sample number.

  **if** $c \geq |T|$ **then**

    **break** ; // Early stopping if target data is fully covered.

$P = 96$ and the threshold $K$ set to 1, 100, 1000, and 10,000, leading to the trajectory set $\mathcal{T}$ with subsets $\mathcal{T}_{top1}$, $\mathcal{T}_{top100}$, $\mathcal{T}_{top1000}$, and $\mathcal{T}_{top10000}$, with 384 trajectories in total.

### C.2 Evaluation Environment Design and Feedback Collection

During implementation, the environment assesses the general capability of the checkpoints via 500 high-quality general-domain questions and answers from the validation set of the MMLU dataset [2]. The math reasoning capability is evaluated with 500 random samples from the training split of the MATH dataset [3].

Leveraging this evaluation environment, we collect feedback data by training a small proxy model $\mathcal{M}_p$ with the LLaMA model structure and 50M parameters on each sampled trajectory from scratch. The model checkpoint is evaluated on this environment at each data reweighting step, resulting in 27,266 feedbacks. Formally, for the $i$-th data mixing distribution $\rho_i \in \tau$, we obtain a tuple $(\rho_i, \mathcal{R}(\mathcal{M}_p^i))$, where $\mathcal{R}(\mathcal{M}_p^i)$ denotes the environment feedback for the model checkpoint $\mathcal{M}_p^i$ after training on the $i$-th domain reweighting step. Notably, the feedback at the start state is obtained with the initialized base proxy model.

### C.3 SFT-based Warming Up

We first perform Supervised Fine-Tuning (SFT) to reduce the parameter searching space in the reinforcement learning phase. We train the agent from scratch on the high-quality $\mathcal{T}_{top1}$ trajectory subset (obtained during the trajectory sampling process) with a simple MSE loss. At the data reweighting step $t$, the agent is optimized as follows:

$$\mathcal{L}_{SFT} = \sum \left(\hat{\rho}_t - f(\tilde{\rho}_0, \tilde{\rho}_1, ..., \tilde{\rho}_{t-1})\right)^2 \quad (4)$$

where $\hat{\rho}_t$ denotes the ground-truth distribution in step $t$. Notably, before the SFT process, we standardize the environment feedback on each target field across data reweighting steps within all trajectories by forcing their mean value to 0 and standard deviation to 1. This is to regularize the reward space for the agent and avoid out-of-distribution rewards from unseen target models. The feedback for later reinforcement learning and agent inference processes also utilizes this standardization procedure.

[2] https://huggingface.co/datasets/cais/mmlu

[3] https://huggingface.co/datasets/EleutherAI/hendrycks_math

### C.4 Conservative Q-Learning

We select Conservative Q-Learning (CQL) (Kumar et al., 2020) as the optimization algorithm. CQL prevents overestimation of Q-values for out-of-distribution actions by encouraging the learned Q-function to be conservative. Specifically, CQL introduces a conservative penalty for the Q-function optimization process with the following training objective:

$$\underbrace{\mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\left[\left(Q(s,a) - \left(r + \gamma \max_{a'} Q(s', a')\right)\right)^2\right]}_{\text{Bellman error}} + \alpha \cdot \underbrace{\left(\mathbb{E}_{s\sim\mathcal{D}}\left[\mathbb{E}_{a'\sim\mathcal{U}(\mathcal{A})}(Q(s,a'))\right] - \mathbb{E}_{(s,a)\sim\mathcal{D}}[Q(s,a)]\right)}_{\text{Conservative penalty}} \quad (5)$$

where the first term denotes the standard Bellman error for Q-learning, and the second term denotes a conservative penalty to minimize the Q value expectations for randomly sampled actions from the current state under another distribution $\mathcal{U}(\mathcal{A})$. The target model and the Q function are then trained in an actor-critic (Sutton et al., 1999) structure, where the data agent acts as the actor model, optimized with policy gradient to enhance actions that maximize the Q value feedback from the critic model. Another neural network is initialized from scratch as the parameterized Q function, which acts as the critic model to evaluate model actions and optimized via Eqn. 5.

During implementation, we randomly sample fragments $\tau$ (don't have to be full trajectories) from the data mixing trajectory set $\mathcal{T}$. At domain reweighting step $t$, $s = [\rho_0, \rho_1, ..., \rho_{t-1}]$, $a = \rho_t$, and $s' = [\rho_0, \rho_1, ..., \rho_t]$. The scalar reward value $r$ is obtained as the gain of a linear combination of environment feedback $\mathcal{R}(\mathcal{M}_p^t)$ compared to that of the last step:

$$r = \sum_{i=1}^{|D|} \lambda_i \mathcal{S}(\mathcal{M}_p^t, D_i) - \sum_{i=1}^{|D|} \lambda_i \mathcal{S}(\mathcal{M}_p^{t-1}, D_i) \quad (6)$$

During implementation, we set all coefficients to be equal to encourage balanced consideration of target field capabilities: $\lambda_i = \frac{1}{|D|}$. The critic model $f'$ is parameterized by another single-layer Transformer decoder, followed by a linear layer and sigmoid function to project the representations into a Q-value scalar, with the following inference function:

$$Q(s,a) = f'(\rho_0, \rho_1, ..., \rho_t) \quad (7)$$

The agent model and the Q-function model are iteratively optimized in this actor-critic manner until convergence.

The advantage of CQL over SFT is expected. The theoretical foundation of the data mixing agent lies in learning from both actions that enhance and degrade model performance. CQL, as SOTA off-policy RL algorithm, is inherently designed to optimize policies using both positive and negative signals from sampled trajectories. Specifically, it promotes actions associated with high Q-values while suppressing those associated with low Q-values.

In contrast, SFT is an instance of imitation learning, which optimizes the policy by fitting it to high-quality (i.e., "good") trajectories. By design, SFT cannot exploit informative signals from suboptimal or "bad" trajectories, where the policy should instead learn to avoid certain actions. When incorporated, such trajectories may even introduce noise rather than useful learning signals. For this reason, we employ SFT only as a warm-up strategy and subsequently apply CQL to fully optimize the policy over all trajectories.

We believe the strong generalization capability of the agent stems from the same intuition that motivated the development of the data mixing agent in the first place: there exists a rich heuristic space for domain reweighting. These heuristics are largely model- and data-agnostic, and can therefore be unified within a compact agent model to guide data mixing trajectories in a principled manner. Our approach differs from prior work in that, rather than relying on manually designed algorithms, we train a model to parameterize these heuristics and enable automated domain reweighting. As a result, we expect our method to achieve comparable or superior generalization performance, consistent with the empirical success of previous heuristic-based approaches.

### C.5 System Pipeline of Data Mixing Agent

Following Google's definition [4], an AI agent is a software system that autonomously pursues goals and completes tasks by exhibiting reasoning, planning, and adaptation capabilities. Under this definition, the proposed data mixing agent qualifies as an AI agent. Concretely, it is implemented as a Transformer-based learning system that operates as an independent decision-making module. Its objective is to predict an optimal data mixing recipe for the next domain reweighting step, given the historical training trajectory as input. By encoding past data mixing decisions and their associated outcomes, the agent effectively maintains memory over previous trajectories and reasons over them to plan future data allocation strategies. Moreover, empirical results demonstrate that the agent can learn and adapt across domains, generalizing effectively to diverse target fields such as math reasoning and code generation.

In the overall system pipeline, the data mixing agent is invoked directly at each reweighting step to predict the next set of domain mixing weights, rather than calling a language model for decision making. This design aligns with the general notion of AI agents as decision-making systems that are not necessarily language models themselves.

[4] https://cloud.google.com/discover/what-are-ai-agents

## D Experimental Settings

### D.1 Target Models

We aim to rigorously evaluate domain reweighting methods on target models that do not possess math or coding capabilities. Since most open-source models have been optimized on large-scale data from the math reasoning or code generation field, we pre-train three models from scratch, with the same LLaMA3 model architecture (Grattafiori et al., 2024) of 32 Transformer layers and 3B model parameters, on 100B randomly sampled tokens from the DCLM (Li et al., 2024), Fineweb-Edu (Penedo et al., 2024), and Nemotron-CC (Su et al., 2024) dataset, resulting in three target models: **LLaMA-DCLM**, **LLaMA-FWE**, **LLaMA-Nemo**. We also include the **Pythia-1.4B** model (Biderman et al., 2023) to evaluate performance on existing open-source models and scenarios when data from the source field is not directly available.

### D.2 Baseline Methods

We compare Data Mixing Agent (**DataAgent**$_{RL}$) with the following baseline methods:

- **Base Model**: direct evaluation of the target models on the benchmarks, reflecting model capabilities before the continual pretraining phase;
- **Naive Training**: continually training the base model on data from the target field without curating any data mixtures from source-field data;

- **RegMix** (Liu et al., 2024b): state-of-the-art static domain re-weighting method. It trains large quantities of 1B-sized small proxy models with LLaMA structure (512 models in our implementation) on random domain distributions, then evaluates these models on the target benchmarks. The best data mixing recipe is determined by fitting a regression model to the feedback and selecting distributions that lead to the highest scores. We mostly follow the settings of Wettig et al. (2025) in implementing the RegMix algorithm;

- **Dynamic Batch Loading (DBL)** (Xia et al., 2023): State-of-the-art dynamic domain reweighting method that modifies the data mixture on the fly. It first estimates a reference optimal loss on each domain reweighting step via the scaling law (Hoffmann et al., 2022), then computes the excess loss between the current evaluation loss and the optimal loss on each domain, increasing data ratios for slowly learning domains for the next training data batch. We skip the procedure for fitting the scaling function and directly utilize the loss of the base model on the general environment set as the reference loss for the general domain, the loss of the naively trained model on the math environment set as the reference loss for math reasoning, and the loss of the naively trained model on the code environment set as the reference loss for code generation. The DBL uses the same reweighting samples per step as Data Mixing Agent until the target field data is fully covered. We only use DBL as a baseline on the 2-dimensional domain space, due to the lack of evaluation sets for all domains in the 52-dimensional domain space;

- **DataAgent**$_{SFT}$: the data mixing agent model without the reinforcement learning process. The model mostly provides heuristically appropriate trajectories because it's only fine-tuned on the $\mathcal{T}_{top1}$ dataset. We include this baseline method to assess the effectiveness of off-policy optimization with CQL.

### D.3 Target Model and Data

For data from the source field, the self-pretrained LLaMA-3B models utilize their corresponding pretraining data, each with 100B tokens. We use randomly sampled data from the Pythia-1.4B model as the source field data for itself, applying the agent in scenarios where the data from the source field is not directly available. Following the method in Appendix B, we sample 10B tokens by pre-pending the start-of-sentence token to start generation with a default temperature of 1.0. The generated data are then filtered through the Nvidia text quality classifier [5], where all data within the "Low quality" class are discarded, resulting in a source field dataset with around 7.7B tokens. For data from the math reasoning field, we select the math split (10B tokens) of the Dolmino-mix-1124 dataset [6], which was used for the mid-training process of the OLMo2 model series (OLMo et al., 2024), including data sources such as TuluMath (Ivison et al., 2024), MathCoder (Wang et al., 2023), and Metamath (Yu et al., 2023). For data from the code generation field, we select the GitHub training split [7] of the SlimPajama-DC dataset (Shen et al., 2023) with 30B tokens.

### D.4 Elaborations on Target Model Selection

Our experiments aim to validate the proposed data mixing agent as a domain reweighting method rather than to achieve state-of-the-art leaderboard performance. Therefore, absolute performance gaps to SOTA LLMs do not undermine the conclusions. Specifically, we employ 3B-parameter models pretrained on 100B tokens, whereas strong open-source LLMs (e.g., Llama-3.1-405B-Instruct) use hundreds of billions of parameters and are pretrained on trillions of tokens. We avoid larger open-source LLMs for three reasons: (1) many LLMs have been heavily optimized for math or code, introducing biases that confound controlled evaluation of domain reweighting; (2) their pretraining data compositions are not publicly available, making them unsuitable for continual pretraining studies; and (3) their large scale (typically>70B parameters) renders continual pretraining prohibitively expensive.

### D.5 Evaluation Benchmarks

We evaluate target models' general capabilities by evaluating with the lm_eval evaluation library [8] on the MMLU (Hendrycks et al.,

[5] https://huggingface.co/nvidia/quality-classifier-deberta

[6] https://huggingface.co/datasets/allenai/dolmino-mix-1124

[7] https://huggingface.co/datasets/MBZUAI-LLM/SlimPajama-627B-DC

[8] https://github.com/EleutherAI/lm-evaluation-harness

2020), HellaSwag (Zellers et al., 2019) (Hella.), OpenBookQA (Mihaylov et al., 2018) (OBQA), Winogrande (Sakaguchi et al., 2021) (Wino.), ARC-Challenge (Clark et al., 2018) (ARC-C), PiQA (Bisk et al., 2020), SciQ (Welbl et al., 2017), and LogiQA (Liu et al., 2020) benchmarks. We evaluate the math reasoning capabilities using the math_lm_eval library [9] on the GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021), Minerva (Lewkowycz et al., 2022), and MathQA (Amini et al., 2019) benchmarks. We evaluate the code generation capabilities using the eval_plus library [10] on the HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021) benchmarks. The MMLU and GSM8K benchmarks are evaluated with a 5-shot setting, the Minerva benchmark is evaluated in a 4-shot setting. Other benchmarks are evaluated with a zero-shot setting.

### D.6 Target Model Training Setting

Firstly, the agent is trained based on a 26-dimensional domain definition, leading to a 52-dimensional domain reweighting space. To evaluate on different domain spaces, we further employ the data mixing agent on the original 2-dimensional space based on data sources (source and target). Note that this action does not require the agent to retrain, as its action can be directly converted by summing the 26 probabilities for source/target fields into a single dimension, still preserving a probability distribution. During training, we set the number of reweighting samples per step $R_{tgt}$ to 64K and the maximum data reweighting steps $M_{tgt}$ to 80. We use the same evaluation environment as the Data Mixing Agent training when the target field is math reasoning, and change the MATH validation set to 1,000 random samples from the GitHub validation split of the SlimPajama-DC dataset when the target field is code generation. Due to resource limits and the highly imbalanced distribution of code data in the 26-dimensional domain space, we only train on the 2-dimensional data reweighting space for code generation.

### D.7 Implementation

We continually pre-train the target model in a distributed manner on 8 nodes with a total of 64 Nvidia A100 GPUs with 40GB of memory. The code for training the target model with data mixing agents is built upon the Megatron-LM framework (Shoeybi et al., 2019). The SFT-based warm-up stage is conducted on the OpenRLHF library (Hu et al., 2024). The CQL-based off-policy reinforcement learning framework is built on the d3rlpy (Seno and Imai, 2022) library with further modifications to support training on Huggingface Transformer models.

[9] https://github.com/ZubinGou/math-evaluation-harness

[10] https://github.com/evalplus/evalplus

## E Experimental Results

### E.1 Full Results and Analysis on Math Reasoning

The evaluation results of continual pretraining on the math reasoning target field are shown in Table 3. The data agent and all domain reweighting baseline methods take significantly more GPU training hours than naive training. The main reason is that these domain reweighting methods require a mixture of target-field data and source-field data, while naive training only requires target-field data. By training on more tokens, domain reweighting methods significantly mitigate the catastrophic forgetting problem encountered in naive training, usually outperforming naive training by a large margin on average performance. According to the results, we have the following observations:

**Naive training significantly improves target model performance on the target field but leads to drastic collapse on the capabilities of the source field.** Compared to the base model, the average math reasoning performance increases by an average of 22.77% on the four target models, indicating the effectiveness of training on high-quality in-distribution data for the target field. However, the performance on general benchmarks drops by an average of 11.96%, showing a significant degradation in the source-field model capability. These results further highlight the existence of catastrophic forgetting problems in continual pre-training scenarios, motivating exploration in data mixture and domain reweighting algorithms. While data mixing agent bears an average of 0.57% gap to naive training in the target domain, it outperforms naive training by an average of 11.99% in the source domain, showing a significant mitigation of the catastrophic forgetting problem.

**Domain reweighting algorithms such as RegMix can achieve balanced performance across fields.** According to the results, the RegMix method exhibits a trade-off effect across domains. On the 2-dimensional data reweighting space, it

Table 3: The evaluation results of continual pretraining on the math reasoning target field, reflected on 12 benchmarks. We also separately report the average results on general benchmarks, math reasoning benchmarks, and all benchmarks.

(a) Model performances on the 2-dimensional data reweighting space based on data sources.

| Method | Avg. | General Benchmarks | | | | | | | | | Math Benchmarks | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | MMLU | Hella. | OBQA | Wino. | ARC-C | PiQA | SciQ | LogiQA | Avg. | GSM8K | Minerva | MATH | MathQA | Avg. |
| **LLaMA-DCLM** | | | | | | | | | | | | | | | |
| Base Model | 38.15 | 34.5 | **64.5** | 37.0 | 61.56 | 36.69 | **75.84** | 84.2 | 28.11 | 52.8 | 2.55 | 4.1 | 4.22 | 24.52 | 8.85 |
| Naive Training | 38.51 | 27.11 | 37.0 | 28.2 | 54.22 | 28.58 | 60.28 | 68.7 | 26.88 | 41.37 | 59.21 | **16.16** | **22.85** | 32.96 | 32.80 |
| RegMix | 44.01 | 30.42 | 59.72 | 36.6 | 61.72 | 34.73 | 73.88 | 85.1 | 28.73 | 51.36 | 55.87 | 11.5 | 17.7 | 32.16 | 29.31 |
| DBL | 43.50 | 29.0 | 55.66 | 35.0 | **63.64** | 32.11 | 72.5 | **88.42** | 28.89 | 50.65 | 56.22 | 12.2 | 18.24 | 30.14 | 29.2 |
| DataAgent$_{SFT}$ | 44.95 | 33.81 | 60.23 | 34.2 | 60.43 | 36.26 | 73.28 | 87.3 | 29.33 | 51.86 | 57.84 | 12.7 | 21.3 | 32.73 | 31.14 |
| DataAgent$_{RL}$ | **47.03** | **34.06** | 63.38 | **42.14** | 62.35 | **36.92** | 74.85 | 87.89 | **30.29** | **54.04** | **59.24** | 14.8 | 22.75 | **35.3** | **33.02** |
| **LLaMA-FWE** | | | | | | | | | | | | | | | |
| Base Model | 37.65 | 34.47 | 60.52 | 37.8 | 57.77 | 40.53 | 74.21 | 85.4 | **28.26** | 52.37 | 2.5 | 2.52 | 4.06 | 23.72 | 8.2 |
| Naive Training | 38.51 | 27.25 | 37.03 | 28.4 | 53.51 | 26.96 | 61.1 | 69.6 | 28.11 | 41.5 | **58.91** | 12.58 | **24.7** | **33.94** | **32.53** |
| RegMix | 43.83 | 32.83 | 60.2 | 36.45 | 54.51 | 37.17 | 71.72 | 86.5 | 28.0 | 50.92 | 55.9 | 12.2 | 20.35 | 30.1 | 29.64 |
| DBL | 43.92 | 31.27 | 56.58 | **40.7** | 56.27 | 38.32 | 71.18 | **88.26** | 26.0 | 51.07 | 54.2 | 12.05 | 20.72 | 31.52 | 29.62 |
| DataAgent$_{SFT}$ | 45.23 | **34.65** | **60.83** | 38.28 | 59.3 | **40.8** | **74.6** | 85.6 | 26.96 | **52.63** | 56.26 | 12.19 | 21.92 | 31.32 | 30.42 |
| DataAgent$_{RL}$ | **45.48** | 33.78 | 60.44 | 38.8 | **59.59** | 38.89 | 73.12 | 84.9 | 27.49 | 52.13 | 58.07 | **13.46** | 23.96 | 33.28 | 32.19 |
| **LLaMA-Nemo** | | | | | | | | | | | | | | | |
| Base Model | 38.22 | 34.22 | 64.51 | 37.6 | 59.12 | 36.26 | **75.57** | **88.4** | 26.73 | 52.8 | 2.5 | 4.85 | 5.3 | 23.62 | 9.07 |
| Naive Training | 37.86 | 27.06 | 37.4 | 28.0 | 52.88 | 27.05 | 59.74 | 68.6 | 25.19 | 40.74 | **59.05** | **12.3** | **24.52** | **32.53** | **32.1** |
| RegMix | 44.13 | **35.03** | **65.15** | 35.9 | 59.83 | 36.45 | 72.85 | 86.2 | 28.88 | 52.54 | 49.39 | 10.7 | 19.23 | 30.0 | 27.33 |
| DBL | 42.63 | 33.2 | 64.82 | 34.0 | **62.06** | **39.23** | 70.79 | 78.11 | 24.0 | 50.77 | 45.91 | 10.74 | 20.01 | 28.74 | 26.35 |
| DataAgent$_{SFT}$ | 44.12 | 34.06 | 64.25 | 39.04 | 60.4 | 38.1 | 74.17 | 86.75 | 29.16 | 53.24 | 47.81 | 9.07 | 17.8 | 28.86 | 25.89 |
| DataAgent$_{RL}$ | **45.8** | 34.27 | 63.95 | **39.8** | 61.58 | 38.74 | 74.49 | 86.9 | **29.95** | **53.71** | 54.28 | 10.83 | 22.94 | 31.85 | 29.98 |
| **Pythia-1.4B** | | | | | | | | | | | | | | | |
| Base Model | 33.47 | **30.74** | **52.0** | 33.2 | **57.3** | **28.33** | **70.89** | **79.3** | **27.5** | **47.41** | 1.67 | 4.39 | 2.1 | 14.16 | 5.58 |
| Naive Training | 31.06 | 25.8 | 26.76 | 18.8 | 41.93 | 21.81 | 52.95 | 62.2 | 21.11 | 33.92 | **48.98** | **10.16** | **14.64** | **27.52** | **25.33** |
| RegMix | 34.07 | 29.37 | 48.3 | **33.4** | 43.88 | 21.16 | 65.7 | 64.46 | 25.2 | 41.43 | 40.2 | 6.92 | 7.93 | 22.28 | 19.33 |
| DataAgent$_{SFT}$ | 33.6 | 30.66 | 45.23 | 26.6 | 43.83 | 22.25 | 61.94 | 72.2 | 26.88 | 41.2 | 37.1 | 7.19 | 7.85 | 21.45 | 18.4 |
| DataAgent$_{RL}$ | **35.12** | 30.0 | 48.5 | 30.6 | 41.17 | 25.66 | 65.54 | 75.8 | 24.49 | 42.72 | 40.26 | 7.73 | 8.75 | 22.94 | 19.92 |

(b) Model performances on the 52-dimensional data reweighting space based on the Nvidia domain classifier.

| Method | Avg. | General Benchmarks | | | | | | | | | Math Benchmarks | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | MMLU | Hella. | OBQA | Wino. | ARC-C | PiQA | SciQ | LogiQA | Avg. | GSM8K | Minerva | MATH | MathQA | Avg. |
| **LLaMA-DCLM** | | | | | | | | | | | | | | | |
| Base Model | 38.15 | **34.5** | **64.5** | 37.0 | 61.56 | 36.69 | **75.84** | 84.2 | 28.11 | 52.8 | 2.55 | 4.1 | 4.22 | 24.52 | 8.85 |
| Naive Training | 38.51 | 27.11 | 37.0 | 28.2 | 54.22 | 28.58 | 60.28 | 68.7 | 26.88 | 41.37 | **59.21** | 16.16 | 22.85 | **32.96** | **32.80** |
| RegMix | 44.67 | 34.38 | 62.17 | 38.2 | 61.93 | **36.95** | 74.97 | 87.5 | 29.49 | 53.19 | 55.78 | 10.36 | 15.75 | 28.67 | 27.64 |
| DataAgent$_{SFT}$ | 45.75 | 34.47 | 63.36 | 40.6 | 62.35 | 35.87 | 74.32 | **89.2** | **29.96** | 53.77 | 56.77 | 11.12 | 18.76 | 32.3 | 29.74 |
| DataAgent$_{RL}$ | **46.84** | 32.99 | 62.64 | **41.6** | **63.98** | 36.64 | 73.5 | 89.1 | 31.5 | **53.99** | 59.04 | **16.48** | **22.9** | 31.72 | 32.54 |
| **LLaMA-FWE** | | | | | | | | | | | | | | | |
| Base Model | 37.65 | 34.47 | 60.52 | 37.8 | 57.77 | **40.53** | 74.21 | 85.4 | 28.26 | **52.37** | 2.5 | 2.52 | 4.06 | 23.72 | 8.2 |
| Naive Training | 38.51 | 27.25 | 37.03 | 28.4 | 53.51 | 26.96 | 61.1 | 69.6 | 28.11 | 41.5 | 58.91 | 12.58 | **24.7** | **33.94** | 32.53 |
| RegMix | 43.78 | **35.64** | 57.64 | **39.4** | 57.56 | 38.2 | 67.4 | 85.03 | 29.34 | 51.28 | 53.68 | 10.84 | 20.35 | 30.0 | 28.72 |
| DataAgent$_{SFT}$ | 45.21 | 34.27 | 61.2 | 37.95 | 58.57 | 40.28 | **75.17** | 85.8 | 28.23 | 52.68 | 54.5 | 13.27 | 22.17 | 31.1 | 30.26 |
| DataAgent$_{RL}$ | **45.55** | 32.55 | 58.64 | 35.8 | **59.42** | 39.76 | 73.34 | **87.2** | **29.35** | 52.01 | **58.96** | **14.68** | 23.62 | 33.32 | **32.65** |
| **LLaMA-Nemo** | | | | | | | | | | | | | | | |
| Base Model | 38.22 | 34.22 | 64.51 | 37.6 | 59.12 | 36.26 | **75.57** | 88.4 | 26.73 | 52.8 | 2.5 | 4.85 | 5.3 | 23.62 | 9.07 |
| Naive Training | 37.86 | 27.06 | 37.4 | 28.0 | 52.88 | 27.05 | 59.74 | 68.6 | 25.19 | 40.74 | **59.05** | **12.3** | **24.52** | **32.53** | **32.1** |
| RegMix | 44.86 | **35.68** | **66.4** | 41.4 | **61.56** | 35.82 | 72.4 | **87.53** | 29.34 | **53.77** | 48.17 | 10.91 | 19.03 | 30.04 | 27.04 |
| DataAgent$_{SFT}$ | 45.16 | 34.8 | 64.01 | 38.2 | 60.73 | 36.9 | 73.92 | 87.3 | **29.37** | 53.15 | 52.5 | 10.7 | 20.77 | 32.71 | 29.17 |
| DataAgent$_{RL}$ | **45.9** | 33.89 | 62.7 | **42.6** | 59.82 | **40.0** | 74.86 | 84.34 | 28.29 | 53.31 | 56.78 | 11.82 | 23.61 | 32.03 | 31.06 |
| **Pythia-1.4B** | | | | | | | | | | | | | | | |
| Base Model | 33.47 | 30.74 | **52.0** | 33.2 | **57.3** | **28.33** | **70.89** | **79.3** | 27.5 | **47.41** | 1.67 | 4.39 | 2.1 | 14.16 | 5.58 |
| Naive Training | 31.06 | 25.8 | 26.76 | 18.8 | 41.93 | 21.81 | 52.95 | 62.2 | 21.11 | 33.92 | **48.98** | **10.16** | **14.64** | **27.52** | **25.33** |
| RegMix | 34.64 | 29.6 | 47.02 | **34.78** | 46.32 | 24.76 | 62.83 | 69.3 | 27.03 | 42.71 | 38.28 | 7.24 | 7.51 | 21.1 | 18.53 |
| DataAgent$_{SFT}$ | 34.01 | **30.95** | 46.93 | 32.2 | 43.93 | 21.23 | 62.2 | 68.8 | **27.88** | 41.77 | 38.84 | 7.19 | 7.3 | 20.68 | 18.5 |
| DataAgent$_{RL}$ | **35.25** | 28.8 | 46.97 | 28.8 | 43.61 | 25.91 | 65.3 | 71.31 | 23.36 | 41.76 | 45.44 | 8.1 | 10.01 | 25.38 | 22.23 |

Table 4: The evaluation results of continual pretraining on the code generation target field, reflected on 10 benchmarks. The data is reweighted based on the 2-dimensional domain space.

| Method | Avg. | General Benchmarks | | | | | | | | | Code Benchmarks | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | MMLU | Hella. | OBQA | Wino. | ARC-C | PiQA | SciQ | LogiQA | Avg. | HumanEval | MBPP | Avg. |
| **LLaMA-DCLM** | | | | | | | | | | | | | |
| Base Model | 44.52 | **34.5** | **64.5** | 37.0 | **61.56** | **36.69** | 75.84 | **84.2** | 28.11 | **52.8** | 8.6 | 14.2 | 11.4 |
| Naive Training | 40.08 | 27.6 | 42.96 | 29.37 | 53.1 | 24.76 | 70.5 | 62.95 | 24.46 | 41.96 | **27.3** | **37.8** | **32.55** |
| RegMix | 44.85 | 31.22 | 57.13 | 33.4 | 57.43 | 29.05 | **76.1** | 82.09 | 29.33 | 49.47 | 21.1 | 31.6 | 26.35 |
| DBL | 43.52 | 31.7 | 55.7 | 35.2 | 53.8 | 31.28 | 68.58 | 78.26 | **29.8** | 48.04 | 20.6 | 30.3 | 25.45 |
| DataAgent$_{SFT}$ | 45.07 | 32.69 | 63.43 | 35.0 | 49.88 | 33.46 | 72.85 | 78.66 | 29.61 | 49.45 | 22.4 | 32.8 | 27.6 |
| DataAgent$_{RL}$ | **46.3** | 33.84 | 63.79 | **37.8** | 57.3 | 35.05 | 73.2 | 78.57 | 27.34 | 50.86 | 22.0 | 34.1 | 28.05 |
| **Pythia-1.4B** | | | | | | | | | | | | | |
| Base Model | 38.91 | **30.74** | **52.0** | **33.2** | **57.3** | **28.33** | **70.89** | **79.3** | **27.5** | **47.41** | 4.9 | 4.9 | 4.9 |
| Naive Training | 35.14 | 26.03 | 38.48 | 24.4 | 46.61 | 21.59 | 59.87 | 56.73 | 20.58 | 36.79 | **24.7** | **32.4** | **28.55** |
| RegMix | 38.96 | 28.2 | 47.66 | 28.6 | 52.09 | 26.76 | 64.3 | 69.49 | 24.78 | 42.74 | 20.7 | 27.0 | 23.85 |
| DataAgent$_{SFT}$ | 40.82 | 30.1 | 47.93 | 30.44 | 54.46 | 26.76 | 69.87 | 73.11 | 24.87 | 44.69 | 30.1 | 29.1 | 25.35 |
| DataAgent$_{RL}$ | **41.63** | 29.9 | 46.31 | 31.8 | 56.9 | 27.35 | 69.2 | 78.26 | 23.27 | 45.37 | 23.3 | 30.0 | 26.65 |

outperforms the base model on math reasoning by an average of 18.47%, while largely preserving general capabilities with a mere 2.28% degradation on the corresponding benchmarks. RegMix also outperforms the naive training by 5.03% on the overall average performance. Similar conclusions can be drawn from the results on the 52-dimensional domain space in Table 3b. These results show that the catastrophic forgetting problem can be considerably alleviated by carefully curating data mixtures of source and target fields.

**Data Mixing Agent significantly outperforms other methods in balanced performance across fields.** In Table 3a, for the in-distribution LLaMA-DCLM target model, DataAgent$_{RL}$ outperforms the RegMix results on 7 out of 8 general benchmarks and all 4 math benchmarks. It achieves the best average performance 54.04% and 33.02% on general/math benchmarks, even outperforming the base model in general ability and the naively trained model on math reasoning. These results prove that DataAgent$_{RL}$ can effectively curate the data mixture to improve both general and math reasoning capabilities. With careful domain reweighting, increasing capability on the target field can further enhance performance on the source field. Overall, DataAgent$_{RL}$ achieves 47.03% on average, surpassing RegMix by 3.02%, DBL by 3.53%, and the base model by 8.88%. DataAgent$_{RL}$ also outperforms DataAgent$_{SFT}$ by a large margin of 2.08%. This advantage shows that the empirical guidance presented in the trajectory sampling algorithm is trivial compared to heuristics derived from the broader sampling of data mixing trajectories, underscoring the importance of reinforcement learning with CQL as a crucial step towards capable agents.

**The capabilities of data mixing agents can generalize across target models, source-field data, and domain spaces without retraining.** Though our data mixing agent is trained on the 52-dimensional data reweighting space with trajectories sampled with the DCLM data, it effectively guides domain reweighting for four target models across 2 domain definitions. For example, in Table 3a, DataAgent$_{RL}$ outperforms RegMix by an average of 1.66%, DBL by an average of 2.37% on the two unseen target models: LLaMA-FWE and LLaMA-Nemo, based on the 2-dimensional domain space. In Table 3b, DataAgent$_{RL}$ outperforms RegMix by an average of 1.41% based on the 52-dimensional domain space. These results indicate that Data Mixing Agent learns data- and model-agnostic heuristics from the sampled trajectories that can guide domain reweighting on multiple source-field data distributions, which is crucial to the efficiency of this algorithm, as the feedback collection for sampled data trajectories requires considerable computations. With these generalization capabilities, the agent is still expected to perform well in applications to new target models and source-field data without re-training.

**Data mixing agent is effective in guiding continual pre-training on estimated start state and data mixtures with synthetic source-field data.** We prove this by reweighting domains on the Pythia-1.4B target model with the estimated start state obtained as in Appendix B and source-field data obtained as described in Appendix D.3. In math reasoning, DataAgent$_{RL}$ also outperforms RegMix by 0.59% in Table 3a and 3.7% in Table 3b. However, the preservation on general capabil-

ities significantly drops, with a 4.69% and 5.65% gap on 2-dimensional and 52-dimensional domain spaces. Overall, DataAgent$_{RL}$ still significantly improves average performance compared to the base model, and outperforms RegMix by 1.05% in Table 3a and 0.61% in Table 3b.

### E.2 Full Results and Analysis on Code Generation

We evaluate the Data Mixing Agent's generalization to unseen target fields by directly utilizing the agent trained on the math reasoning field to guide domain reweighting for the code generation field. The results are shown in Table 4. We have the following observations:

**The capabilities of Data Mixing Agent can partially generalize across target fields without retraining.** DataAgent$_{RL}$ achieves the best average performance of 46.3% and 41.63% on the LLaMA-DCLM and Pythia-1.4B target models, outperforming the DBL method by an average of 2.78% and RegMix method by 2.67%. These results prove that heuristics learned in the math reasoning field can be partially transferred to the code generation field without modifying the weights of the agent. However, we observe a degradation in DataAgent$_{RL}$'s advantage over the baseline methods in code generation. For example, DataAgent$_{RL}$ outperforms naive training by 6.22%, while in Table 3a, the advantage is 8.52%. This is mainly due to that applying DataAgent$_{RL}$ to code generation leads to a major 3.18% drop on general benchmarks compared to math reasoning, which indicates the existence of heuristics that are dependent on the target field and the potential misalignment when converting them to a new target field.

**The Data Mixing Agent still demonstrates strong generalization to synthetic source-field data and unseen target fields.** This is validated through continual pre-training in the code generation domain using synthetic data from the Pythia-1.4B model. DataAgent$_{RL}$ outperforms RegMix by 2.63% on general benchmarks, 2.8% on code benchmarks, and 2.67% on average. These results highlight the agent's ability to generalize effectively, enabling its application to scenarios where the source-field data is unavailable and the target model is trained on previously unseen target fields.

### E.3 Full Analysis on Domain Reweighting Trajectories

In this section, we showcase the domain reweighting process guided by Data Mixing Agent to train the LLaMA-DCLM model on the math reasoning field, aiming to provide more intuitions on its actions based on the heuristics and feedback. The trajectories on the 2-dimensional and 52-dimensional domain spaces are provided in Fig. 5, Fig. 6, and Fig. 7. We have the following observations:

**Data Mixing Agents follow a less-to-more trend when adapting the target field data along the data mixing trajectory, but DataAgent$_{RL}$ adopts a more fine-grained approach to achieve superior performance.** In Fig. 5, both the DataAgent$_{RL}$ and DataAgent$_{SFT}$ models show an overall trend to increase data from the target field and decrease data from the source field, but with different strategies. DataAgent$_{SFT}$ shows a radical trend towards more target field data, increasing the DCLM data ratio almost monotonically from about 45% to over 60% during continual pre-training. DataAgent$_{RL}$ adopts a more conservative three-stage strategy:

- **Early warm-up stage**: the agent prioritizes source field data to stabilize training;
- **Mid-training stage**: the agent rapidly increases the use of target field data to enhance performance on the target capability;
- **Final stage**: the agent gradually reintroduces more source field data, with the data distribution stabilizing around the optimal weights identified by RegMix.

As shown in Table 3a, the superior performance of DataAgent$_{RL}$ on both general and math reasoning benchmarks proves the advantage of its subtle domain reweighting strategy. This performance gap between Data Mixing Agents is mainly due to the comprehensive modeling of the heuristic space during reinforcement learning. DataAgent$_{SFT}$ is only fine-tuned on the $\mathcal{T}_{top1}$ trajectories, which mostly model the inductive biases from the $CalculateInductiveScores$ function. DataAgent$_{RL}$ is further optimized on a broad range of trajectories via reinforcement learning, including $\mathcal{T}_{top1}$, $\mathcal{T}_{top100}$, $\mathcal{T}_{top1000}$, and $\mathcal{T}_{top10000}$, with contrastive supervision signals to increase probabilities of actions that improve overall performance and avoid actions that hurt performance measured

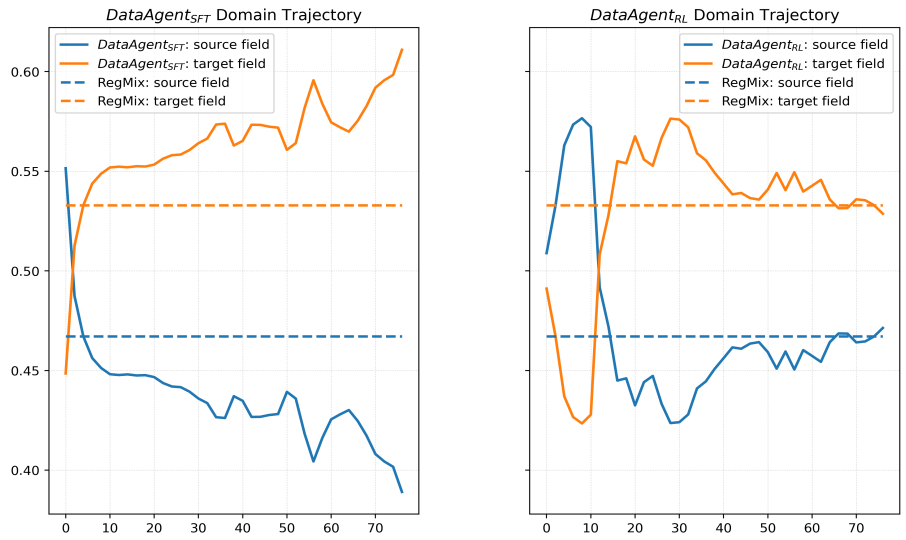


Figure 5: The two data mixing agents' output domain reweighting trajectories based on the 2-dimensional domain space, training on the LLaMA-DCLM model and the math reasoning field. The dashed line denotes the optimal domain distributions determined by RegMix.

by the environment feedback. The visualization of the 52-dimensional domain reweighting trajectories further strengthens the above arguments. In Fig. 6, the DataAgent$_{SFT}$ organizes the target field data from about 60% of the domains to be almost monotonically increasing along the domain reweighting trajectory, while in Fig. 7, the DataAgent$_{RL}$ model introduces more complicated reweighting strategies on about 80% of the domains.

**Data Mixing Agents learn heuristics and perform actions that correspond to human intuitions on the target capabilities.** Our work uses the MMLU evaluation set to represent the general capabilities in the environment. Wettig et al. (2025) summarized the top-3 domains that benefit the MMLU performance: $Science\&Tech.$, $Health$, and $Politics$. In Fig. 7, we observe a significant uplift of the target data distributions in the corresponding domains compared to the RegMix domain distributions: $Science$, $Health$, and $People\&Society$. Wettig et al. (2025) also enumerated domains that can hurt performance on MMLU, such as $Fashion\&Beauty$, while DataAgent$_{RL}$ also conveys an explicit downsampling process in the $Beauty\&Fitness$ domain. These observations further ensure the effectiveness of the learned heuristics, encouraging the discovery of more heuristics via the agent's trajectories. For example, DataAgent$_{RL}$ continuously reduces data from both source and target fields in the $Pets\&Animals$ domain, possibly indicating its lack of importance in enhancing either general or math reasoning capabilities.

### E.4 Full Analysis on Data Efficiency

We explore how efficiently the Data Mixing Agents leverage the source and target field data to improve or preserve model capabilities in the corresponding fields. Training on the mixture of DCLM-100B and the math split of Dolmino-mix-1124 datasets, we record the performance dynamics of the LLaMA-DCLM target model on the general/math evaluation environment with increasing training data (measured in Billion tokens) on the general/math reasoning field. The results are shown in Fig. 8. We have the following observations:

**Data Mixing Agents leverage general field data more efficiently than RegMix, better preserving model capabilities in the source field.** According to the visualization in the general field, the agent methods obtain higher general feedback values from the environment at most token budgets for the source field. The capability measurement for RegMix fluctuates around -2.6, while both data mixing agent models maintain the feedback over -2.575. DataAgent$_{RL}$ further outperforms DataAgent$_{SFT}$ in most cases, with feedback values fluctuating around -2.525, which provides evidence for the heuristics learned during reinforcement learning in preserving the general capabilities. Notably, DataAgent$_{RL}$ shows significantly higher variance in feedback values along the domain reweighting trajectory than both DataAgent$_{SFT}$ and RegMix, reflecting its more active strategies in adjust-

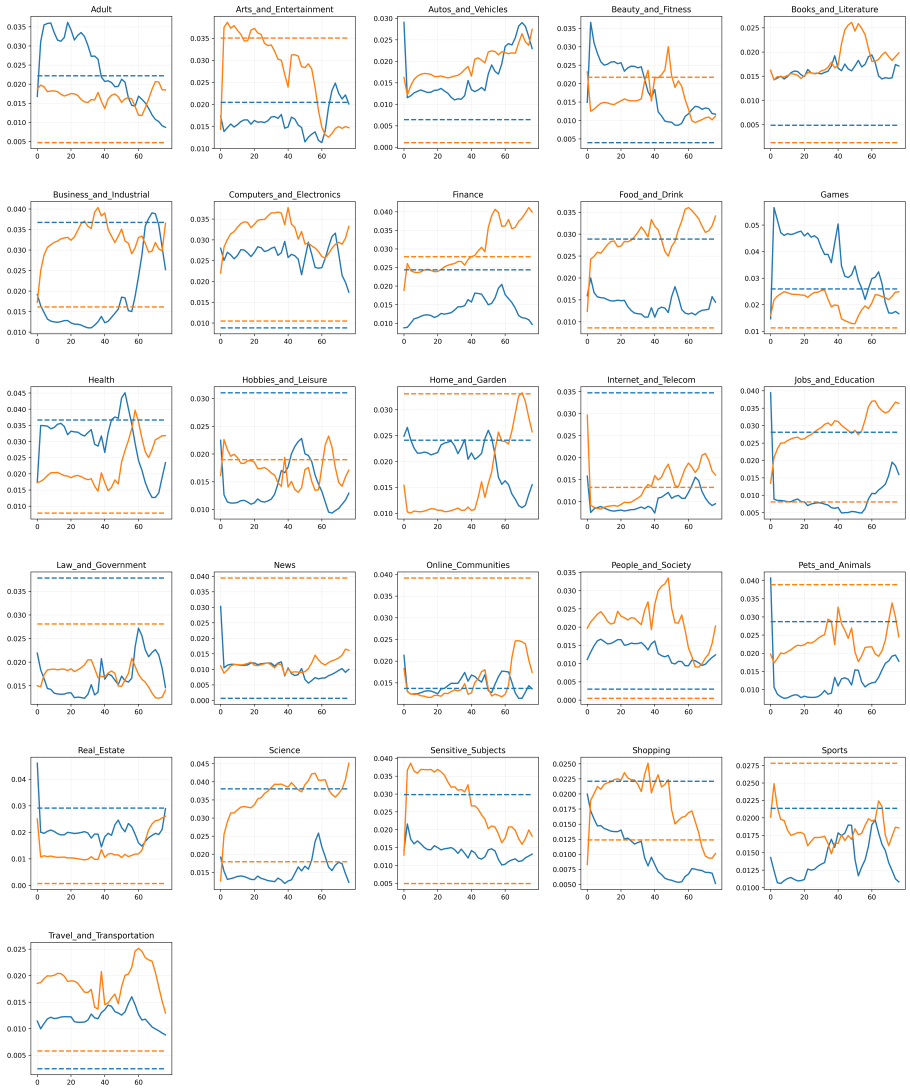


Figure 6: DataAgent$_{SFT}$'s domain reweighting trajectories based on the 52-dimensional domain space, training on the LLaMA-DCLM model and the math reasoning field. The legends within each sub-figure are the same as those of Fig. 5.

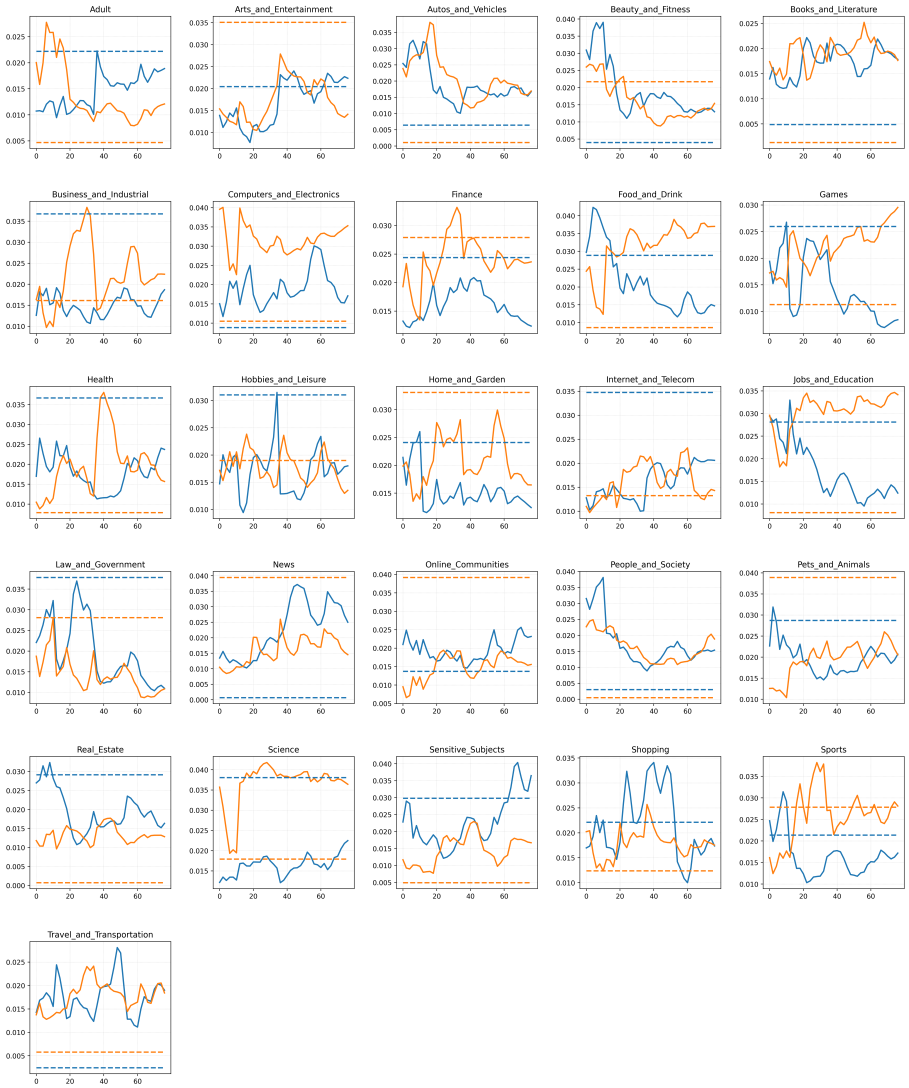


Figure 7: DataAgent$_{RL}$'s domain reweighting trajectories based on the 52-dimensional domain space, training on the LLaMA-DCLM model and the math reasoning field. The legends within each sub-figure are the same as those of Fig. 5.

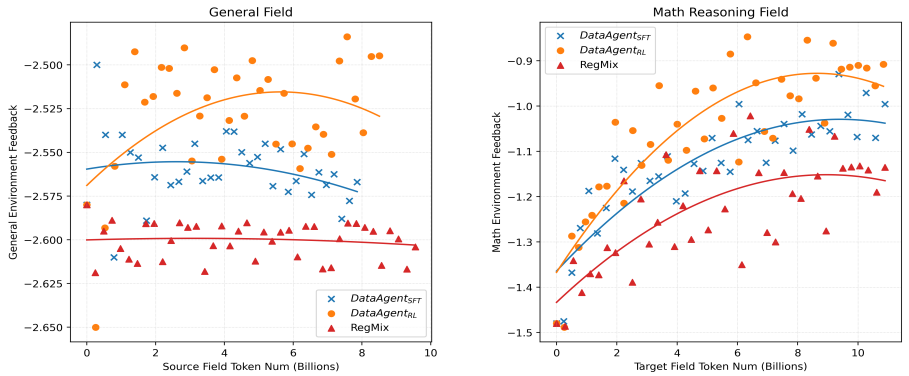


Figure 8: The performance dynamics of the target model on the evaluation environment with increasing training data (measured in Billion tokens) on the corresponding field. We set a total training budget of 10.5B tokens, but DataAgent$_{RL}$ triggers an early stopping at 9.96B tokens, and DataAgent$_{SFT}$ triggers an early stopping at 9.43B tokens.

ing domain reweighting distributions to improve source field capabilities. Its final superior performance on MMLU and the average of general benchmarks (as shown in Table 3a) further indicates the effectiveness of such strategies.

**Data Mixing Agents leverage data from the math reasoning field more efficiently than RegMix, resulting in greater improvements in the target field performance.** As presented in the visualization of the math reasoning field, though all methods show logarithmic-scale improvements from the math reasoning environment, the data mixing agent methods show a faster momentum in increasing general feedback values from the environment at most token budgets for the target field. RegMix performance stabilizes around -1.2 while both data mixing agent methods achieve performance over -1.1. These results show that our method can better arrange the continual pre-training data to improve model capability in the target field. DataAgent$_{RL}$ also outperforms DataAgent$_{SFT}$ with the optimized feedback values over -1.0. The leading performance of DataAgent$_{RL}$ on both general and math reasoning fields proves its effectiveness in coordinating the source and target field data to improve performance on multiple target capabilities.

**Data Mixing Agents achieve balanced continual pre-training performance with less reliance on data from the source field.** As described in Fig. 8, while we set a total training budget of 21B tokens, DataAgent$_{RL}$ triggers an early stopping at 19.92B tokens, and DataAgent$_{SFT}$ triggers an early stopping at 18.86B tokens, due to the exhaustion of the target field data. These results show that the data mixing agent can achieve superior performance than RegMix on both the general and math reasoning fields while relying on 2.14B fewer tokens in the source field, further proving the efficiency of their domain reweighting process.

### E.5 Consistency between Rewards and Model Performance

To examine whether the proposed environment feedback provides a consistent and reliable signal for downstream model performance, we analyze multiple checkpoints along the LLaMA-DCLM training trajectory in the math domain. Specifically, we evaluate five checkpoints on different training stages (with 3.8B to 19.92B trained tokens) and report their corresponding environment feedback signals (denoted as $Env_{general}$ and $Env_{math}$) and average benchmark performance on general-domain and math tasks (denoted as $bench\_general\_avg$ and $bench\_math\_avg$).

The results are shown in Table 5. Across these checkpoints, we observe a strong monotonic relationship between the environment feedback and final benchmark performance. The Pearson's correlation coefficient between $Env_{general}$ and bench_general_avg is 0.943, while the correlation between $Env_{math}$ and bench_math_avg reaches 0.919. These high correlations indicate that the environment feedback closely tracks the model's eventual evaluation accuracy, validating its consistency as a training signal. This result demonstrates that the reward used by the data mixing agent is well aligned with downstream task performance, thereby supporting the reliability and effectiveness of our method.

## F Related Work

### F.1 Continual Pre-training

Continual pre-training is an effective and efficient method for adapting LLMs to new target fields where the pre-training data do not align well, such as knowledge-intensive and complex-reasoning tasks. In math reasoning, DeepSeekMath (Shao et al., 2024) was initialized with the DeepSeek-Coder (Guo et al., 2024) models and continually trained on 500B tokens of high-quality math-related data. In code generation, the Qwen2.5-Coder (Hui et al., 2024) is based on the Qwen2.5 foundation model and continuously trained on 3.64T tokens of data in the code field. Continual pre-training is also used in other fields such as finance (Xie et al., 2024), system research (Lin et al., 2025), and medicine (Tu et al., 2024).

The catastrophic forgetting problem is widely encountered in continual pre-training works (Hui et al., 2024; Lin et al., 2025; Luo et al., 2023; Yang et al., 2024). Existing works mostly curate mixtures of data from the target field and data from the original field to obtain balanced performance. For example, Qwen2.5-Coder manually determined an optimal data mixing recipe of 7:2:1 in code data, text data, and math data for the Qwen2.5-Coder training dataset, leading to over 20% improvement in average performance on multiple fields compared to training solely on code data.

| Training Tokens | $Env_{general}$ | $Env_{math}$ | bench_general_avg | bench_math_avg |
|---|---|---|---|---|
| 3.8B | -2.558 | -1.486 | 52.96 | 10.02 |
| 7.64B | -2.564 | -1.058 | 51.08 | 20.70 |
| 12.84B | -2.591 | -0.896 | 48.00 | 32.95 |
| 16.52B | -2.473 | -0.950 | 58.22 | 23.10 |
| 19.92B | -2.500 | -0.902 | 54.04 | 33.02 |

Table 5: Environment feedback signals and benchmark performance at different training stages in the math domain, along the LLaMA-DCLM trajectory.

### F.2 Data Re-weighting in Pre-training

Domain reweighting is an emerging research field that aims to develop an optimal data mixing strategy for the fixed data mixture to achieve the best possible performance on the target model (Xie et al., 2023; Liu et al., 2024b; Xia et al., 2023; Luo et al., 2024). Doremi (Xie et al., 2023) trains a reference model based on initial domain weights, which is used to guide the training of another proxy model with the group DRO (Sagawa et al., 2019) algorithm to determine the optimal domain weights for the target model. RegMix (Liu et al., 2024b) trained large quantities of small proxy models on random domain distributions, then evaluates these models on the target benchmarks. The best data mixing recipe is determined by fitting a regression model to these data and selecting distributions that lead to the highest scores. Other works focus on balancing the loss of multiple target fields to achieve balanced optimization (Xia et al., 2023; Luo et al., 2024). For example, Xia et al. (2023) proposed a batch loading algorithm that loads training data from each domain in proportion to its corresponding rate of loss reduction, which increases the future domain distributions for domains that have slow loss reduction.

Recent works have also explored the effect of domain space definition on data reweighting performance (Wettig et al., 2025; Rukhovich et al., 2025; Diao et al., 2025; Xi et al., 2025), strengthening the importance of carefully defined domains. For example, previous data mixing methods mostly utilized the default domain space defined by data sources. Wettig et al. (2025) carefully defined a 24-dimensional domain space from both the topic (e.g., Science&Tech, Fashion&Beauty) and format (e.g., Academic writing, Content listing) perspectives, and re-organized the training data into these domain spaces. Extensive data mixing experiments on these novel domain spaces showed their effectiveness in improving model training performances compared to the source-based domain space. Inspired by their success, we also train the Data Mixing Agent based on these superior ways of domain space definition.