OLMIX: A FRAMEWORK FOR DATA MIXING THROUGHOUT LM DEVELOPMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Data mixing—determining the ratios of data from different domains—is a first-order concern for training language models (LMs), but existing mixing methods have poorly understood design choices and assume that the set of domains remain fixed throughout development. We present OLMIX, a framework that addresses two challenges encountered during LM development. First, the configuration space for developing a mixing method is not well understood—design choices across existing methods lack justification or consensus and overlook practical issues like data constraints. We conduct a comprehensive empirical study of this space, identifying which design choices lead to a strong mixing method. Second, the domain set evolves throughout LM development as datasets are revised and expanded—a problem setting largely unaddressed by existing works. We study how to efficiently recompute the mixture after the domain set is updated, given an existing mix from before the update. We introduce mixture reuse, a mechanism that reuses existing relative ratios and recomputes ratios only for domains affected by an update. Over a sequence of five domain-set updates mirroring real-world LM development, mixture reuse matches the performance of fully recomputing the mix after each update with 74% less compute and improves over training without mixing by 11.6% on downstream tasks.

1 INTRODUCTION

Modern language models (LMs) are trained on datasets composed of many domains, such as web text, code, and PDFs. The composition of these domains is crucial for strong downstream performance, making data mixing a first-order component of LM development (Grattafiori et al., 2024; Qwen et al., 2024; Olmo et al., 2025). However, naively searching for a good mix is computationally inefficient, requiring many training runs—possibly thousands of GPU hours—to assess performance. This has resulted in a growing literature on data mixing methods that aim to find strong mixtures systematically with less compute (Xie et al., 2023; Fan et al., 2024; Chen et al., 2025a), *inter alia*. Many mixing methods that achieve promising results follow a common offline schema (Liu et al., 2025b;a; Ye et al., 2025): 1. train a set of smaller proxy models on different mixtures (a "swarm"), 2. fit a regression model on this swarm to predict performance from mixture weights, and 3. select a final mix that optimizes predicted performance.

P1: The mixing configuration problem (§3). At the start of LM development, we must configure a mixing method on an initial set of domains. However, the configuration space of existing methods is poorly understood; design choices are often unjustified, contradict across methods, or fail to address practical issues like data constraints. As a result, practitioners have conflicting or little guidance on which choices lead to a strong mixing method.

In this paper, we identify two challenges in applying data mixing methods both at the *start of* and *throughout* LM development (Figure 1). We present our framework OLMIX, which we describe alongside these motivating problems:

The first component of OLMIX addresses P1 through a comprehensive empirical study of mixing configurations within the offline schema. Specifically, we resolve conflicting guidance in prior work by answering the following research questions:

RQ1 *What is the smallest proxy model size that generalizes to larger target models?*

RQ2 *What is the minimum number of proxy runs (swarm size) needed to learn an effective mix for a given domain set?*

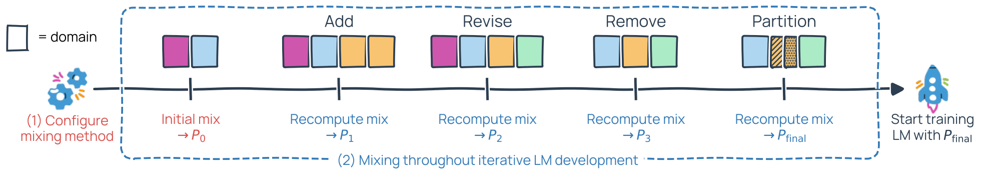
= domain
(1) Configure mixing method
Add
Revise
Remove
Partition
Initial mix → P_0
Recompute mix → P_1
Recompute mix → P_2
Recompute mix → P_3
Recompute mix → P_{final}
Start training LM with P_{final}
(2) Mixing throughout iterative LM development

Figure 1: Two problems with data mixing encountered during LM development: (1) How to best configure your mixing method? (2) How to efficiently mix under evolving domain sets?

RQ3 *Is there an optimal family of regression models for predicting mix performance?*

We further address an unexplored but practical setting:

RQ4 *How can we perform data mixing under finite data constraints?*

These findings inform our base mixing method, OLMIXBASE (Alg. 1), which we use to address P2.

P2: The evolving domain problem (§4). Existing mixing literature assumes a fixed domain set (Albalak et al., 2024), but real-world LM development is iterative. For example, the datasets used throughout OLMo 1–3 (Olmo et al., 2025) and SmolLM 1–3 (Bakouch et al., 2025) undergo various changes: datasets are repeatedly added, removed, revised, and partitioned (Wettig et al., 2025). As a result, mixtures must be recomputed as the domain set evolves. However, fully recomputing from scratch (e.g., using OLMIXBASE) after each domain update becomes increasingly expensive as practitioners make more updates to their datasets. Instead, we may be able to leverage information from historical mixtures, motivating a new problem:

RQ5 *Is there an efficient strategy for recomputing the data mix after a domain update?*

The second component of OLMIX introduces *mixture reuse*, a recomputation strategy that reuses past mixing ratios to propose strong mixes while remaining computationally efficient over the course of LM development. Mixture reuse freezes the relative weights of unchanged domains while recomputing the domains that have evolved. We theoretically analyze when mixture reuse matches full recomputation, showing that the performance gap depends on 1) how much optimal weights shift after domains are updated and 2) a coupling effect between domains (i.e., whether domains impact the same downstream tasks). Finally, we evaluate mixture reuse in an end-to-end case study that mirrors real-world LM data development, where the domain set evolves through a sequence of 5 updates culminating in 64 domains. When training 1B-parameter models on 100B tokens, mixture reuse improves over the natural distribution by 11.6% and captures 95% of full recomputation's gains while requiring 74% fewer proxy runs.

2 RELATED WORK

We provide an extended related work in Appendix A. A vast body of mixing methods follows the offline schema (Liu et al., 2025a; Ye et al., 2025; Ge et al., 2025b; Kang et al., 2025; Diao et al., 2025; Que et al., 2024; Wettig et al., 2025; Belenki et al., 2025; Held et al., 2025). Other mixing approaches dynamically explore mixtures during a proxy run (Fan et al., 2024; Xie et al., 2023), and online mixing methods (Chen et al., 2023; 2025a; Jiang et al., 2024; Albalak et al., 2023) dynamically adjust mixtures during final training. Given the prevalence of the offline schema, our work studies its key design choices.

Real-world LM development involves iterative refinement of training data. Works like DCLM (Li et al., 2024), Dolma Soldaini et al. (2024), and FineWeb (Penedo et al., 2024) document domain updates. Projects like OLMo 1–3 (Olmo et al., 2025) and SmolLM 1–3 (Bakouch et al., 2025) showcase how training data evolves over time through these updates. Motivated by these works, OLMIX views data mixing as an ongoing process throughout LM development.

3 THE MIXING CONFIGURATION PROBLEM

We present the first component of OLMIX: a comprehensive empirical study of the configuration space of mixing methods. In §3.1, we formalize the data mixing problem and describe the offline mixing schema. Then, we present our empirical study of key design choices and our findings (§3.2). These findings inform OLMIXBASE (§3.3), the mixing method we used throughout development.

3.1 BACKGROUND

3.1.1 THE DATA MIXING PROBLEM

Our goal is to determine ratios over training data domains that result in strong downstream performance.

Domain set and mixes. Let $\mathcal{D} = \{D_1, D_2, ..., D_m\}$ be a set of m domains, where each domain D_i has a training dataset of size N_i tokens. A domain is a group of data, ranging from coarse-grained **sources** (defined by data provenance) to fine-grained **topics** (semantically coherent partitions of a source) (Wettig et al., 2025). We specify a data mixture via a probability vector $p \in \triangle^{m-1}$, such that training an LM with R total tokens according to p uses a dataset with $p_i \cdot R$ tokens from D_i for each domain i.

Model and evaluation. We train a target LM of S parameters for R tokens on p, denoted as $\text{LM}(S, R, p)$. We then evaluate this model on a suite of n downstream tasks. We measure the performance $f_i(\text{LM}(S,R,p))$ on each task i in terms of bits-per-byte (BPB), the negative log likelihood of the correct answer normalized by answer length in UTF-8 bytes. Heineman et al. (2025) showed that BPB can be used for decision-making at small model scales and Huang et al. (2024) showed that BPB sets correlate with downstream performance across capabilities and model families.

Goal. Given a domain set $\mathcal{D}$ and target training configuration of S parameters and R tokens, we aim to find a mixture $p^\star$ that minimizes the average BPB across all downstream tasks—that is, minimizing $\frac{1}{n}\sum_{i=1}^{n} f_i(\text{LM}(S,R,p))$.

3.1.2 THE OFFLINE MIXING SCHEMA

Many existing methods follow an offline mixing schema; these methods are also described as "function fitting-based offline methods" in Liu et al. (2025b)'s survey. We describe the three steps of this schema.

Step 1: Swarm construction. We sample the space of possible mixes by training a "swarm" of K small proxy models of size S_{small} on R_{small} tokens, each with different mixture weights, $p^1, p^2, ..., p^K \in \triangle^{m-1}$ sampled from some distribution $\mathcal{P}$. We evaluate each proxy model on the downstream task suite to obtain $y_{ij} := f_i(\text{LM}(S_{\text{small}}, R_{\text{small}}, p^j))$, the performance of the jth proxy model on the ith task, for all tasks over the entire swarm. Altogether, this creates a dataset $\{(p^j, \{y_{ij}\}_{i=1}^n)\}_{j=1}^K$ of mixture weights paired with their performance across all tasks.

Step 2: Regression model. We fit regression models using the above dataset to capture the relationship between the mixture weights and downstream performance. We learn a function $\hat{f} \in \mathcal{F}$, where $\hat{f}(p) \approx \frac{1}{n}\sum_{i=1}^{n} f_i(\text{LM}(S_{\text{small}}, R_{\text{small}}, p))$. This enables us to predict the average downstream BPB of a proxy model trained on any candidate mix p.

Step 3: Mixture optimization. We propose a mix by solving an optimization problem that uses the regression model $\hat{f}$ as a surrogate for true performance. We solve $\underset{p \in \mathcal{S}}{\text{minimize}} \hat{f}(p)$, where $\mathcal{S} \subseteq \triangle^m$ denotes the feasible set, which may be the full probability simplex or a restricted subset.

3.2 STUDY: CONFIGURING A MIXING METHOD

Key design choices in the offline mixing schema are not well understood in existing literature. We conduct a comprehensive study on each design choice and use our findings to develop OLMIXBASE (Algorithm 1). We present selected findings, with full findings in Appendix B.

3.2.1 EXPERIMENTAL SETUP

Data. We use DCLM (Li et al., 2024) partitioned into 24 topic-based domains using WebOrganizer (Wettig et al., 2025). In Appendix C, we also provide results for mixing over data sources.

Model. We train 1B parameter decoder-only transformer models using Olmo 2 architecture (OLMo et al., 2024) to 100B tokens (5x Chinchilla). We use a batch size of 512, sequence length of 4096, and max learning rate of 0.0018. See Appendix G for more details.

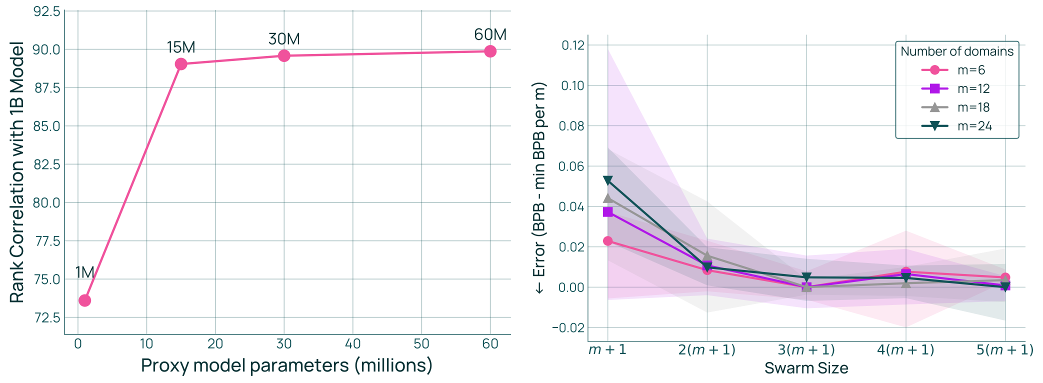

Figure 2: Left: Correlation in performances between proxy models and 1B target models fixing mixture p. Proxy models with over 15M parameters achieve strong rank correlation (§3.2.2). Right: Error vs. Swarm Size. Curves collapse across different m, indicating $\mathcal{O}(m)$ runs are needed (§3.2.3).

Evaluation. We measure the BPB (bits-per-byte) over gold responses on 52 downstream tasks spanning math, code, and commonsense QA. For QA tasks, the gold continuation is the answer text; for math and code tasks the gold continuation is a human-written response. We treat subtasks (e.g. MMLU or coding language subsets) as standalone tasks when taking an macro-average of BPB scores. See Table 10 for the entire list of tasks.

3.2.2 RQ1: Proxy Model Size

What is the smallest proxy model size (the number of parameters, $S_{\textbf{small}}$) such that decision-making generalizes to larger target models? Proxy models must provide reliable signal that transfers to the target model's scale while ideally being small so that computational costs are low. Existing methods use sizes ranging from 1M to 400M parameters, making it unclear what size is sufficient.

We train proxy models of varying sizes (1M, 15M, 30M, 60M) with 5x Chinchilla multiplier paired with 1B target models on the same mixtures and measure rank correlation of their downstream performance. High correlation indicates that rankings at the proxy scale predict rankings at the target scale, ensuring reliability of proxy swarm decisions.

Figure 2 (left) shows that proxy models with >15M parameters achieve strong rank correlation ($\rho > 89$). Notably, while RegMix (Liu et al., 2025a) recommends using a 1M proxy configuration, our results show 1M models achieve only $\rho = 73$, suggesting that they are unreliable for guiding mixture decisions.[1]

3.2.3 RQ2: Swarm Size

How many proxy runs K do we need to learn a good mix on m domains? The swarm size directly controls the cost-performance tradeoff: more runs increase computational cost but improve mix quality. Existing methods use fixed swarm sizes ranging from 20 to over 500 proxies, and few relate these choices to the number of domains m.

For a fixed number of domains m, we run a sweep over swarm sizes $K = c(m+1)$ for linear multiples $c = 1,2,3,4,5$. We perform this sweep for $m = 6,12,18,24$ domains and 3 seeds. For each m, we define error as the difference between the BPB of a proposed mix and the best BPB achieved achieved by any mix found in the corresponding sweep for that m. We evaluate on 30M proxy models and verify that the same trends hold when evaluating at the 1B scale (Figure 8 in Appendix C).

Figure 2 (right) shows that sample complexity is linear in m: the error curves across m collapse when plotted against the linear multiple c. This reveals that $\mathcal{O}(m)$ runs are sufficient for strong performance—a finding that contradicts prior assumptions of quadratic scaling (Ye et al., 2025; Ge et al., 2025b) and provides practitioners with a concrete prescription for allocating compute.

[1] Investigating the public RegMix code, we found their 1M implementation is closer to 15M, which our results suggest is a good proxy size.

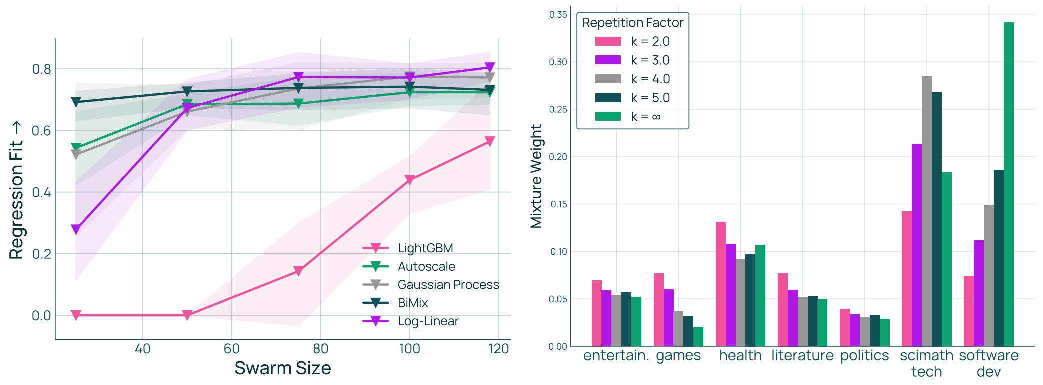

Figure 3: Left: Fit versus Swarm Size across Regression Models. Different models require different amounts of data, and the log-linear model achieves the best fit (§3.2.4). Right: Proposed Mixture Weights versus Repetition Factor. Capping repetition (k) leads to different proposed mixes (§3.2.5).

3.2.4 RQ3: Regression Model Family

Is there an optimal family of regression models ($\mathcal{F}$) for predicting mix performance? The regression model is central: proposed mixes are optimized directly using its predictions. Each existing method proposes a different model and reports strong fit, making it unclear which to adopt.

We compare five regression families: LightGBM (Liu et al., 2025a), Gaussian Process (Chen et al., 2025b), BiMix (Ge et al., 2025b), AutoScale (Kang et al., 2025), and log-linear (Ye et al., 2025). We measure the regression fit across regression models and swarm sizes $K = 25, 50, 75, 100, 125$ over 3 seeds.

We discover that swarm size is a key confounding factor: different models excel at different swarm sizes (Figure 3 left), potentially explaining the lack of consensus. Notably, log-linear models (Ye et al., 2025) achieve the best regression fit overall across swarm sizes. We verify this choice via downstream evaluation (training 1B models) at $K = 125$, where the log-linear model's proposed mixes again achieves the best performance (Figure 9 in Appendix C).

3.2.5 RQ4: Data-Constrained Settings

How do we mix under finite data constraints? In practice, LM training is often data-constrained: the target model's requested training tokens R exceed the available data. In this regime, some mixtures cause excessive sample repetition, degrading performance (Muennighoff et al., 2025). For example, a mixture that proposes 40% code even when code comprises only 5% of the available data would oversample code 8 times. Existing mixing methods assume compute-constrained settings with effectively infinite data and provide no way to control repetition.

How might we adapt the offline mixing schema to a data-constrained setting? We study the effects of enforcing repetition constraints in the mixing objective (minimizing BPB); that is, recasting the third step of the offline schema as a constrained optimization problem with an additional term $p_j \leq kN_j/R$, for $j \in [m]$. k is the repetition factor, the max allowable repetitions for any document.

Figure 3 (right) shows how the proposed mix is impacted smoothly by repetition constraints during mix optimization. As k increases, high-utility domains rapidly increase while low-utility domains decrease, showing that repetition constraints significantly impact proposed mixes. This suggests data mixes should be optimized in accordance with a target training budget.

3.3 Configuring OlmixBase

Our results in §3.2 let us define OlmixBase, a concrete configuration of the offline mixing schema (Table 2; Algorithm 1). Selected configurations resulting from our findings are highlighted in blue.

Algorithm 1 OLMIXBASE

```
1: Input: Domains $\mathcal{D}=\{D_1,\ldots,D_m\}$, swarm size $K=\mathcal{O}(m)$, repetition factor $k$, requested tokens $R$.
2: Sample mixes $p^1,\ldots,p^K \sim \mathcal{P}$ on $\mathcal{D}$.
3: Train proxy models $S_{\text{small}} > 15\text{M}$ parameters on these mixes, and evaluate on downstream tasks to
   get a dataset of mixes and performance, $\{(p^j,\{y_{ij}\}_{i=1}^n\}_{j=1}^K$, where $y_{ij}=f_i(\text{LM}(S_{\text{small}},R_{\text{small}},p^j))$.
4: for $i \in [n]$ do
5:     Use $\{(p^j,y_{ij})\}_{j=1}^K$ to fit the log-linear model $\hat{f}_i(p)=c_i+\exp(A_i^\top p)$, where $c_i \in \mathbb{R}^+$, $A_i \in \mathbb{R}^m$.
6: end for
7: Solve the optimization problem:
```

$$\text{minimize}_{p\in\triangle^{m-1}} \frac{1}{n}\sum_{i=1}^n \hat{f}_i(p) \qquad \text{subject to} \quad p_j \leq \frac{kN_j}{R}\ \forall j\in[m] \tag{1}$$

```
8: Return $p^\star$, the solution to (1).
```

4 THE EVOLVING DOMAIN PROBLEM

OLMIXBASE assumes a fixed domain set $\mathcal{D}$. Here, we study the *evolving domain problem* encountered throughout LM development. We first define the problem setting (§4.1). Then, we present our method, FULLMIXTUREREUSE (§4.2), and theoretical analysis of it (§4.3). We introduce PARTIALMIXTUREREUSE as a coupling-aware extension (§4.4), and we evaluate these methods empirically in §5.

4.1 PROBLEM SETUP

During LM development, the domain set evolves from $\mathcal{D}$ to an updated set $\mathcal{D}'=\{D'_1,\ldots,D'_{m'}\}$ of size m', where each domain D'_i now has N'_i tokens. We use $q\in\triangle^{m'-1}$ to express data mixtures over $\mathcal{D}'$. In practice, domain updates are typically localized, affecting only a subset of domains while leaving others unchanged. To capture this structure, we partition the domain sets as $\mathcal{D}=[\mathcal{D}_1,\mathcal{D}_2]$ and $\mathcal{D}'=[\mathcal{D}_1,\mathcal{D}'_2]$, where $\mathcal{D}_1$ is the **unaffected domain set**, and $\mathcal{D}_2$ is the **affected domain set** that is transformed into $\mathcal{D}'_2$.

In studying the data development practices in long-run LM development projects like OLMo 1–3 (Olmo et al., 2025) and SmolLM 1–3 (Bakouch et al., 2025), we identified four patterns of domain updates, which we call **domain update operators**, that describe how $\mathcal{D}_2$ transforms into $\mathcal{D}'_2$:

- `Add`: $\mathcal{D}_2=\emptyset$ and $\mathcal{D}'_2$ contains the newly added domains. This occurs when new datasets are created.
- `Remove`: $\mathcal{D}_2$ contains one or more domains and $\mathcal{D}'_2=\emptyset$. This can occur, for instance, when domains are discarded due to low utility.
- `Partition`: $\mathcal{D}_2$ consists of one domain that is split into multiple subdomains in $\mathcal{D}'_2$ such that $\mathcal{D}_2=\bigcup_{D'_i\in\mathcal{D}'_2} D'_i$. Partitioning is commonly used to obtain finer-grained mixtures, which can improve performance (Wettig et al., 2025; Diao et al., 2025; Ge et al., 2025a).
- `Revise`: $\mathcal{D}_2$ is one domain that is modified to produce the domain in $\mathcal{D}'_2$. This occurs when contents of the dataset are modified, such as samples being rewritten (Team et al., 2025; Maini et al., 2024).

Problem We have an existing mixture $\tilde{p}\in\triangle^{m-1}$ over the current domain set $\mathcal{D}$ proposed through a procedure like OLMIXBASE. But now, the domain set has evolved from $\mathcal{D}$ to $\mathcal{D}'$, and our goal is to compute an updated optimal $q^\star\in\triangle^{m'-1}$ on $\mathcal{D}'$:

$$\text{minimize}_{q\in\triangle^{m'-1}} \frac{1}{n}\sum_{i=1}^n f_i(\text{LM}(S,R,q)) \qquad \text{s.t.} \quad q_j \leq \frac{kN'_j}{R} \quad \forall j\in[m'] \tag{2}$$

4.2 MIXTURE REUSE

Our goal is to leverage the existing mixture $\tilde{p}$ to efficiently compute a strong mixture on an updated domain set $\mathcal{D}'$ without full recomputation over all m' domains. Our core idea is to freeze the relative ratios among the unaffected domains according to $\tilde{p}$, aggregate them into a single *virtual domain*, and only recompute its total weight along with the weights of the affected domains $\mathcal{D}_2'$. This reduces the dimensionality of the optimization from m' to $1+|\mathcal{D}_2'|$.

For example, suppose $\mathcal{D}$ contains $m=3$ domains with mixture $\tilde{p}=[0.25,0.25,0.5]$, and $\mathcal{D}'$ is formed by adding one new domain. Instead of learning a 4-dimensional mixture, we learn a 2-dimensional mixture over the virtual domain and the new domain. If that mixture is $[0.4,0.6]$, we can unravel it to induce the mixture over the $m'=4$ domains:

$$q=0.4\cdot[0.25,0.25,0.5]\cup[0.6]=[0.1,0.1,0.2,0.6].$$

4.2.1 Formalizing the Mixture Reuse Problem

We formalize our reuse mechanism by presenting the mixture reuse problem.

First, we introduce notation to handle domains we will reuse versus recompute. Define $\mathcal{D}_{\text{fix}}:=\mathcal{D}_1$ as the unaffected domains whose relative ratios we will keep fixed from $\tilde{p}$, and $\mathcal{D}_{\text{comp}}:=\mathcal{D}_2'$ as the affected domains whose ratios we recompute. We remap these variables because, as seen later in §4.4, we may choose to recompute unaffected domains.

We partition any mix q on $\mathcal{D}'$ as $q=[\rho q_{\mathcal{D}_{\text{fix}}},(1-\rho)q_{\mathcal{D}_{\text{comp}}}]$, where $\rho\in[0,1]$ is the total weight on the unaffected domains, and $q_{\mathcal{D}_{\text{fix}}}\in\triangle^{|\mathcal{D}_{\text{fix}}|-1}$ and $q_{\mathcal{D}_{\text{comp}}}\in\triangle^{|\mathcal{D}_{\text{comp}}|-1}$ are the relative mixes within each domain subset. Similarly, we define $\tilde{p}_{\mathcal{D}_{\text{fix}}}\in\triangle^{|\mathcal{D}_{\text{fix}}|-1}$ as the normalized ratios from $\tilde{p}$ restricted to $\mathcal{D}_{\text{fix}}$.

With this notation, the mixture reuse problem is the following approximation of (2):

$$\text{minimize}_{q\in\triangle^{m'-1}}\frac{1}{n}\sum_{i=1}^{n}f_i(\text{LM}(S,R,q))\qquad\text{s.t.}\quad q_j\le\frac{kN_j'}{R}\quad\forall j\in[m'],\quad q_{\mathcal{D}_{\text{fix}}}=\tilde{p}_{\mathcal{D}_{\text{fix}}}\tag{3}$$

The new constraint $q_{\mathcal{D}_{\text{fix}}}=\tilde{p}_{\mathcal{D}_{\text{fix}}}$ enforces that the relative ratios among $\mathcal{D}_{\text{fix}}$ are the same as in $\tilde{p}$. The mixture reuse problem hence presents a tradeoff, which we analyze in §4.3: it has reduced dimensionality due to the constraint but is an approximation of the original problem.

4.2.2 FullMixtureReuse

We now describe FullMixtureReuse, our approach for solving the mixture reuse problem (3) using OlmixBase. To handle the constraint $q_{\mathcal{D}_{\text{fix}}}=\tilde{p}_{\mathcal{D}_{\text{fix}}}$, we perform a change of variables. We aggregate all domains in $\mathcal{D}_{\text{fix}}$ into a single virtual domain and define $r=[\rho,(1-\rho)q_{\mathcal{D}_{\text{comp}}}]\in\triangle^{|\mathcal{D}_{\text{comp}}|}$, where ρ is the weight on the whole of $\mathcal{D}_{\text{fix}}$ and the remaining entries correspond to $\mathcal{D}_{\text{comp}}$. Any r, given $\tilde{p}_{\mathcal{D}_{\text{fix}}}$, can be expanded into a mixture q over $\mathcal{D}'$ that satisfies the constraint. This transforms the mixture reuse problem into standard mixing over $1+|\mathcal{D}_{\text{comp}}|$ domains. We can now apply OlmixBase (Algorithm 1) over r, which requires only $\mathcal{O}(|\mathcal{D}_{\text{comp}}|)$ proxy runs instead of $\mathcal{O}(m')$:

Step 1: Swarm construction. We sample K mixture weights $r^1,\ldots,r^K\in\triangle^{|\mathcal{D}_{\text{comp}}|}$ For each r^j, we expand it into q^j, train a proxy model using q^j, and evaluate to obtain performance $\{y_{ij}\}_{i=1}^{n}$.

Step 2: Regression model. As in OlmixBase, for each task $i\in[n]$ we fit a log-linear regression model: $\hat{g}_i(r)=d_i+\exp(B_i^\top r)$, where $d_i\in\mathbb{R}^+,B_i\in\mathbb{R}^{|\mathcal{D}_{\text{comp}}|}$.

Step 3: Mixture optimization. Solve:

$$\text{minimize}_{r\in\triangle^{|\mathcal{D}_{\text{comp}}|}}\frac{1}{n}\sum_{i=1}^{n}\hat{g}_i(r)\qquad\text{s.t.}\quad\rho\le\min_{j\in\mathcal{D}_{\text{fix}}}\left\{\frac{kN_j'}{R\tilde{p}_j}\right\},\ r_j\le\frac{kN_j'}{R}\quad\forall j\in\mathcal{D}_{\text{comp}}\tag{4}$$

Denote the solution $r^\star(\tilde{p}_{\mathcal{D}_{\text{fix}}})$ and expand it to be $q^\star(\tilde{p}_{\mathcal{D}_{\text{fix}}})$ on $\mathcal{D}'$. The full approach is in Algorithm 2.

4.3 Theoretical Analysis

FullMixtureReuse reduces costs by reusing existing ratios, but when does it match versus degrade compared to full recomputation? We theoretically analyze this, finding that performance depends on (1) how much the optimal mix changes due to the domain update, and (2) a coupling effect between $\mathcal{D}_{\text{fix}}$ and $\mathcal{D}_{\text{comp}}$. Importantly, we empirically validate our theory in Appendix F: **we measure the terms in our bounds and show they tightly track actual performance gaps across settings.**

Assumptions. We make the following assumptions.

1. Log linear model holds: For each task $i \in [n]$, the performance can be expressed as $f_i(\mathrm{LM}(S,R,q)) = c_i + \exp(A_i^\top q)$ for some $c_i \in \mathbb{R}^+, A_i \in \mathbb{R}^{m'}$.
2. OLMIXBASE minimizes (2), and FULLMIXTUREREUSE minimizes (3).

Definitions. Define performance of q as $F(q) := \frac{1}{n}\sum_{i=1}^{n} f_i(\mathrm{LM}(S,R,q))$. $q^\star$ is the optimal solution to (2), and $q^\star(\tilde{p}_{\mathcal{D}_{\text{fix}}})$ is the solution to (3). We study the *performance gap* of FULLMIXTUREREUSE, $F(q^\star(\tilde{p}_{\mathcal{D}_{\text{fix}}})) - F(q^\star)$.

Let $q^\star_{\mathcal{D}_{\text{fix}}}$ be the optimal mix on $\mathcal{D}_{\text{fix}}$ after the update, so that $q^\star = [\rho^\star q^\star_{\mathcal{D}_{\text{fix}}}, (1-\rho^\star) q^\star_{\mathcal{D}_{\text{comp}}}]$. $\kappa(\mathcal{D}_{\text{fix}}, \mathcal{D}_{\text{comp}})$ is a coupling term (defined in Appendix E) that captures task influence of $\mathcal{D}_{\text{fix}}$ versus $\mathcal{D}_{\text{comp}}$.

4.3.1 PERFORMANCE CHARACTERIZATION

Our first result characterizes the performance gap in terms of $\tilde{p}_{\mathcal{D}_{\text{fix}}}$. See Appendix E.2 for the proof.

Theorem 4.1 (Performance gap bound). *There exists $C_1 > 0$ such that the performance gap satisfies*

$$F(q^\star(\tilde{p}_{\mathcal{D}_{fix}})) - F(q^\star) \le C_1 \kappa(\mathcal{D}_{fix}, \mathcal{D}_{comp}) \|\tilde{p}_{\mathcal{D}_{fix}} - q^\star_{\mathcal{D}_{fix}}\|$$

The performance gap is controlled by two factors:

- $\|\tilde{p}_{\mathcal{D}_{\text{fix}}} - q^\star_{\mathcal{D}_{\text{fix}}}\|$ (“reuse gap”): how close the mix we reuse is to the optimal mix after the update.
- $\kappa(\mathcal{D}_{\text{fix}}, \mathcal{D}_{\text{comp}})$: this coupling term is large when $\mathcal{D}_{\text{fix}}$ and $\mathcal{D}_{\text{comp}}$ impact the same downstream tasks. It controls the rate at which performance depends on the reuse gap.

When both terms are small, FULLMIXTUREREUSE matches full recomputation. Our empirical validation (Appendix F.2, Figure 15) confirms that the reuse gap correlates with the performance gap.

4.3.2 REUSE GAP

Theorem 4.1 shows the performance gap is governed by the reuse gap, but when is the reuse gap small? We analyze the case where $\tilde{p}$ is optimal before the update and domains are added. Proofs and results on other operators are in Appendix E.3.

Theorem 4.2. *Define $\tilde{p}$ as the solution to* (2) *on $\mathcal{D}$ and suppose that new domains are added. There exists a finite $C_2 > 0$ such that the reuse gap is bounded by*

$$\|\tilde{p}_{\mathcal{D}_{fix}} - q^\star_{\mathcal{D}_{fix}}\| \le C_2 \kappa(\mathcal{D}_{fix}, \mathcal{D}_{comp})(1-\rho^\star)$$

The reuse gap is controlled by two factors:

1. $\kappa(\mathcal{D}_{\text{fix}}, \mathcal{D}_{\text{comp}})$: the coupling term, also in Theorem 4.1.
2. $1-\rho^\star$: This term captures how much newly added domains move the optimum, since 1 is effectively ρ induced by $\tilde{p}$ before domains are added. This term can be small when: 1) the added domains are low utility, or 2) the added domains are small, which constrains their max weight in mixture optimization.

Our empirical validation (Appendix F.2, Figures 16, 18) shows that the success of FULLMIXTUREREUSE when adding domains correlates with small $1-\rho^\star$ and low coupling.

4.4 PARTIALMIXTUREREUSE

Our analysis (§4.3) characterizes when FULLMIXTUREREUSE maintains strong performance while reducing cost. PARTIALMIXTUREREUSE provides a flexible middle ground between FULLMIXTUREREUSE and full recomputation by strategically selecting which unaffected domains to reuse. This is especially effective when coupling between reused and recomputed domains is high.

Method Let $\mathcal{D}_{\text{partial}} \subset \mathcal{D}_1$ be a subset of unaffected domains we to reuse. We redefine the domains we reuse versus recompute as $\mathcal{D}_{\text{fix}} := \mathcal{D}_{\text{partial}}$ and $\mathcal{D}_{\text{comp}} := (\mathcal{D}_1 \setminus \mathcal{D}_{\text{partial}}) \cup \mathcal{D}'_2$, and then apply FULLMIXTUREREUSE (Algorithm 2) with this new partition. The mixing problem has dimension $1 + |\mathcal{D}'_2| + |\mathcal{D}_1 \setminus \mathcal{D}_{\text{partial}}|$, interpolating between FULLMIXTUREREUSE's $1 + |\mathcal{D}'_2|$ and full recomputation's m'. Since swarm cost scales linearly with dimension (RQ2), this directly interpolates computational cost.

Interpretation and Empirical Validation Carefully choosing $\mathcal{D}_{\text{partial}}$ can reduce the coupling $\kappa(\mathcal{D}_{\text{fix}}, \mathcal{D}_{\text{comp}})$ from Theorem 4.1. An intuitive example: when adding code data to web topics, the web source's software development topic should be recomputed alongside it since both strongly influence code evaluation tasks. Figures 17- 19 (Appendix F.2) confirms that this reduces coupling and improves performance compared to reusing all web topics.

5 Experimental Results

We evaluate mixture reuse across a sequence of five domain updates mirroring real-world LM development. Our results show that these methods maintain strong performance while substantially reducing computational costs compared to full recomputation. See Appendix G for full details.

5.1 Setup

Domain Updates We simulate a real-world LM development workflow that undergoes 5 updates (Table 1, Table 9). The final domain set contains 64 domains (token counts in Table 8).

Table 1: Simulated domain updates for §5 experiments.

Operation	**Dataset(s)**	Δm
Initial	WebOrganizer over Common Crawl	24
`Add`	New code dataset	+15
`Add`	Six PDF and math datasets	+6
`Revise`	Filter low quality PDF documents	0
`Remove`	Low quality math dataset	-1
`Partition`	WebOrganizer over PDF dataset	+20

Methods compared. We consider:

- Natural: mix proportional to domain sizes.
- Full recomputation: apply OlmixBase after each update (high performance, high cost).
- Swarm reuse: naively reuse all compatible proxy runs and apply OlmixBase (see Alg. 3).
- FullMixtureReuse (our method): reuse ratios for unaffected domains, recompute affected ones.
- PartialMixtureReuse (our method): selectively reuse domains to reduce coupling, such as between software development subset of the web and code data (see Appendix F.2).

Experimental protocol. For full recomputation, we use swarm size $K \approx c(m+1)$ and vary $c \in \{1,2,3\}$ to showcase different compute regimes. For swarm reuse and mixture reuse, we set swarm sizes to roughly match that of full recomputation with $c = 1$. We report results over 3 random seeds of swarms with $k = 4$, $R = 1T$, and in Appendix G.2 we use $R = 6$T (i.e., a more data-constrained setting).

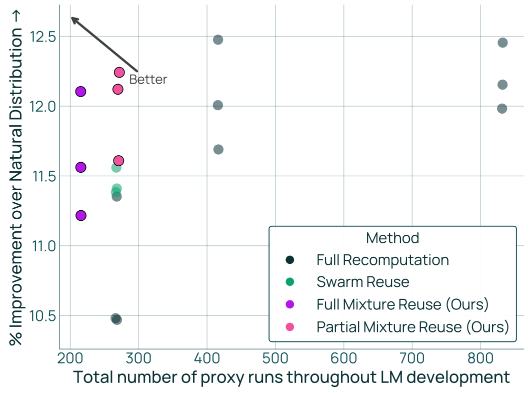

Figure 4: Performance improvement versus cost under evolving domains. Mixture reuse achieves $>95\%$ of the improvement of full recomputation while using $>69\%$ fewer proxy runs (§5.2).

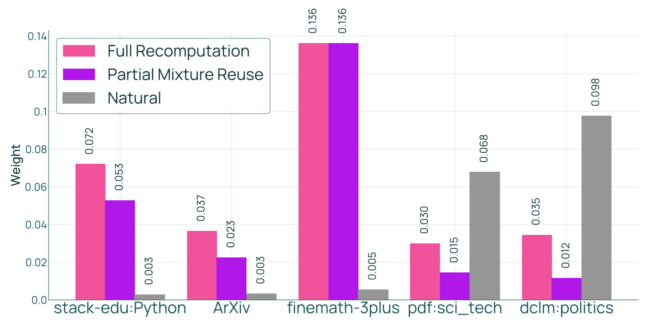

Figure 5: Proposed mixes for different mixing strategies. The mixes produced by full recomputation and PARTIALMIXTUREREUSE are similar to each other, confirming the performance results in Figure 4. Domains shown have the greatest difference in weights between full recomputation and natural.

5.2 RESULTS

FULLMIXTUREREUSE roughly matches full recomputation at substantially lower cost. Figure 4 shows relative improvement in downstream performance (average BPB) over the natural distribution versus the total number of proxy runs across all 5 updates. FULLMIXTUREREUSE achieves 95% of full recomputation's (c=3) performance improvement (+11.6% versus +12.2%) while using 74% fewer total proxy runs (216 versus 832).

PARTIALMIXTUREREUSE closes the remaining gap. By selectively recomputing some unaffected domains, PARTIALMIXTUREREUSE achieves 98% of full recomputation's performance (+12.0%) with 272 total runs—still 67% fewer than full recomputation.

Mixture reuse outperforms swarm reuse. Swarm reuse achieves +11.4% with 268 runs, underperforming FULLMIXTUREREUSE (+11.6% with 216 runs) despite using more runs. This is likely because (1) representing old swarms on updated domain sets over-explores biased subspaces, and (2) swarms cannot be reused when domains are removed or revised.

At matched proxy run budgets, mixture reuse outperforms alternatives. At a budget of 216-272 proxy runs, FULLMIXTUREREUSE and PARTIALMIXTUREREUSE achieve +11.6% and +12.0% improvement over the natural distribution, respectively. At this same budget, swarm reuse achieves +11.4% and full recomputation achieves +10.8% ($c = 1$). Mixture reuse is not only more efficient than full recomputation but also produces better mixes at the same cost.

Our best mixture is 3.05× more data-efficient than the natural distribution. Beyond final performance, we measure data efficiency: how many training steps does our best learned mixture need to match the natural distribution's final performance? Figure 21 in Appendix G.2 shows that PARTIALMIXTUREREUSE reaches the natural distribution's final BPB in approximately 20,000 steps versus 61,000 steps—a 3.05× speedup.

Qualitative analysis of learned mixtures. Figure 5 shows the final proposed mixes. Full recomputation and PARTIALMIXTUREREUSE produce similar allocations, while natural distribution differs substantially. This aligns with our findings on target model downstream performance.

6 DISCUSSION

We presented OLMIX, addressing two key challenges in data mixing for language models: configuring effective mixing methods and efficiently updating mixtures as domain sets evolve. We conduct a large empirical study of design choices and introduce mixture reuse. In future work, we aim to extend these ideas to online mixing and encourage co-design of mixing methods with other LM development workflows, emphasizing not effective mixing methods but how they are derived.

REFERENCES

Alon Albalak, Liangming Pan, Colin Raffel, and William Yang Wang. Efficient online data mixing for language model pre-training, 2023. URL https://arxiv.org/abs/2312.02406.

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

Loubna Ben Allal, Anton Lozhkov, and Elie Bakouch. Smollm - blazingly fast and remarkably powerful, 07 2024.

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, Joshua Lochner, Caleb Fahlgren, Xuan-Son Nguyen, Clémentine Fourrier, Ben Burtenshaw, Hugo Larcher, Haojun Zhao, Cyril Zakka, Mathieu Morlon, Colin Raffel, Leandro von Werra, and Thomas Wolf. Smollm2: When smol goes big – data-centric training of a small language model, 2025. URL https://arxiv.org/abs/2502.02737.

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

Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen McAleer, Albert Q. Jiang, Jia Deng, Stella Biderman, and Sean Welleck. Llemma: An open language model for mathematics, 2023.

Elie Bakouch, Loubna Ben Allal, Anton Lozhkov, Nouamane Tazi, Lewis Tunstall, Carlos Miguel Patiño, Edward Beeching, Aymeric Roucher, Aksel Joonas Reedi, Quentin Gallouédec, Kashif Rasul, Nathan Habib, Clémentine Fourrier, Hynek Kydlicek, Guilherme Penedo, Hugo Larcher, Mathieu Morlon, Vaibhav Srivastav, Joshua Lochner, Xuan-Son Nguyen, Colin Raffel, Leandro von Werra, and Thomas Wolf. SmolLM3: smol, multilingual, long-context reasoner. https://huggingface.co/blog/smollm3, 2025.

Lior Belenki, Alekh Agarwal, Tianze Shi, and Kristina Toutanova. Optimizing pre-training data mixtures with mixtures of data expert models, 2025. URL https://arxiv.org/abs/2502.15950.

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

Federico Cassano, John Gouwar, Daniel Nguyen, Sydney Nguyen, Luna Phipps-Costin, Donald Pinckney, Ming-Ho Yee, Yangtian Zi, Carolyn Jane Anderson, Molly Q Feldman, et al. Multipl-e: A scalable and extensible approach to benchmarking neural code generation. *arXiv preprint arXiv:2208.08227*, 2022.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. 2021.

Mayee F. Chen, Nicholas Roberts, Kush Bhatia, Jue Wang, Ce Zhang, Frederic Sala, and Christopher Ré. Skill-it! a data-driven skills framework for understanding and training language models, 2023. URL `https://arxiv.org/abs/2307.14430`.

Mayee F. Chen, Michael Y. Hu, Nicholas Lourie, Kyunghyun Cho, and Christopher Ré. Aioli: A unified optimization framework for language model data mixing, 2025a. URL `https://arxiv.org/abs/2411.05735`.

Shengzhuang Chen, Xu Ouyang, Michael Arthur Leopold Pearce, Thomas Hartvigsen, and Jonathan Richard Schwarz. Admire-bayesopt: Accelerated data mixture re-weighting for language models with bayesian optimization, 2025b. URL `https://arxiv.org/abs/2508.11551`.

Hyung Won Chung, Noah Constant, Xavier Garcia, Adam Roberts, Yi Tay, Sharan Narang, and Orhan Firat. Unimax: Fairer and more effective language sampling for large-scale multilingual pretraining, 2023. URL `https://arxiv.org/abs/2304.09151`.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? Try ARC, the AI2 reasoning challenge. *CoRR*, arXiv:1803.05457, 2018.

Shizhe Diao, Yu Yang, Yonggan Fu, Xin Dong, Dan Su, Markus Kliegl, Zijia Chen, Peter Belcak, Yoshi Suhara, Hongxu Yin, Mostofa Patwary, Yingyan, Lin, Jan Kautz, and Pavlo Molchanov. Climb: Clustering-based iterative data mixture bootstrapping for language model pre-training, 2025. URL `https://arxiv.org/abs/2504.13161`.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2368–2378, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1246. URL `https://aclanthology.org/N19-1246`.

Simin Fan, Matteo Pagliardini, and Martin Jaggi. Doge: Domain reweighting with generalization estimation, 2024. URL `https://arxiv.org/abs/2310.15393`.

Albert Ge, Tzu-Heng Huang, John Cooper, Avi Trost, Ziyi Chu, Satya Sai Srinath Namburi GNVV, Ziyang Cai, Kendall Park, Nicholas Roberts, and Frederic Sala. R&b: Domain regrouping and data mixture balancing for efficient foundation model training, 2025a. URL `https://arxiv.org/abs/2505.00358`.

Ce Ge, Zhijian Ma, Daoyuan Chen, Yaliang Li, and Bolin Ding. Bimix: A bivariate data mixing law for language model pretraining, 2025b. URL `https://arxiv.org/abs/2405.14908`.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathurx, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal

Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh

Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The llama 3 herd of models, 2024. URL `https://arxiv.org/abs/2407.21783`.

Dirk Groeneveld, Iz Beltagy, Pete Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, A. Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, Shane Arora, David Atkinson, Russell Authur, Khyathi Raghavi Chandu, Arman Cohan, Jennifer Dumas, Yanai Elazar, Yuling Gu, Jack Hessel, Tushar Khot, William Merrill, Jacob Daniel Morrison, Niklas Muennighoff, Aakanksha Naik, Crystal Nam, Matthew E. Peters, Valentina Pyatkin, Abhilasha Ravichander, Dustin Schwenk, Saurabh Shah, Will Smith, Emma Strubell, Nishant Subramani, Mitchell Wortsman, Pradeep Dasigi, Nathan Lambert, Kyle Richardson, Luke S. Zettlemoyer, Jesse Dodge, Kyle Lo, Luca Soldaini, Noah A. Smith, and Hanna Hajishirzi. Olmo: Accelerating the science of language models. *ArXiv*, abs/2402.00838, 2024. URL `https://api.semanticscholar.org/CorpusID:267365485`.

David Heineman, Valentin Hofmann, Ian Magnusson, Yuling Gu, Noah A. Smith, Hannaneh Hajishirzi, Kyle Lo, and Jesse Dodge. Signal and noise: A framework for reducing uncertainty in language model evaluation, 2025. URL `https://arxiv.org/abs/2508.13144`.

William Held, Bhargavi Paranjape, Punit Singh Koura, Mike Lewis, Frank Zhang, and Todor Mihaylov. Optimizing pretraining data mixtures with llm-estimated utility, 2025. URL `https://arxiv.org/abs/2501.11747`.

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

Yuzhen Huang, Jinghan Zhang, Zifei Shan, and Junxian He. Compression represents intelligence linearly. In *First Conference on Language Modeling*, 2024. URL `https://openreview.net/forum?id=SHMj84U5SH`.

Yiding Jiang, Allan Zhou, Zhili Feng, Sadhika Malladi, and J. Zico Kolter. Adaptive data optimization: Dynamic sample selection with scaling laws, 2024. URL `https://arxiv.org/abs/2410.11820`.

Feiyang Kang, Yifan Sun, Bingbing Wen, Si Chen, Dawn Song, Rafid Mahmood, and Ruoxi Jia. Autoscale: Scale-aware data mixing for pre-training llms, 2025. URL `https://arxiv.org/abs/2407.20177`.

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

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

Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Gadre, Hritik Bansal, Etash Guha, Sedrick Keh, Kushal Arora, Saurabh Garg, Rui Xin, Niklas Muennighoff, Reinhard Heckel, Jean Mercat, Mayee Chen, Suchin Gururangan, Mitchell Wortsman, Alon Albalak, Yonatan Bitton, Marianna Nezhurina, Amro Abbas, Cheng-Yu Hsieh, Dhruba Ghosh, Josh Gardner, Maciej Kilian, Hanlin Zhang, Rulin Shao, Sarah Pratt, Sunny Sanyal, Gabriel Ilharco, Giannis Daras, Kalyani Marathe, Aaron Gokaslan, Jieyu Zhang, Khyathi Chandu, Thao Nguyen, Igor Vasiljevic, Sham Kakade, Shuran Song, Sujay Sanghavi, Fartash Faghri, Sewoong Oh, Luke Zettlemoyer, Kyle Lo, Alaaeldin El-Nouby, Hadi Pouransari, Alexander Toshev, Stephanie Wang, Dirk Groeneveld, Luca Soldaini, Pang Wei Koh, Jenia Jitsev, Thomas Kollar, Alexandros G. Dimakis, Yair Carmon, Achal Dave, Ludwig Schmidt, and Vaishaal Shankar. Datacomp-lm: In search of the next generation of training sets for language models, 2024. URL https://arxiv.org/abs/2406.11794.

Qian Liu, Xiaosen Zheng, Niklas Muennighoff, Guangtao Zeng, Longxu Dou, Tianyu Pang, Jing Jiang, and Min Lin. Regmix: Data mixture as regression for language model pre-training, 2025a. URL https://arxiv.org/abs/2407.01492.

Yajiao Liu, Congliang Chen, Junchi Yang, and Ruoyu Sun. Rethinking data mixture for large language models: A comprehensive survey and new perspectives, 2025b. URL https://arxiv.org/abs/2505.21598.

Pratyush Maini, Skyler Seto, He Bai, David Grangier, Yizhe Zhang, and Navdeep Jaitly. Rephrasing the web: A recipe for compute and data-efficient language modeling, 2024. URL https://arxiv.org/abs/2401.16380.

MosaicML. Llm foundry - jeopardy dataset. https://github.com/mosaicml/llm-foundry/blob/main/scripts/eval/local_data/world_knowledge/jeopardy_all.jsonl, 2024. Accessed: 2024-11-10.

Niklas Muennighoff, Alexander M. Rush, Boaz Barak, Teven Le Scao, Aleksandra Piktus, Nouamane Tazi, Sampo Pyysalo, Thomas Wolf, and Colin Raffel. Scaling data-constrained language models, 2025. URL https://arxiv.org/abs/2305.16264.

Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, Nathan Lambert, Dustin Schwenk, Oyvind Tafjord, Taira Anderson, David Atkinson, Faeze Brahman, Christopher Clark, Pradeep Dasigi, Nouha Dziri, Michal Guerquin, Hamish Ivison, Pang Wei Koh, Jiacheng Liu, Saumya Malik, William Merrill, Lester James V. Miranda, Jacob Morrison, Tyler Murray, Crystal Nam, Valentina Pyatkin, Aman Rangapur, Michael Schmitz, Sam Skjonsberg, David Wadden, Christopher Wilhelm, Michael Wilson, Luke Zettlemoyer, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. 2 olmo 2 furious, 2024. URL https://arxiv.org/abs/2501.00656.

Team Olmo, :, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, Jacob Morrison, Jake Poznanski, Kyle Lo, Luca Soldaini, Matt Jordan, Mayee Chen, Michael Noukhovitch, Nathan Lambert, Pete Walsh, Pradeep Dasigi, Robert Berry, Saumya Malik, Saurabh Shah, Scott Geng, Shane Arora, Shashank Gupta, Taira Anderson, Teng Xiao, Tyler Murray, Tyler Romero, Victoria Graf, Akari Asai, Akshita Bhagia, Alexander Wettig, Alisa Liu, Aman Rangapur, Chloe Anastasiades, Costa Huang, Dustin Schwenk, Harsh Trivedi, Ian Magnusson, Jaron Lochner, Jiacheng Liu, Lester James V. Miranda, Maarten Sap, Malia Morgan, Michael Schmitz, Michal Guerquin, Michael Wilson, Regan Huff, Ronan Le Bras, Rui Xin, Rulin Shao, Sam Skjonsberg, Shannon Zejiang Shen, Shuyue Stella Li, Tucker Wilde, Valentina Pyatkin, Will Merrill, Yapei Chang, Yuling Gu, Zhiyuan Zeng, Ashish Sabharwal, Luke Zettlemoyer, Pang Wei Koh, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. Olmo 3, 2025. URL https://arxiv.org/abs/2512.13961.

Ankit Pal, Logesh Kumar Umapathi, and Malaikannan Sankarasubbu. Medmcqa: A large-scale multi-subject multi-choice dataset for medical domain question answering. In Gerardo Flores, George H Chen, Tom Pollard, Joyce C Ho, and Tristan Naumann (eds.), *Proceedings of the Conference on Health, Inference, and Learning*, volume 174 of *Proceedings of Machine Learning Research*, pp. 248–260. PMLR, 07–08 Apr 2022. URL https://proceedings.mlr.press/v174/pal22a.html.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The lambada dataset: Word prediction requiring a broad discourse context. *arXiv preprint arXiv:1606.06031*, 2016.

Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, Thomas Wolf, et al. The FineWeb Datasets: Decanting the Web for the Finest Text Data at Scale. In *The Thirty-eight Conference on Neural Information Processing Systems; Datasets and Benchmarks Track*, 2024.

Haoran Que, Jiaheng Liu, Ge Zhang, Chenchen Zhang, Xingwei Qu, Yinghao Ma, Feiyu Duan, Zhiqi Bai, Jiakai Wang, Yuanxing Zhang, Xu Tan, Jie Fu, Wenbo Su, Jiamang Wang, Lin Qu, and Bo Zheng. D-cpt law: Domain-specific continual pre-training scaling law for large language models, 2024. URL `https://arxiv.org/abs/2406.01375`.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2024. URL `https://arxiv.org/abs/2412.15115`.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In Jian Su, Kevin Duh, and Xavier Carreras (eds.), *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2383–2392, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1264. URL `https://aclanthology.org/D16-1264`.

Siva Reddy, Danqi Chen, and Christopher D. Manning. CoQA: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266, 2019. doi: 10.1162/tacl_a_00266. URL `https://aclanthology.org/Q19-1016`.

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

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. Social IQa: Commonsense reasoning about social interactions. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 4463–4473, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1454. URL `https://aclanthology.org/D19-1454`.

Luca Soldaini and Kyle Lo. peS2o (Pretraining Efficiently on S2ORC) Dataset, 2023. URL `https://github.com/allenai/pes2o`.

Luca Soldaini, Rodney Kinney, Akshita Bhagia, Dustin Schwenk, David Atkinson, Russell Authur, Ben Bogin, Khyathi Chandu, Jennifer Dumas, Yanai Elazar, Valentin Hofmann, Ananya Harsh Jha, Sachin Kumar, Li Lucy, Xinxi Lyu, Nathan Lambert, Ian Magnusson, Jacob Morrison, Niklas Muennighoff, Aakanksha Naik, Crystal Nam, Matthew E. Peters, Abhilasha Ravichander, Kyle Richardson, Zejiang Shen, Emma Strubell, Nishant Subramani, Oyvind Tafjord, Pete Walsh, Luke Zettlemoyer, Noah A. Smith, Hannaneh Hajishirzi, Iz Beltagy, Dirk Groeneveld, Jesse Dodge, and Kyle Lo. Dolma: an open corpus of three trillion tokens for language model pretraining research, 2024.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4149–4158, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1421. URL `https://aclanthology.org/N19-1421`.

Kimi Team, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, Zhuofu Chen, Jialei Cui, Hao Ding, Mengnan Dong, Angang Du, Chenzhuang Du, Dikang Du, Yulun Du, Yu Fan, Yichen Feng, Kelin Fu, Bofei Gao, Hongcheng Gao, Peizhong Gao, Tong Gao, Xinran Gu, Longyu Guan, Haiqing Guo, Jianhang Guo, Hao Hu, Xiaoru Hao, Tianhong He, Weiran He, Wenyang He, Chao Hong, Yangyang Hu, Zhenxing Hu, Weixiao Huang, Zhiqi Huang, Zihao Huang, Tao Jiang, Zhejun Jiang, Xinyi Jin, Yongsheng Kang, Guokun Lai, Cheng Li, Fang Li, Haoyang Li, Ming Li, Wentao Li, Yanhao Li, Yiwei Li, Zhaowei Li, Zheming Li, Hongzhan Lin, Xiaohan Lin, Zongyu Lin, Chengyin Liu, Chenyu Liu, Hongzhang Liu, Jingyuan Liu, Junqi Liu, Liang Liu, Shaowei Liu, T. Y. Liu, Tianwei Liu, Weizhou Liu, Yangyang Liu, Yibo Liu, Yiping Liu, Yue Liu, Zhengying Liu, Enzhe Lu, Lijun Lu, Shengling Ma, Xinyu Ma, Yingwei Ma, Shaoguang Mao, Jie Mei, Xin Men, Yibo Miao, Siyuan Pan, Yebo Peng, Ruoyu Qin, Bowen Qu, Zeyu Shang, Lidong Shi, Shengyuan Shi, Feifan Song, Jianlin Su, Zhengyuan Su, Xinjie Sun, Flood Sung, Heyi Tang, Jiawen Tao, Qifeng Teng, Chensi Wang, Dinglu Wang, Feng Wang, Haiming Wang, Jianzhou Wang, Jiaxing Wang, Jinhong Wang, Shengjie Wang, Shuyi Wang, Yao Wang, Yejie Wang, Yiqin Wang, Yuxin Wang, Yuzhi Wang, Zhaoji Wang, Zhengtao Wang, Zhexu Wang, Chu Wei, Qianqian Wei, Wenhao Wu, Xingzhe Wu, Yuxin Wu, Chenjun Xiao, Xiaotong Xie, Weimin Xiong, Boyu Xu, Jing Xu, Jinjing Xu, L. H. Xu, Lin Xu, Suting Xu, Weixin Xu, Xinran Xu, Yangchuan Xu, Ziyao Xu, Junjie Yan, Yuzi Yan, Xiaofei Yang, Ying Yang, Zhen Yang, Zhilin Yang, Zonghan Yang, Haotian Yao, Xingcheng Yao, Wenjie Ye, Zhuorui Ye, Bohong Yin, Longhui Yu, Enming Yuan, Hongbang Yuan, Mengjie Yuan, Haobing Zhan, Dehao Zhang, Hao Zhang, Wanlu Zhang, Xiaobin Zhang, Yangkun Zhang, Yizhi Zhang, Yongting Zhang, Yu Zhang, Yutao Zhang, Yutong Zhang, Zheng Zhang, Haotian Zhao, Yikai Zhao, Huabin Zheng, Shaojie Zheng, Jianren Zhou, Xinyu Zhou, Zaida Zhou, Zhen Zhu, Weiyu Zhuang, and Xinxing Zu. Kimi k2: Open agentic intelligence, 2025. URL `https://arxiv.org/abs/2507.20534`.

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

Alexander Wettig, Kyle Lo, Sewon Min, Hannaneh Hajishirzi, Danqi Chen, and Luca Soldaini. Organize the web: Constructing domains enhances pre-training data curation, 2025. URL `https://arxiv.org/abs/2502.10341`.

Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifeng Lu, Percy Liang, Quoc V. Le, Tengyu Ma, and Adams Wei Yu. Doremi: Optimizing data mixtures speeds up language model pretraining, 2023. URL `https://arxiv.org/abs/2305.10429`.

Wanyun Xie, Francesco Tonin, and Volkan Cevher. Chameleon: A flexible data-mixing framework for language model pretraining and finetuning, 2025. URL `https://arxiv.org/abs/2505.24844`.

Jiasheng Ye, Peiju Liu, Tianxiang Sun, Jun Zhan, Yunhua Zhou, and Xipeng Qiu. Data mixing laws: Optimizing data mixtures by predicting language modeling performance, 2025. URL `https://arxiv.org/abs/2403.16952`.

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

A Extended Related Work

Mixing Methods. The offline mixing schema, also described as Function Fitting-based Methods in Liu et al. (2025b) has been extensively used in existing mixing methods. In Table 2, we provide an overview of several offline mixing methods and how they address the key design choices we study; however, there is significant variation. Some methods use explicit parametric regression models (Que et al., 2024), while others use nonparametric approaches like LightGBM and Gaussian Processes (Liu et al., 2025a; Chen et al., 2025b). Some methods explicitly model the role of proxy model size or training budget (Ge et al., 2025b; Ye et al., 2025; Kang et al., 2025) while others assume direct generalization from proxy models to target models (Held et al., 2025). There are also several methods that build on top of RegMix, exploring how it can be augmented with better domains (Wettig et al., 2025), iterative swarms (Diao et al., 2025), or more features (Belenki et al., 2025). Our work examines key design choices for the offline schema that underlies all these methods.

Aside from the offline schema, DoReMi (Xie et al., 2023) and DoGE (Fan et al., 2024) produce a set of mixture weights by using one or two proxy runs that dynamically explore the mixture weight space; this is in contrast to using many proxy runs with static mixes. While these approaches have proven effective in some settings, previous analysis (Chen et al., 2025a) has suggested some suboptimality in how the dynamic update rules are constructed, and the offline schema is generally considered simpler to implement.

Online mixing methods (Chen et al., 2023; Jiang et al., 2024; Albalak et al., 2023) adjust the mixture weights throughout the final training run. Rather than exploring the mixture space and learning from it offline, these methods implicitly do this on the fly during training. Note that our notion of dynamics, which is over the domain set and during LM development (i.e., before the final training run), is different from the dynamic aspect of online mixing methods.

Existing mixture literature largely assumes fixed domain sets. The recent work Chameleon (Xie et al., 2025) addresses adaptability to domain changes by computing domain weights from learned embeddings using kernel ridge leverage scores, which allows direct transfer to new data without proxy retraining. However, Chameleon focuses on adding new domains and computes weights directly from domain embeddings without the swarm-based regression approach of offline methods. In contrast, our mixture reuse approach explicitly handles various domain update operations (additions, removals, revisions, and partitions) and is designed to work within the offline mixing schema. Moreover, our approach provides theoretical and empirical analysis of when and why existing mixes can be reused.

Data-constrained settings. Several works have addressed training language models under data constraints. UniMax (Chung et al., 2023) proposes an allocation algorithm for multilingual pretraining that distributes a fixed token budget to maximize uniform coverage across languages while capping repetitions to avoid overfitting on low-resource languages. However, their allocation procedure is not integrated with data mixing methods and focuses specifically on the multilingual setting. Muennighoff et al. (2025) empirically study scaling laws when data is limited and must be repeated, finding that up to 4 epochs of repetition yields negligible degradation and proposing modified scaling laws that account for diminishing returns of repeated tokens. While these works address data constraints through allocation procedures and scaling laws, our work integrates repetition constraints directly into data mixing, enabling principled allocation of limited data budgets while producing a mix that yields strong downstream performance.

LM Data Development. Real-world LM development involves iterative refinement of training data. Works like DCLM (Li et al., 2024), Dolma (Soldaini et al., 2024), and FineWeb (Penedo et al., 2024) extensively document the curation processes involved in creating high-quality pretraining corpora, including quality filtering, deduplication strategies, and careful domain selection. Continuous projects like OLMo 1-3 (Groeneveld et al., 2024; OLMo et al., 2024; Olmo et al., 2025) and SmolLM 1-3 (Allal et al., 2024; 2025; Bakouch et al., 2025) showcase how training data evolves over time through these updates. For instance, the OLMo series documents changes in domain composition, filtering strategies, and data sources across model versions, while SmolLM similarly shows iterative improvements to the training mixture. These projects illustrate that domain sets are not static but undergo frequent revisions including dataset additions (e.g., new web crawls), removals (e.g., deprecated sources), quality improvements (e.g., enhanced filtering), and domain partitioning (e.g., splitting broad categories into specialized subdomains). Motivated by these works, OLMIX views data mixing as an ongoing process throughout LM development, providing both systematic methods for initial mixture configuration and efficient strategies for maintaining strong mixtures as domain sets evolve.

Table 2: Design choices across offline mixing methods. For each design choice and mixing method, we identify the method's configuration. We shade cells in pink if we found empirical justification for their configuration. In the final column, we present the configuration for OLMIXBASE.

Design Choice	**RegMix** (Liu et al., 2025a)	**DML** (Ye et al., 2025)	**AutoScale** (Kang et al., 2025)	**BiMix** (Ge et al., 2025b)	**ADMIRE-BayesOpt** (Chen et al., 2025b)	**CLIMB** (Diao et al., 2025)	**OLMIXBASE (Algorithm 1)**
Swarm Construction							
Proxy model size	1M	70, 160, 305, 410M	Target	280M	1M, 60M	350M	30M
Swarm size (vs m domains)	512 ($m=17$)	20 ($m=7$)	$2m+1$	4	101 ($m=17$)	112 ($m=21$)	$3(m+1)$
Swarm distribution	Dirichlet with natural prior	Exponential grid	Exponential grid	Entropy-weighted	Dynamic	Dirichlet with natural prior	Dirichlet with natural prior (sparse for topics, dense for sources)
Regression Model							
Regression model family	LightGBM	Log-Linear	Power Law	Power Law	Gaussian Process	LightGBM	Log-Linear
Regression granularity	Aggregated	Aggregated	Per-Task	Per-Task	Aggregated	Aggregated	Per-Task
Mixture Optimization							
Data repetition constraints	No	No	No	No	No	No	Yes
Optimization solver	Search	Search	Gradient Descent	Exact Solver	Search	Search	Exact Solver with KL reg.

B ADDITIONAL DESIGN CHOICES

In addition to RQ1-4 in §3.2, we identify several other design choices in the offline mixing schema. We empirically study each design choice and report our findings.

We conduct an empirical study where we explore each OLMIX design choice and present our procedure on how to decide on each of them. Appendix C provides the exact experimental setup for each research question.

B.1 RQ6: SWARM DISTRIBUTION

How should we specify the distribution $\mathcal{P}$ to sample the mixes for the swarm runs?

We study the distribution $\mathcal{P}$ from which swarm mixtures should be sampled. The distribution determines how effectively the swarm explores the mixture space. However, existing works rarely study the impact of the swarm distribution, with only CLIMB (Diao et al., 2025) comparing a Dirichlet versus a random uniform distribution.

We investigate two aspects of the swarm distribution. To evaluate each aspect, we measure the average BPB of 1B target models trained on proposed mixes. As an intermediate metric, we also report the regression fit (Pearson correlation between predicted and true per-task BPB) on a held-out set of mixtures.

- *Should each proxy run include samples from all domains (dense), or should some runs focus on subsets of domains (sparse)?* Sparse distributions enable discovering that certain domains should be excluded, while dense distributions ensure all domains are represented in all proxy runs. We generate sparse and dense swarms at the **topic level** (on DCLM topics) and **source level** to capture both fine and coarse-grained domains.
- *Does centering the swarm around a promising mixture region improve results?* We test three Dirichlet priors: 1) the natural distribution based on domain token counts, 2) a strong prior, and 3) a weak prior (see Appendix C for exact constructions).

Figure 6 shows that sparse swarms outperform dense at the topic level and vice versa at the source level—both for downstream BPB (left) and regression fit (right). Neither is universally better; the choice depends on the domain set. One hypothesis is that the best topic-level mixes exclude certain low-signal topics (e.g., adult content in DCLM), while the best source-level mixes utilize all sources. This aligns with how these domains were constructed: DCLM was partitioned into topics that are potentially uninformative, while sources are intentionally curated. **We find that the choice of swarm**

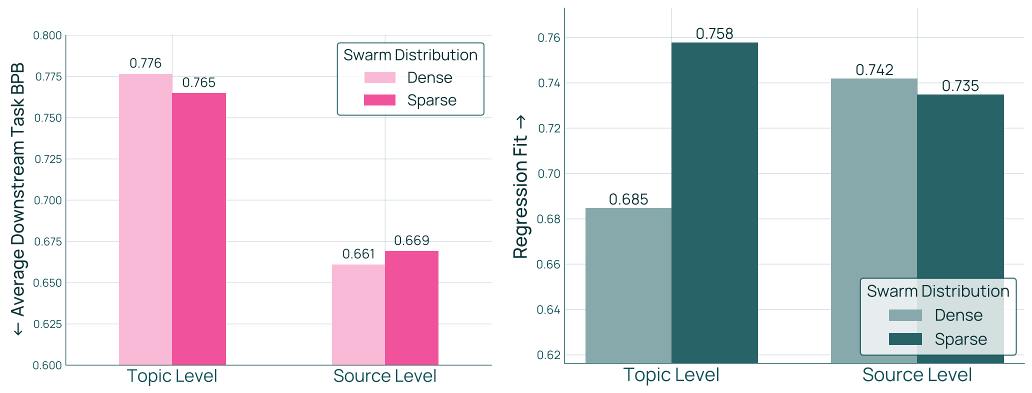

Figure 6: Performance (left) and regression fit (right) of dense versus sparse swarm distributions. While sparse swarms perform better at the topic level, dense swarms perform better at the source level, suggesting that this design choice is data-dependent.

Table 3: Natural and strong priors achieve roughly comparable downstream performance, but the weak prior does considerably worse, suggesting that this design choice is fairly robust and that the natural distribution is a reasonable configuration.

Dirichlet Prior	Avg Downstream Task BPB ↓	Regression Fit ↑
Natural	0.765	0.748
Strong	0.763	0.835
Weak	0.797	0.661

distribution depends on the domain set; we recommend using sparse distributions for topic-level mixing and dense distributions for source-level mixing.

Table 3 shows that centering on the natural distribution and strong prior results in roughly similar downstream performance, but using the weak prior significantly degrades performance. **We recommend that practitioners center the swarm around strong priors when available, but if prior knowledge is uncertain, the natural distribution remains a reasonable fallback.**

B.2 RQ7: REGRESSION GRANULARITY

At what granularity should we fit the regression models in order to construct $\hat{f}(p)$?

We study the granularity at which to fit regression models: one model per task, one model per task family (e.g., math, code, and QA), or one model for the entire average BPB. This involves a tradeoff between expressivity and noise. Per-task and per-family regression models can capture distinct relationships between mixtures and various capabilities, increasing expressivity of the overall function approximating average BPB, $\hat{f}(p)$. However, they are fit to noisier individual measurements, while modeling only average BPB may involve less noisy targets.

We compare three approaches. In the **per-task** approach, we fit individual functions $\hat{f}_i(p)$ using $\{(p^j, y_{ij})\}_{j=1}^K$ for each task $i \in [n]$, and then minimize $\frac{1}{n}\sum_{i=1}^n \hat{f}_i(p)$. In the **per-family** approach, we group tasks into three families F_1, F_2, F_3: math, code, and QA. We fit one model $\bar{f}_i(p)$ to each family's average BPB using $\{(p^j, \frac{1}{|F_i|}\sum_{k\in F_i} y_{kj})\}_{j=1}^K$ for $i = 1,2,3$, and then minimize $\sum_{i=1}^3 \frac{|F_i|}{n}\bar{f}_i(p)$. In the **aggregated** approach, we fit a single function $\hat{f}_{\text{avg}}(p)$ directly to the average BPB across tasks using $\{(p^j, \frac{1}{n}\sum_{i=1}^n y_{ij})\}_{j=1}^K$ and minimize $\hat{f}_{\text{avg}}(p)$. We evaluate downstream performance (average BPB of 1B target models) and regression fit (Pearson correlation on held-out mixtures).

Table 4 shows that per-task yields the best downstream performance and regression fit. Regression fit degrades monotonically as granularity decreases, consistent with reduced expressivity. While per-family has better regression fit than aggregated, their downstream performance is comparable; this

Table 4: Performance and regression fit across regression granularities. As granularity of the regression target increases, the regression fit improves, and the downstream performance also trends towards improvement.

Method	Avg Downstream Task BPB ↓	Regression Fit ↑
Per-Task	0.765	0.983
Per-Family	0.777	0.958
Aggregated	0.774	0.866

is likely due to noise in transferring from proxy to target models, since the performance on 30M models (Figure 13) exhibits trends that are consistent with the regression fit. Nevertheless, per-task provides clear benefits on both metrics. **We recommend fitting separate regression functions per task.**

B.3 RQ8: Optimization solver

How do we solve the mixture optimization problem?

We study how to solve for the mixture that minimizes the average predicted BPB $\hat{f}(p)$. Under log-linear regression, $\hat{f}(p)$ is convex, enabling exact solvers. However, since regression functions imperfectly predict true performance (Figure 9 left), it is unclear whether exact optimization of the surrogate objective yields the mix with the best downstream performance or if incorporating some regularization may be better.

We consider three solving strategies. The **exact solver** uses CVXPY to compute the global optimal mix. The **search**-based solver, used in RegMix (Liu et al., 2025a), samples candidate mixes from a Dirichlet distribution with a natural prior (i.e., a mix proportional to the domain sizes). The **exact solver with KL regularization** modifies the objective to minimize$_{p\in\triangle^{m-1}}\hat{f}(p)+\lambda D_{\mathrm{KL}}(p||p_0)$, where $\lambda>0$ controls the strength of regularization towards the natural distribution p_0; we consider $\lambda\in\{0.01,0.05\}$. We measure the average BPB of 1B target models trained on the proposed mixes. We also examine the optimal objective value, the predicted average BPB at the proxy model scale, as a sanity check that the exact solver should obtain the lowest predicted BPB.

Figure 7 (left) shows that the exact solver with $\lambda=0.05$ achieves the best performance. In contrast, the right panel confirms that the exact solver obtains the lowest predicted BPB, while adding a KL penalty degrades it. Taken together, these results indicate that although the average predicted BPB is optimized most effectively by the exact solver, noise from regression fitting and proxy-to-target model transfer makes moderate regularization beneficial. The search baseline achieves poor performance, consistent with it being a heuristic. **We recommend using an exact solver with a KL regularization of 0.05 to solve the mixture optimization problem**. See Appendix C.7, Figure 14 for additional source-level results.

C Offline Schema Study Details

All experiments for the design choice study, unless specified, use the same configuration as OlmixBase. For mixing over the DCLM topics, we constructed a sparse swarm of size $K=128$. We also mix at the source level over DCLM, Stack-Edu, ArXiv, FineMath 3+, olmOCR Science PDFs, Wikipedia, and Pes2o (see Table 8). For the sources, we constructed a dense swarm of size $K=64$. We use log-linear regression per task, an exact solver with $\lambda=0.05$, and enforced repetition constraints in the optimization problem with $R=$ 6T and $k=4$.

C.1 RQ1: Proxy model size details

Details Table 5 contains details about the architectures of the 1M, 15M, 30M, and 60M proxy models we train.

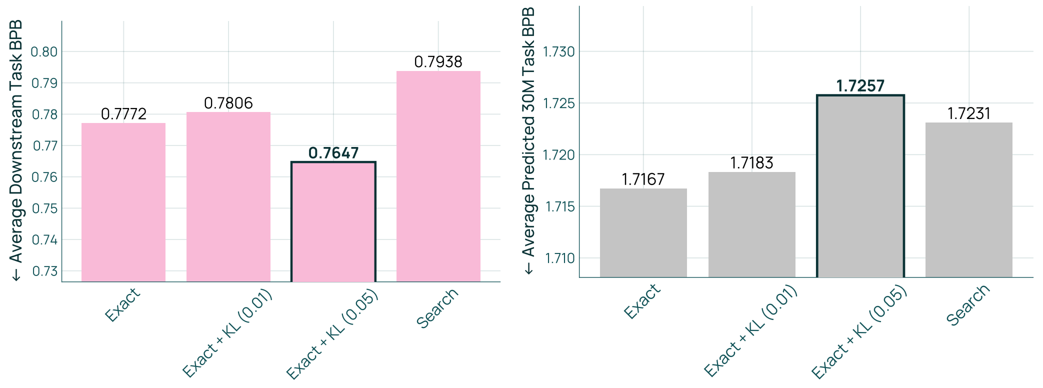

Figure 7: Downstream performance (left) and 30M predicted performance (right) across optimization solvers. The predicted performances of various solvers confirms that the exact solver minimizes the predicted BPB, as expected, followed by KL penalties of 0.01 and 0.05. However, the exact solver does not obtain the best downstream performance, and instead some KL regularization helps.

Table 5: Proxy Model Architecture Configurations.

Parameter	**1M**	**15M**	**30M**	**60M**
Vocab size	100,352	100,352	100,352	100,352
`n_layers`	4	8	4	8
`n_heads`	4	4	8	8
`d_model`	16	128	256	384
`head_dim`	4	32	32	48

C.2 RQ2: Swarm size details

Additional results In Figure 8, we vary $K = c(m+1)$ for $c = 1,2,3,4$ and $m = 6,24$ and study its effect on downstream BPB of 1B models. Despite having fewer m, the error curves when scaled according to c collapse together, suggesting that the $\mathcal{O}(m)$ sample complexity holds at both 30M and 1B model scales.

Details For the results in Figure 2 and Figure 8, we constructed proposed mixes using a search-based solver rather than an exact solver. We also did not enforce repetition constraints and had originally included 5 additional metrics in our evaluation suite: Qasper Yes/No, Sciriff Yes/No, LabBench DBQA, LabBench ProtocolQA, and MedQA EN.

For $m = 24$, we used all WebOrganizer DCLM topics. For $m < 24$, we selected the m largest domains. In particular, for $m = 18$, we used: Art and Design, Crime and Law, Education and Jobs, Electronics and Hardware, Entertainment, Finance and Business, Games, Health, Literature, Politics, Religion, Science Math and Technology, Social Life, Software, Software Development, Sports and Fitness, Transportation, Travel and Tourism. For $m = 12$, we used: Crime and Law, Education and Jobs, Entertainment, Finance and Business, Games, Health, Literature, Politics, Religion, Science Math and Technology, Software, and Software Development. For $m = 6$, we used: Entertainment, Finance and Business, Games, Health, Politics, and Science Math and Technology.

C.3 RQ6: Swarm distribution details

Details To construct the sparse and dense swarms, we first generated mixes sampled from a Dirichlet distribution. For the dense swarm, we discarded any mixes that have domains with 0 weight, and for the sparse swarm, we enforced that if the weight of a domain is less than 0.05, then it is clipped to 0 and the entire mixture is normalized afterwards.

For the centering experiments, we compared a swarm with a natural Dirichlet prior to a strong and weak prior. The strong prior was the proposed mix obtained using the natural swarm. The weak prior was a mix that aimed to maximize BPB using the the natural swarm using a simulation-based solver.

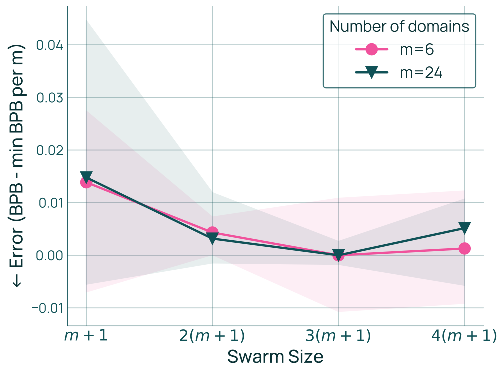

Figure 8: Error on 1B models versus swarm size. Similar to Figure 2 (right), we see that the error curves collapse across both m, supporting our finding that K needs $\mathcal{O}(m)$ runs to obtain strong downstream performance. Results are averaged across 3 random seeds, and the intervals indicate min and max results.

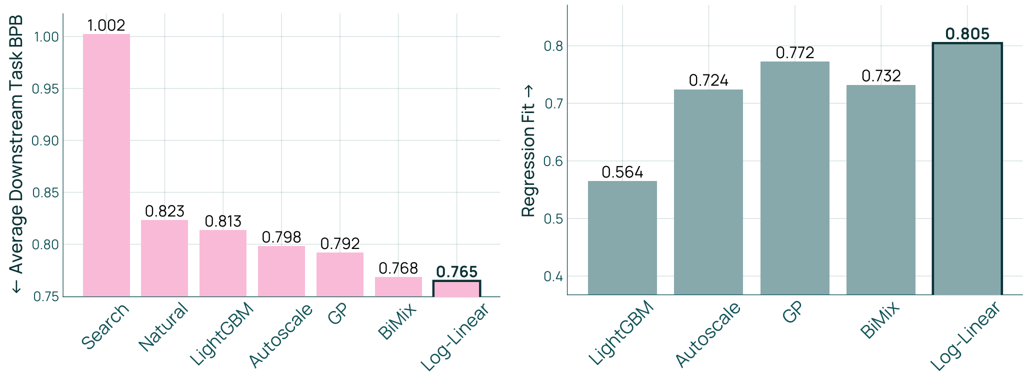

Figure 9: Performance (left) and regression fit (right) across regression model families when mixing on DCLM topics. For both metrics, the log-linear regression model outperforms other approaches.

Regardless of the choice of Dirichlet prior, we used the natural distribution in the KL regularization term of the objective for fair comparison.

For both sets of experiments, we constructed a held-out test set containing of 30 samples, where 15 were from a Dirichlet distribution centered around the sparse swarm's optimal mix and 15 were from a Dirichlet distribution centered around the dense swarm's optimal mix. This measures regression fit in high-performance regions of the mixture space, which matters in the mixture optimization step.

C.4 RQ3: Regression model family details

Additional results Figures 9 and 10 compare the downstream performance and regression fit (measured on three random train-test splits of the swarm, each with 10 test mixes) across several regression model families for two settings: DCLM topics, and at the source level. In both settings and across both metrics, we see that the log-linear regression model outperforms other approaches.

In Figures 11 and 12, we plot the relationship between swarm size, number of domains, and regression fit across all regression model families. These figures reveal that different regression model families have different sample complexity and regimes in which they do well. This offers a potential explanation for why existing methods lack consensus.

Details We describe how we adapt the regression models from BiMix (Ge et al., 2025b), Autoscale (Kang et al., 2025), and Data Mixing Laws (Ye et al., 2025) to our setting where performance is measured on a set of downstream tasks.

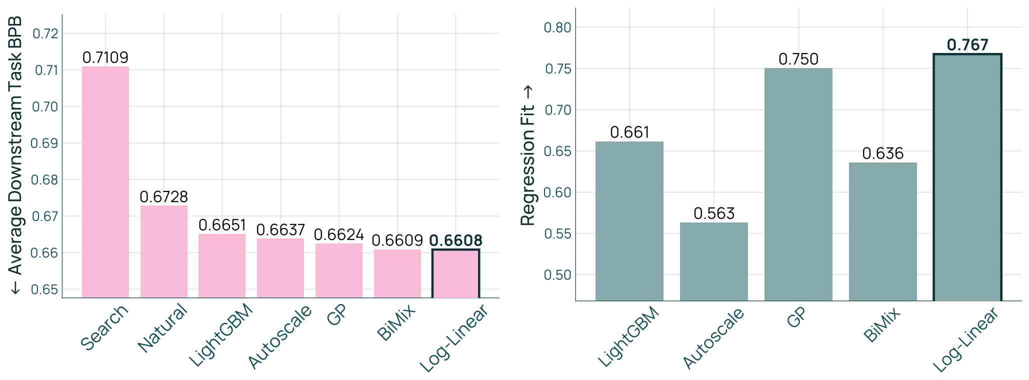

Figure 10: Performance (left) and regression fit (right) across regression model families when mixing across sources. For both metrics, the log-linear regression model outperforms other approaches again.

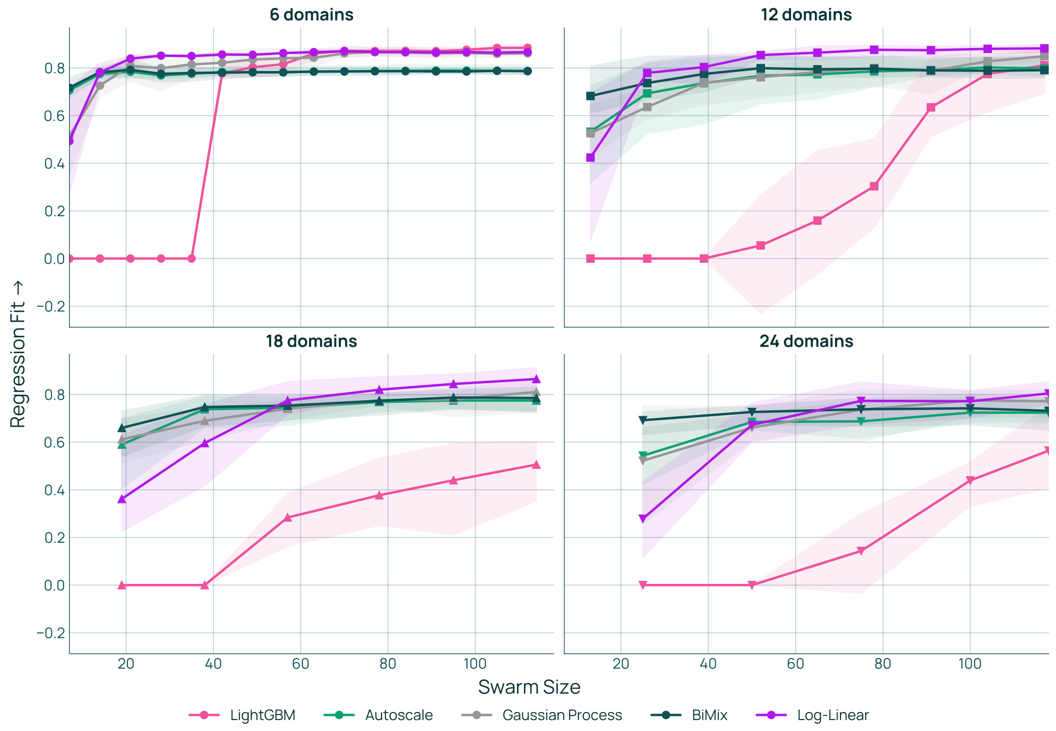

Figure 11: Regression fit versus swarm size and number of domains across regression models when mixing on DCLM topics. Across $m=6,12,18,24$ domains, each regression model requires different swarm sizes to obtain good regression fit. Results are averaged over 3 random seeds.

- BiMix: The BiMix mixing law models the validation loss on domain i as a function of the mixture ratio on domain i and the number of training steps s:

$$\hat{f}_i(p,s) = \frac{A_i}{p_i^{\alpha_i}} \left(\frac{B_i}{s^{\beta_i}} + C_i \right) \forall i \in [m] \tag{5}$$

  Since we do not model the number of steps and instead directly transfer the proposed mix from the proxy scale to the target scale, this mixing law simplifies to $\hat{f}_i(p) = A_i p_i^{-\alpha_i}$, where the constants are absorbed into A_i. To extend this mixing law to downstream tasks rather than validation loss on training domains, we assume that task BPB can be modeled as a weighted average of training domain-

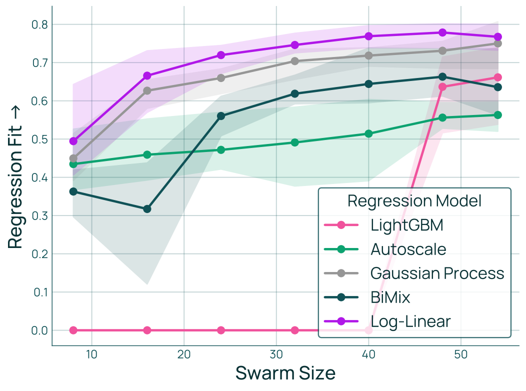

Figure 12: Regression fit versus swarm size and number of domains across regression models when mixing across sources. Across $m=6,12,18,24$ domains, each regression model requires different swarm sizes to obtain good regression fit. Results are averaged over 3 random seeds.

specific mixing laws. Therefore, for each downstream task i, the BiMix mixing law for our setting is:

$$\hat{f}_i(p) = \sum_{j=1}^{m} A_{ij} p_j^{-\alpha_{ij}} \tag{6}$$

- AutoScale: The AutoScale mixing law models validation loss on a held-out dataset i as a function of the mixture and the number of tokens R:

$$\hat{f}(p,R) = \sum_{j=1}^{m} \left((N_0^j + p_j R)^{-\gamma_j} + l_j \right), \tag{7}$$

where N_0^j is the "effective" number of tokens of all other domains excluding j. The original paper fits separate parameters for each domain using perturbed runs; we instead fit all parameters jointly using our larger swarm. We make two modifications. First, we replace the per-domain constants l_j with a single constant term c_i corresponding to a given task i. Second, instead of directly learning N_0^j, which can be quite large given the order of magnitude of R, we rewrite $N_0^j + p_j R$ as $R(A_{ij} + p_j)$ and learn A_{ij}. Therefore, for each downstream task i, the AutoScale mixing law for our setting is:

$$\hat{f}_i(p) = c_i + \sum_{j=1}^{m} (R(A_{ij} + p_j))^{-\alpha_{ij}} \tag{8}$$

- Data Mixing Laws: The mixing law models the validation loss on domain i as:

$$\hat{f}_i(p) = c_i + k_i \exp\left(\sum_{j=1}^{m} t_{ij} p_j \right) \tag{9}$$

The paper also extends this to out-of-domain validation sets by expressing them as a linear combination of training domains, namely $\hat{f}(p) = \sum_{i=1}^{m} s_i \left(c_i + k_i \exp\left(\sum_{j=1}^{m} t_{ij} p_j \right) \right)$. We use the first, simpler form (9) directly for downstream tasks, treating each task's performance as a function of the mixture on training domains. Moreover, we observe that k_i can be absorbed into the linear term; rewriting (9) as $c_i + \exp(\sum_{j=1}^{m} t_{ij} p_j + \log k_i)$ shows that k_i acts as an intercept term. Since p sums up to 1, this intercept is redundant and can be dropped. Therefore, our final log-linear mixing law is

$$\hat{f}_i(p) = c_i + \exp\left(\sum_{j=1}^{m} A_{ij} p_j \right) \tag{10}$$

Next, we provide implementation details for each model:

Table 6: Downstream performance of BiMix and AutoScale regression models on DCLM topics using exact optimization, exact + KL regularization ($\lambda = 0.05$), and search. Based on these findings, we use an exact solver with Autoscale and a search-based solver for BiMix for the rest of our study.

Method	Exact (0.0)	Exact + KL (0.05)	Search
BiMix	0.776733	0.790188	**0.768153**
Autoscale	**0.798084**	0.804814	0.808775

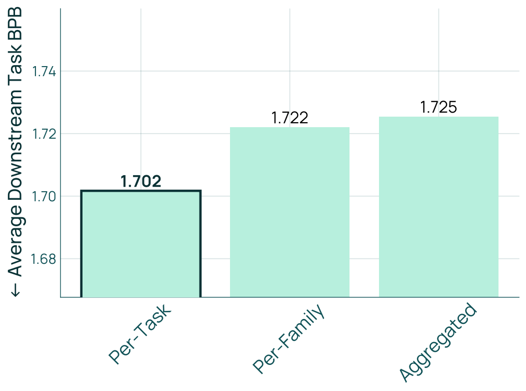

Figure 13: 30M performance across regression granularities. The performance is monotonic in the task granularity, matching with the regression fit findings from Table 4.

- Search: we select the swarm run that has the best average BPB and satisfies the data repetition constraints.
- LightGBM: we use the hyperparameters from RegMix (Liu et al., 2025a). However, we did not use an evaluation set for early stopping, since we noticed that RegMix used the same set of mixes for early stopping and for reporting performance. We solve the optimization problem using search, as is done in RegMix.
- Gaussian Process: we use scikit-learn's GaussianProcessRegressor. We solve the optimization problem using search, as is done in Chen et al. (2025b).
- BiMix: we use SciPy least squares to fit the regression models. We solve the optimization problem using search, an exact solver, and exact solver with KL regularization ($\lambda = 0.05$) in Table 6. Based on the results, we decided to use a search-based solver when evaluating BiMix.
- AutoScale: we use SciPy least squares to fit the regression models. We solve the optimization problem using search, an exact solver, and exact solver with KL regularization ($\lambda = 0.05$)in Table 6. Based on the results, we decide to use an exact solver when evaluating Autoscale.
- Log-linear: we use the fitting code from Ye et al. (2025). We solve the optimization problem using CVXPY, while results for other solvers are discussed in Appendix B.3.

C.5 RQ7: Regression granularity details

Additional results Figure 13 shows the performance of various regression granularities at the 30M scale. While the downstream BPB results at the 1B scale (Table 4) demonstrates that per-family performs the worst, the 30M performance results are in line with the regression fits of each granularity; per-task performs the best, followed by per-family and aggregated.

Details The task families we used are defined in Table 10: Math, Code, and QA. We constructed a held-out test set containing of 45 samples, consisting of 15 samples from a Dirichlet distribution centered around each regression granularity's respective proposed mix. This measures regression fit in high-performance regions of the mixture space, which matters in the mixture optimization step.

Table 7: Performance and repetition constraint satisfaction. Using a constrained swarm with unconstrained optimization does not satisfy a repetition constraint with $k=4$; the proposed mix repeats samples of a domain 5 times. Between the unconstrained and constrained swarm with constrained optimization (approaches 2 and 3), the latter approach achieves the best downstream performance.

Approach	Satisfies rep constraint? ($k=4$)	Average task BPB ↓
1) Constrained Swarm, Unconstrained Opt	No (5)	0.774694
2) Unconstrained Swarm, Constrained Opt	Yes (4)	0.764718
3) Constrained Swarm, Constrained Opt	Yes (4)	0.785517

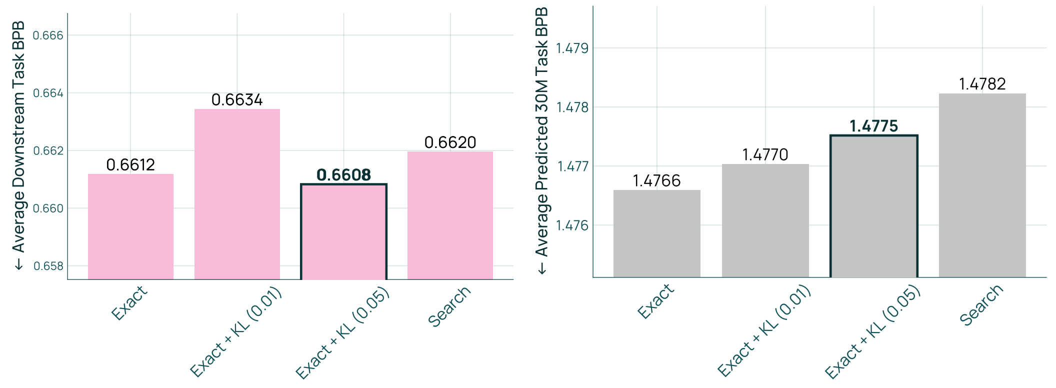

Figure 14: Performance (left) and regression fit (right) across optimization solvers when mixing across sources. The predicted performances of various solvers confirms that the exact solver minimizes the predicted BPB, as expected, followed by KL penalties of 0.01 and 0.05. However, the exact solver does not obtain the best downstream performance, and instead some KL regularization helps.

C.6 RQ4: Data repetition constraints details

Additional results We study several ways of enforcing repetition constraints. 1) We use a constrained swarm and unconstrained optimization: we construct a swarm where all of its mixes satisfy the repetition constraint. Mixture optimization solves the unconstrained problem minimize$_{p\in\triangle^{m-1}} \hat{f}(p)$, relying on the regression models to encode what mixes are feasible. 2) We consider an unconstrained swarm and constrained optimization: we sample swarm mixes from $\mathcal{P}$ without restrictions, and we add the repetition constraints to the mixture optimization problem. 3) We consider both a constrained swarm and constrained optimization. We test with $k=4$, and for each approach we verify whether the proposed mixture satisfies the repetition constraint.

Table 7 shows that approach 2, using an unconstrained swarm with constrained optimization results in a proposed mix that satisfies the repetition constraints while maintaining performance. Approach 1) fails to satisfy the constraints, resulting in samples from one of the domains being repeated 5 times, which is more than $k=4$. Approach 3) satisfies constraints, but yields worse downstream performance than 2), likely because the constrained swarm provides less coverage of the mixture space.

Details To construct the constrained swarm, we used $K=128$ proxy runs. We set $R=6$T requested tokens and a repetition factor of $k=4$. The constrained swarm is constructed via rejection sampling from the Dirichlet prior. For constrained optimization, we specified the repetition constraint in CVXPY.

C.7 RQ8: Optimization solver details

Additional results Figure 14 shows the performance and predicted performance across optimization solvers when mixing at the source level. Similar to Figure 7, we see that Exact + KL (0.05) outperforms the other solvers in terms of downstream BPB. We also sanity check our optimizers by examining predicted performance, confirming that the exact optimizer obtains the lowest predicted performance while adding a KL term degrades performance.

Details For the search approach, we used the adaptive search proposed in Wettig et al. (2025), which proceeds in multiple rounds such that the best mix from the current round is used as a Dirichlet prior to generate candidate mixes for the next round.

D Algorithm Details

D.1 Solving the Conditional Mixing Problem

We explain how to solve (3), by using a change of variables that transforms it into a standard mixing problem (without the explicit $q_{\mathcal{D}_{\text{fix}}} = \tilde{p}_{\mathcal{D}_{\text{fix}}}$ constraint) over a *collapsed* space. After this change of variables, the constraint is absorbed into the variable and standard mixing methods like OlmixBase can be applied in the collapsed space.

For any feasible $q = [\rho \tilde{p}_{\mathcal{D}_{\text{fix}}}, (1-\rho) q_{\mathcal{D}_{\text{comp}}}]$, let the corresponding mix in collapsed space be $r = [\rho, (1-\rho) q_{\mathcal{D}_{\text{comp}}}] \in \triangle^{|\mathcal{D}_{\text{comp}}|}$. r is defined over a virtual domain, containing all of $\mathcal{D}_{\text{fix}}$, and the domains in $\mathcal{D}_{\text{comp}}$. Formally, we index the elements of r by $\{v\} \cup \mathcal{D}_{\text{comp}}$ and define mapping functions that convert between r and q.

- Collapse $\Psi(q) = r$: sum all weights on $\mathcal{D}_{\text{fix}}$ into the virtual domain weight $r_v = \sum_{j \in \mathcal{D}_{\text{fix}}} q_j$, and keep weights on $\mathcal{D}_{\text{comp}}$ the same, $r_j = q_j$ for $j \in \mathcal{D}_{\text{comp}}$.
- Expand $\Phi_{\tilde{p}_{\mathcal{D}_{\text{fix}}}}(r) = q$: multiply the virtual domain weight by $\tilde{p}_{\mathcal{D}_{\text{fix}}}$, i.e., $q_j = r_v \cdot \tilde{p}_j$ for $j \in \mathcal{D}_{\text{fix}}$, and keep weights on $\mathcal{D}_{\text{comp}}$ the same, $q_j = r_j$ for $j \in \mathcal{D}_{\text{comp}}$.

Then, we can write the mixture reuse problem (3) in terms of r:

$$\begin{aligned} &\text{minimize}_{r \in \triangle^{|\mathcal{D}_{\text{comp}}|}} \frac{1}{n} \sum_{i=1}^{n} g_i(r) \\ &\text{s.t.} \quad r_v \le \min_{j \in \mathcal{D}_{\text{fix}}} \left\{ \frac{k N'_j}{R \tilde{p}_j} \right\} \quad r_j \le \frac{k N'_j}{R} \quad \forall j \in \mathcal{D}_{\text{comp}} \end{aligned} \tag{11}$$

where $g_i(r) := f_i(\text{LM}(S, R, \Phi_{\tilde{p}_{\mathcal{D}_{\text{fix}}}}(r)))$ is the performance of the mix $\Phi_{\tilde{p}_{\mathcal{D}_{\text{fix}}}}(r)$ on task i, and the repetition constraints have been rewritten in terms of r. This optimization problem looks similar to (2)—determine the mix r that minimizes the average BPB, subject to some per-domain constraints. Therefore, we can apply OlmixBase to obtain our FullMixtureReuse method, detailed in Algorithm 2. The blue coloring indicates aspects of this algorithm that are different from OlmixBase, showing that the key difference is simply two expansion steps, and otherwise handling everything in collapsed space over r.

D.2 Swarm Reuse Algorithm

We present Algorithm 3, corresponding to the Swarm Reuse approach in §5, as another way to reuse prior information from mixing. The blue coloring indicates aspects of this algorithm that are different from OlmixBase.

E Proofs for Section 4.3

We first provide definitions, assumptions, and notations.

E.1 Definitions and Notation

Define performance of q as $F(q) := \frac{1}{n} \sum_{i=1}^{n} f_i(\text{LM}(S, R, q))$. We also write $F(q)$ as a bivariate version, $F(r, p_1) := F(\Phi_{p_1}(r))$.

Recall $q^\star$ is the optimal solution to (2) on $\mathcal{D}'$, and $q^\star(\tilde{p}_{\mathcal{D}_{\text{fix}}})$ is solution returned by FullMixtureReuse when reusing $\tilde{p}_{\mathcal{D}_{\text{fix}}}$ (3). We define $q^\star_{\mathcal{D}_{\text{fix}}}$ as the optimal mix on $\mathcal{D}_{\text{fix}}$ after the domain set update; that is, $q^\star = [\rho^\star q^\star_{\mathcal{D}_{\text{fix}}}, \ (1-\rho^\star) q^\star_{\mathcal{D}_{\text{comp}}}]$. We can also write $q^\star(\tilde{p}_{\mathcal{D}_{\text{fix}}}) = [\rho^\star(\tilde{p}_{\mathcal{D}_{\text{fix}}} \tilde{p}_{\mathcal{D}_{\text{fix}}}, (1-\rho^\star(\tilde{p}_{\mathcal{D}_{\text{fix}}})) q^\star_{\mathcal{D}_{\text{comp}}}(\tilde{p}_{\mathcal{D}_{\text{fix}}})]$.

Algorithm 2 FullMixtureReuse

```
1: Input: Domain set $\mathcal{D}' = \mathcal{D}_{\text{fix}} \cup \mathcal{D}_{\text{comp}}$, existing mix $\tilde{p} \in \triangle^{m-1}$, swarm size $K = \mathcal{O}(m)$, repetition factor $k$, requested tokens $R$.
2: Sample mixes $r^1, \ldots, r^K \in \triangle^{|\mathcal{D}_{\text{comp}}|}$ and expand each collapsed mix $r^j$ into $q^j := \Phi_{\tilde{p}_{\mathcal{D}_{\text{fix}}}}(r^j)$.
3: Train proxy models on the expanded mixes and evaluate on downstream tasks to get a dataset of mixes and performance, $\{(r^j, \{y_{ij}\}_{i=1}^n\}_{j=1}^K$, where $y_{ij} := f_i(\text{LM}(S_{\text{small}}, R_{\text{small}}, q^j))$.
4: for $i \in [n]$ do
5:     Use $\{(r^j, y_{ij})\}_{j=1}^K$ to fit the log-linear model $\hat{g}_i(r) = d_i + \exp(B_i^\top r)$, where $d_i \in \mathbb{R}^+$ and $B_i \in \mathbb{R}^{|\mathcal{D}_{\text{comp}}|}$.
6: end for
7: Solve the following optimization problem to get $r^\star(\tilde{p}_{\mathcal{D}_{\text{fix}}})$:
```

$$\text{minimize}_{r \in \triangle^{|\mathcal{D}_{\text{comp}}|}} \frac{1}{n} \sum_{i=1}^n \hat{g}_i(r) \tag{12}$$

$$\text{s.t.} \quad r_v \leq \min_{j \in \mathcal{D}_{\text{fix}}} \left\{ \frac{kN'_j}{R\tilde{p}_j} \right\}, r_j \leq \frac{kN'_j}{R} \quad \forall j \in \mathcal{D}_{\text{comp}}$$

```
8: Return $q^\star(\tilde{p}_{\mathcal{D}_{\text{fix}}}) := \Phi_{\tilde{p}_{\mathcal{D}_{\text{fix}}}}(r^\star(\tilde{p}_{\mathcal{D}_{\text{fix}}}))$, the expanded form of $r^\star(\tilde{p}_{\mathcal{D}_{\text{fix}}})$.
```

We can also write $\tilde{p}$ in terms of $\mathcal{D}_{\text{fix}}$ and $\mathcal{D} \backslash \mathcal{D}_{\text{fix}}$: any p can be expressed as $p = [\pi p_{\mathcal{D}_{\text{fix}}}, (1-\pi) p_{\mathcal{D} \backslash \mathcal{D}_{\text{fix}}}]$, where $\pi \in [0,1]$ (similar to ρ).

The log-linear model is $f_i(q) = c_i + \sum_{i=1}^n \exp(A_i^\top q)$, where A_{ij} intuitively captures a notion of how much domain j impacts task i. We split each A_i into $[A_{i,\text{fix}}, A_{i,\text{comp}}]$ corresponding to $\mathcal{D}_{\text{fix}}$ and $\mathcal{D}_{\text{comp}}$. Define $\alpha_{\text{fix}}, \alpha_{\text{comp}} \in \mathbb{R}^n$ where $\alpha_{i,\text{fix}} = \|A_{i,\text{fix}}\|, \alpha_{i,\text{comp}} = \|A_{i,\text{comp}}\|$.

Define the coupling term $\kappa(\mathcal{D}_{\text{fix}}, \mathcal{D}_{\text{comp}}) = \kappa(\alpha_{\text{fix}}, \alpha_{\text{comp}})$ as

$$\kappa(\alpha_{\text{fix}}, \alpha_{\text{comp}}) := \|(1 + \alpha_{\text{fix}} + \alpha_{\text{comp}}) \odot \alpha_{\text{fix}}\|, \tag{14}$$

where $\odot$ is the Hadamard product.

Define $\bar{F}(\cdot)$ to be the objective without the constant term of the log-linear regression model, $F(\cdot) - \frac{1}{n}\sum_{i=1}^n c_i$.

Denote $p_1 = \tilde{p}_{\mathcal{D}_{\text{fix}}}$ and $p'_1 = q^\star_{\mathcal{D}_{\text{fix}}}$ for simplicity. Let $p_2 = q_{\mathcal{D}_{\text{comp}}}$. Recall that $r^\star(p_1)$ is the solution to (3) when $q_{\mathcal{D}_{\text{fix}}}$ is constrained to be p_1. Any r can be written as $r = [\rho, (1-\rho), p_2]$, so we similarly define $\rho^\star(p_1)$ and $p_2^\star(p_1)$.

E.2 Proof for Theorem 4.1

Assumption E.1. We make the following assumptions:

1. The log-linear model accurately captures the relationship between the mixture ratios and target model performance on each task; that is, $f_i(\text{LM}(S,R,q)) = c_i + \exp(A_i^\top q)$ for some $c_i \in \mathbb{R}^+, A_i \in \mathbb{R}^{m'}$.
2. For simplicity, applying OlmixBase (Algorithm 1) exactly solves (2). That is, we ignore (i) proxy-to-target transfer error and (ii) estimation error in learning c_i and A_i from the swarm. However, extending the analysis to include these errors is straightforward by adding corresponding approximation terms.
3. We assume that $F(r, p_1)$ is μ-strongly convex in r.
4. Let $\mathcal{S}(q_{\mathcal{D}_{\text{fix}}})$ be the feasible set of (11) defined by repetition constraints:

$$r_v \leq \min_{j \in \mathcal{D}_{\text{fix}}} \left\{ \frac{kN'_j}{Rq_j} \right\}, r_j \leq \frac{kN'_j}{R} \; \forall j \in \mathcal{D}'_2,$$

together with the simplex constraints on r. We assume **mutual feasibility** of $\tilde{p}_{\mathcal{D}_{\text{fix}}}$ and $q^\star_{\mathcal{D}_{\text{fix}}}$: that $r^\star(\tilde{p}_{\mathcal{D}_{\text{fix}}}) \in \mathcal{S}(q^\star_{\mathcal{D}_{\text{fix}}})$ and $r^\star(q^\star_{\mathcal{D}_{\text{fix}}}) \in \mathcal{S}(\tilde{p}_{\mathcal{D}_{\text{fix}}})$. This holds in the following two cases:

Algorithm 3 Swarm Reuse Mixing

```
1: Input: Old domain set $\mathcal{D}$, new domain set $\mathcal{D}'$, old swarm $\{(p^j_{\text{old}}, \{y^{\text{old}}_{ij}\}^n_{i=1})\}^{K_{\text{old}}}_{j=1}$, new swarm size $K_{\text{new}} = \mathcal{O}(|\mathcal{D}'|)$, Dirichlet distribution $\mathcal{P}'$, repetition factor $k$, requested tokens $R$.
2: Sample new mixes $p^1_{\text{new}}, \dots, p^{K_{\text{new}}}_{\text{new}} \sim \mathcal{P}'$ on $\mathcal{D}'$.
3: Train proxy models on new mixes and evaluate on downstream tasks to get new swarm data $\{(p^j_{\text{new}}, \{y^{\text{new}}_{ij}\}^n_{i=1}\}^{K_{\text{new}}}_{j=1}$.
4: Map old swarm mixes to new domain set: for $j \in [K_{\text{old}}]$, compute $\tilde{p}^j_{\text{old}} \in \triangle^{m'-1}$ from $p^j_{\text{old}} \in \triangle^{m-1}$ where:
     • If Add: $\mathcal{D} = [\mathcal{D}_1, \emptyset]$ and $\mathcal{D}' = [\mathcal{D}_1, \mathcal{D}'_2]$. Then $\tilde{p}^j_{\text{old}}(D'_i) = p^j_{\text{old}}(D_i)$ for $D_i \in \mathcal{D}_1$, and $\tilde{p}^j_{\text{old}}(D'_i) = 0$ for $D'_i \in \mathcal{D}'_2$.
     • If Partition: $\mathcal{D} = [\mathcal{D}_1, \{D_{\text{par}}\}]$ and $\mathcal{D}' = [\mathcal{D}_1, \{D^1_{\text{par}}, \dots, D^\ell_{\text{par}}\}]$ where $D_{\text{par}} = \bigcup_{k=1}^\ell D^k_{\text{par}}$. Then $\tilde{p}^j_{\text{old}}(D'_i) = p^j_{\text{old}}(D_i)$ for $D_i \in \mathcal{D}_1$, and $\tilde{p}^j_{\text{old}}(D^k_{\text{par}}) = p^j_{\text{old}}(D_{\text{par}}) \cdot \frac{N_{D^k_{\text{par}}}}{\sum_{k'=1}^\ell N_{D^{k'}_{\text{par}}}}$ for the partitioned domains.
5: Construct combined swarm: $\mathcal{S} = \{(\tilde{p}^j_{\text{old}}, \{y^{\text{old}}_{ij}\})\}^{K_{\text{old}}}_{j=1} \cup \{(p^j_{\text{new}}, \{y^{\text{new}}_{ij}\})\}^{K_{\text{new}}}_{j=1}$.
6: for $i \in [n]$ do
7:     Use combined swarm $\mathcal{S}$ to fit the log-linear model $\hat{f}_i(p) = c_i + \exp(A_i^\top p)$, where $c_i \in \mathbb{R}^+$ and $A_i \in \mathbb{R}^{|\mathcal{D}'|}$.
8: end for
9: Solve the optimization problem:
       $\text{minimize}_{p \in \triangle^{|\mathcal{D}'|-1}} \frac{1}{n}\sum_{i=1}^n \hat{f}_i(p)$     (13)
       $\text{subject to} \quad p_j \leq \frac{kN'_j}{R} \; \forall j \in |\mathcal{D}'|$
10: Return $p^\star$, the solution on new domain set $\mathcal{D}'$.
```

- Both $\tilde{p}_{\mathcal{D}_{\text{fix}}}$ and $q^\star_{\mathcal{D}_{\text{fix}}}$ have an active repetition constraint on at least one domain. Intuitively, this means that among the unaffected domains, there exist some high-utility domains that are data-constrained.
- $r^\star_v(\tilde{p}_{\mathcal{D}_{\text{fix}}}) \leq \min_{j \in \mathcal{D}_{\text{fix}}}\{\frac{kN'_j}{Rq_j}\}$ and $r^\star_v(\tilde{p}_{\mathcal{D}_{\text{fix}}}) \leq \min_{j \in \mathcal{D}_{\text{fix}}}\{\frac{kN'_j}{R\tilde{p}_j}\}$. Intuitively, the contents of the virtual domain are sufficiently low utility such that the proposed mix does not "push against" the tighter of the two r_v constraints.

This assumption is reasonable in our setting: when the repetition constraint is active, a domain is valuable enough that we would allocate more weight if we could. When it is slack, the domain does not warrant allocating weight up to its availability limit. Our two cases simply rule out the marginal situation where one mix is availability-limited while the other is not.

First, we show that the mixture reuse problem and the standard mixing problem are equivalent when the existing mix on the unaffected domains is equal to the optimal mix on the unaffected domains after the domain update. This is the "equality" condition of Theorem 4.1: when $\tilde{p}_{\mathcal{D}_{\text{fix}}} = q^\star_{\mathcal{D}_{\text{fix}}}$, the performance gap is 0.

Lemma E.2. *The solution of* (3), $r^\star(\tilde{p}_{\mathcal{D}_{\text{fix}}})$, *is equivalent to the solution of* (2), $q^\star = [\rho^\star q^\star_{\mathcal{D}_{\text{fix}}}, (1-\rho^\star) q^\star_{\mathcal{D}'_2}]$, *when the existing mix satisfies* $\tilde{p}_{\mathcal{D}_{\text{fix}}} = q^\star_{\mathcal{D}_{\text{fix}}}$. *In other words,* $\Phi_{q^\star_{\mathcal{D}_{\text{fix}}}}(r^\star(q^\star_{\mathcal{D}_{\text{fix}}})) = q^\star$, *and* $q^\star(q^\star_{\mathcal{D}_{\text{fix}}}) = q^\star$.

Proof. For (2), restricting the feasible set to the subspace $q_{\mathcal{D}_{\text{fix}}} = q^\star_{\mathcal{D}_{\text{fix}}}$ does not change the optimal value of this problem, since this subspace still contains the optimal solution to the original problem.

Therefore, (2) is equivalent to:

$$\begin{aligned} &\text{minimize}_{q\in\triangle^{m'-1}} \frac{1}{n}\sum_{i=1}^{m} f_i(\text{LM}(S,R,q)) \\ &\text{s.t.} \quad q_j \leq \frac{kN'_j}{R} \quad \forall j\in[m'] \\ &\qquad\; q_{\mathcal{D}_{\text{fix}}} = q^\star_{\mathcal{D}_{\text{fix}}} \end{aligned}$$

Comparing this formulation of (2) to (3), we can see that the problems are equivalent when $\tilde{p}_{\mathcal{D}_{\text{fix}}} = q^\star_{\mathcal{D}_{\text{fix}}}$. □

Based on this result, we can write $q^\star = q^\star(q^\star_{\mathcal{D}_{\text{fix}}}) = q^\star(p'_1)$. Similarly, $\rho^\star = \rho^\star(p'_1)$, $r^\star = r^\star(p'_1)$, $p^\star_2 = p^\star_2(p'_1)$. Furthermore, the problem of bounding the performance gap now becomes one of bounding $F(q^\star(\tilde{p}_{\mathcal{D}_{\text{fix}}})) - F(q^\star(q^\star_{\mathcal{D}_{\text{fix}}})$ in terms of $\|q^\star(\tilde{p}_{\mathcal{D}_{\text{fix}}}) - q^\star_{Dfix}\|$. We can apply standard tools from optimization and convexity over $F(r,p_1)$ to address this.

E.2.1 Performance Gap Lemmas

Lemma E.3 (FOC Inequality)**.** *Recall that* $r^\star(p_1) = \arg\min_{r\in\mathcal{S}(p_1)} F(r,\ p_1)$ *and* $r^\star(p'_1) = \operatorname{argmin}_{r\in\mathcal{S}(p'_1)} F(r,p'_1)$. *Then,*

$$\langle \nabla_r F(r^\star(p_1),p_1) - \nabla_r F(r^\star(p'_1),p'_1), r^\star(p'_1) - r^\star(p_1)\rangle \geq 0.$$

Proof. Since $r^\star(p_1)$ minimizes $F(\cdot,p_1)$ over the convex set $\mathcal{S}(p_1)$, we have the first-order condition

$$\langle \nabla_r F(r^\star(p_1),p_1), r - r^\star(p_1)\rangle \geq 0 \quad \forall r\in\mathcal{S}(p_1).$$

Using the mutual feasibility assumption, $r^\star(p'_1)\in\mathcal{S}(p_1)$, so plugging $r = r^\star(p'_1)$ yields

$$\langle \nabla_r F(r^\star(p_1),p_1), r^\star(p'_1) - r^\star(p_1)\rangle \geq 0.$$

Similarly, since $r^\star(p'_1)$ minimizes $F(\cdot,p'_1)$ over $\mathcal{S}(p'_1)$ and $r^\star(p_1)\in\mathcal{S}(p'_1)$,

$$\langle \nabla_r F(r^\star(p'_1),p'_1), r^\star(p_1) - r^\star(p'_1)\rangle \geq 0.$$

Adding the two inequalities completes the proof. □

Lemma E.4 (Mean value bound for $\nabla_r F$ along a segment)**.** *Fix* $r\in\mathbb{R}^{|\mathcal{D}'_2|+1}$ *and consider the map*

$$G(\cdot) := \nabla_r F(r,\cdot).$$

Assume G is continuously differentiable on the line segment

$$p_1^t := tp'_1 + (1-t)p_1, \qquad t\in[0,1].$$

Then

$$\|\nabla_r F(r,p'_1) - \nabla_r F(r,p_1)\| \leq \sup_{t\in[0,1]} \left\|\nabla^2_{r,p_1} F(r,p_1^t)\right\| \cdot \|p'_1 - p_1\|.$$

Proof. Define the scalar function

$$h(t) := G(p_1^t) \in \mathbb{R}^{|\mathcal{D}'_2|+1}, \qquad t\in[0,1].$$

Since G is continuously differentiable on the segment, h is differentiable and

$$h'(t) = \nabla_{p_1} G(p_1^t)(p'_1 - p_1),$$

where $\nabla_{p_1} G(p_1^t)$ denotes the Jacobian of G at p_1^t. By the fundamental theorem of calculus,

$$G(p'_1) - G(p_1) = h(1) - h(0) = \int_0^1 h'(t)dt = \int_0^1 \nabla_{p_1} G(p_1^t)(p'_1 - p_1)dt.$$

Taking norms and applying the triangle inequality,

$$\begin{aligned}\|G(p_1')-G(p_1)\| &\le \int_0^1 \|\nabla_{p_1} G(p_1^t)(p_1'-p_1)\| dt \\ &\le \int_0^1 \|\nabla_{p_1} G(p_1^t)\| \|p_1'-p_1\| dt \\ &\le \left(\sup_{t\in[0,1]} \|\nabla_{p_1} G(p_1^t)\|\right)\cdot \|p_1'-p_1\|.\end{aligned}$$

Substituting $G(\cdot)=\nabla_r F(r,\cdot)$ yields the equivalent mixed-derivative form. □

Lemma E.5 (Behavior of $r^\star(\cdot)$ under changes in p_1)**.** *Let* $\Delta := p_1'-p_1$ *and* $\Delta r := r^\star(p_1')-r^\star(p_1)$. *Then*

$$\|\Delta r\| \le \frac{M}{\mu}\|\Delta\|,$$

where

$$M := \sup_{t\in[0,1]} \left\|\nabla^2_{r,p_1} F\big(r^\star(p_1'),p_1^t\big)\right\|, \quad p_1^t := tp_1'+(1-t)p_1.$$

Proof. By μ-strong convexity of $F(\cdot,p_1)$ in r, for $r=r^\star(p_1)$ and $r'=r^\star(p_1')$,

$$F(r',p_1)\ge F(r,p_1)+\langle \nabla_r F(r,p_1), r'-r\rangle + \frac{\mu}{2}\|r'-r\|^2, F(r,p_1) \quad \ge F(r',p_1)+\langle \nabla_r F(r',p_1), r-r'\rangle + \frac{\mu}{2}\|r-r'\|^2.$$

Adding yields

$$\mu\|\Delta r\|^2 \le \langle \nabla_r F(r^\star(p_1'),p_1)-\nabla_r F(r^\star(p_1),p_1), \Delta r\rangle.$$

Next, we add and subtract $\nabla_r F(r^\star(p_1'),p_1')$:

$$\begin{aligned}\mu\|\Delta r\|^2 \le &\langle \nabla_r F(r^\star(p_1'),p_1)-\nabla_r F(r^\star(p_1'),p_1'), \Delta r\rangle \\ &+\langle \nabla_r F(r^\star(p_1'),p_1')-\nabla_r F(r^\star(p_1),p_1), \Delta r\rangle.\end{aligned}$$

By Lemma E.3, the second inner product is less than 0. Thus

$$\mu\|\Delta r\|^2 \le \|\nabla_r F(r^\star(p_1'),p_1)-\nabla_r F(r^\star(p_1'),p_1')\|\cdot\|\Delta r\|.$$

Finally, applying Lemma E.4, we have

$$\mu\|\Delta r\|^2 \le \sup_{t\in[0,1]} \left\|\nabla^2_{r,p_1} F(r^\star(p_1'),p_1^t)\right\|\cdot\|\Delta\|\cdot\|\Delta r\|$$

We cancel $\|\Delta r\|$ to complete the proof. □

Lemma E.6 (Decomposing the performance gap)**.** *Define* $\Delta := p_1'-p_1$, $\Delta r := r^\star(p_1')-r^\star(p_1)$. *Assume* $F(\cdot,\cdot)$ *is convex in* r *and differentiable in both arguments. Then*

$$F(r^\star(p_1),p_1)-F(r^\star(p_1'),p_1') \le \|\nabla_r F(r^\star(p_1),p_1)\|\cdot\|\Delta r\| + \|\Delta\|\cdot \sup_{t\in[0,1]} \|\nabla_{p_1} F(r^\star(p_1'),p_1^t)\|.$$

Proof. Add and subtract $F(r^\star(p_1'),p_1)$:

$$F(r^\star(p_1),p_1)-F(r^\star(p_1'),p_1') = \big(F(r^\star(p_1),p_1)-F(r^\star(p_1'),p_1)\big)+\big(F(r^\star(p_1'),p_1)-F(r^\star(p_1'),p_1')\big).$$

For the first difference, convexity of $F(\cdot,p_1)$ implies

$$F(r^\star(p_1'),p_1)\ge F(r^\star(p_1),p_1)+\langle \nabla_r F(r^\star(p_1),p_1), r^\star(p_1')-r^\star(p_1)\rangle,$$

so

$$F(r^\star(p_1),p_1)-F(r^\star(p_1'),p_1)\le \langle \nabla_r F(r^\star(p_1),p_1), r^\star(p_1)-r^\star(p_1')\rangle \le \|\nabla_r F(r^\star(p_1),p_1)\|\cdot\|\Delta r\|.$$

For the second difference, we apply the mean value theorem along p_1^t:

$$|F(r^\star(p_1'),p_1)-F(r^\star(p_1'),p_1')| \le \|\Delta\|\cdot \sup_{t\in[0,1]} \|\nabla_{p_1} F(r^\star(p_1'),p_1^t)\|.$$

Combining the two bounds completes our proof. □

E.2.2 Gradient Lemmas

Lemma E.7. *The gradient norm* $\|\nabla_r F(r^\star(p_1),p_1)\|$ *can be bounded by*

$$\nabla_r F(r^\star(p_1),p_1) \le \bar{F}(q^\star(p_1))\big\||A_1p_1|+\alpha_{comp}\big\|$$

Proof. First, we write r as $[\rho,b_2]$, where $b_2=(1-\rho)p_2$. Define $s_i = A_{i,1}^\top \rho p_1 + A_{i,2}^\top b_2$. We compute the gradient of $F(r,p_1)$ with respect to ρ and b_2:

$$\nabla_\rho F(r,p_1) = \frac{1}{n}\sum_{i=1}^n \exp(s_i) A_{i,1}^\top p_1$$

$$\nabla_{b_2} F(r,p_1) = \frac{1}{n}\sum_{i=1}^n \exp(s_i) A_{i,2}$$

We can consolidate this into a single expression if we define $a_i = \begin{bmatrix} \langle A_{i,1},p_1\rangle \\ A_{i,2}\end{bmatrix} \in \mathbb{R}^{|\mathcal{D}_2'|+1}$:

$$\nabla_r F(r,p_1) = \frac{1}{n}\sum_{i=1}^n \exp(s_i) a_i$$

Let us write $r^\star(p_1)$ as $[\rho^\star(p_1),(1-\rho^\star(p_1))p_2^\star(p_1)]$. The expression for the gradient at $r^\star(p_1)$ is

$$\nabla_r F(r^\star(p_1),p_1) = \frac{1}{n}\sum_{i=1}^n \exp(A_{i,1}^\top \rho^\star(p_1)p_1 + A_{i,2}^\top(1-\rho^\star(p_1))p_2^\star(p_1))a_i$$

We now bound the norm. First, applying the Cauchy-Schwarz inequality twice, we have:

$$\begin{aligned}
\|\nabla_r F(r^\star(p_1),p_1)\| &\le \frac{1}{n}\sum_{i=1}^n \exp(A_{i,1}^\top \rho^\star(p_1)p_1 + A_{i,2}^\top(1-\rho^\star(p_1))p_2^\star(p_1))\|a_i\| \\
&\le \frac{1}{n}\sqrt{\sum_{i=1}^n \exp(2(A_{i,1}^\top \rho^\star(p_1)p_1 + A_{i,2}^\top(1-\rho^\star(p_1))p_2^\star(p_1)))}\sqrt{\sum_{i=1}^n \|a_i\|^2} \\
&\le \frac{1}{n}\sum_{i=1}^n \exp(A_{i,1}^\top \rho^\star(p_1)p_1 + A_{i,2}^\top(1-\rho^\star(p_1))p_2^\star(p_1))\sqrt{\sum_{i=1}^n \|a_i\|^2}
\end{aligned}$$

Observe that the first summation is equivalent to $\bar{F}(q^\star(p_1))$. Then, to bound the remaining term $\sqrt{\sum_{i=1}^n \|a_i\|^2}$, we have

$$\begin{aligned}
\sqrt{\sum_{i=1}^n \|a_i\|^2} &= \sqrt{\sum_{i=1}^n \langle A_{i,1},p_1\rangle^2 + \|A_{i,2}\|^2} \\
&\le \sqrt{\sum_{i=1}^n (|\langle A_{i,1},p_1\rangle| + \|A_{i,2}\|)^2} \\
&= \big\||A_1p_1|+\alpha_{\text{comp}}\big\|
\end{aligned}$$

where $|\cdot|$ is an elementwise absolute value, and $\alpha_{\text{comp}} \in \mathbb{R}^n$ is constructed where $\alpha_{i,2} = \|A_{i,2}\|$.

Therefore,

$$\nabla_r F(r^\star(p_1),p_1) \le \bar{F}(q^\star(p_1))\big\||A_1p_1|+\alpha_{\text{comp}}\big\|$$

□

Lemma E.8. *The gradient norm $\nabla_{p_1} F(r^\star(p_1'),p_1^t)$ can be bounded by*

$$\|\nabla_{p_1} F(r^\star(p_1'),p_1^t)\| \leq \bar{F}(q^\star(p_1')) \left\| \exp(\alpha_{\mathit{fix}} \|\triangle\|) \odot \alpha_{\mathit{fix}} \right\|$$

where $\odot$ is the Hadamard product, and $\exp$ *is elementwise.*

Proof. Define $s_i = A_{i,1}^\top \rho p_1 + A_{i,2}^\top (1-\rho) p_2$. We compute the gradient of $F(r,p_1)$ with respect to p_1:

$$\nabla_{p_1} F(r,p_1) = \frac{1}{n} \sum_{i=1}^n \exp(s_i) \rho A_{i,1}$$

Let us write $r^\star(p_1')$ as $[\rho^\star(p_1'),(1-\rho^\star(p_1'))p_2^\star(p_1')]$. Then, the gradient of $F(r,p_1)$ at $r^\star(p_1'),p_1^t$ is

$$\nabla_{p_1} F(r^\star(p_1'),p_1^t) = \frac{1}{n} \sum_{i=1}^n \exp(A_{i,1}^\top \rho^\star(p_1') p_1^t + A_{i,2}^\top (1-\rho^\star(p_1')) p_2^\star(p_1')) \rho^\star(p_1') A_{i,1}$$

We now bound the norm of the gradient over $t \in [0,1]$. First, we add and subtract p_1' to create an $\exp$ term defined over p_1' and a difference term with $p_1^t - p_1'$:

$$\|\nabla_{p_1} F(r^\star(p_1'),p_1^t)\| \leq \frac{1}{n} \sum_{i=1}^n \exp(A_{i,1}^\top \rho^\star(p_1')(p_1' + p_1^t - p_1') + A_{i,2}^\top (1-\rho^\star(p_1')) p_2^\star(p_1')) \rho^\star(p_1') \|A_{i,1}\|$$

$$\leq \frac{1}{n} \sum_{i=1}^n \exp(A_{i,1}^\top \rho^\star(p_1') p_1' + A_{i,2}^\top (1-\rho^\star(p_1')) p_2^\star(p_1')) \exp(A_{i,1}^\top \rho^\star(p_1')(p_1^t - p_1')) \|A_{i,1}\|$$

Then, we apply Cauchy-Schwarz inequality to get
$\|\nabla_{p_1} F(r^\star(p_1'),p_1^t)\|$

$$\leq \frac{1}{n} \sqrt{\sum_{i=1}^n \exp(2A_{i,1}^\top \rho^\star(p_1')(p_1^t - p_1')) \|A_{i,1}\|^2} \sqrt{\sum_{i=1}^n \exp(2(A_{i,1}^\top \rho^\star(p_1') p_1' + A_{i,2}^\top (1-\rho^\star(p_1')) p_2^\star(p_1')))}$$

$$\leq \frac{1}{n} \sqrt{\sum_{i=1}^n \exp(2A_{i,1}^\top \rho^\star(p_1')(p_1^t - p_1')) \|A_{i,1}\|^2} \cdot \sum_{i=1}^n \exp(A_{i,1}^\top \rho^\star(p_1') p_1' + A_{i,2}^\top (1-\rho^\star(p_1')) p_2^\star(p_1'))$$

We observe that we can replace the second summation with $\bar{F}(q^\star(p_1'))$, so we only need to further simplify $\sqrt{\sum_{i=1}^n \exp(2A_{i,1}^\top \rho^\star(p_1')(p_1^t - p_1')) \|A_{i,1}\|^2}$. We observe that $\|p_1^t - p_1'\|$ is always bounded by $\|\Delta\|$, and use the fact that $\rho^\star(p_1') \leq 1$:

$$\sqrt{\sum_{i=1}^n \exp(2A_{i,1}^\top \rho^\star(p_1')(p_1^t - p_1')) \|A_{i,1}\|^2} \leq \sqrt{\sum_{i=1}^n \exp(2\|A_{i,1}\| \|p_1^t - p_1'\|) \|A_{i,1}\|^2}$$

$$\leq \sqrt{\sum_{i=1}^n \exp(2\|A_{i,1}\| \|\triangle\|) \|A_{i,1}\|^2} \leq \left\| \exp(\alpha_{\text{fix}} \|\triangle\|) \odot \alpha_{\text{fix}} \right\|$$

Therefore,

$$\|\nabla_{p_1} F(r^\star(p_1'),p_1^t)\| \leq \bar{F}(q^\star(p_1')) \left\| \exp(\alpha_{\text{fix}} \|\triangle\|) \odot \alpha_{\text{fix}} \right\|$$

□

Lemma E.9. *Let $a_{\max} = \max_i \{\|A_{i,1}\|\}$, the largest norm of $A_{i,1}$ across all tasks. Define $b_{\max} \in \mathbb{R}^n$ where $b_{i,\max} = \max\{|\langle A_{i,1}, p_1 \rangle|, |\langle A_{i,1}, p_1' \rangle|\}$ as the larger of the two dot products per task. The gradient norm $\|\nabla^2_{r,p_1} F(r^\star(p_1'),p_1^t)\|$ can be bounded by*

$$\|\nabla^2_{r,p_1} F(r^\star(p_1'),p_1^t)\| \leq \exp(a_{\max} \|\Delta\|) \cdot \bar{F}(q^\star(p_1')) \cdot \left\| (\mathbf{1} + b_{\max} + \alpha_{\mathit{comp}}) \odot \alpha_{\mathit{fix}} \right\|$$

where $\odot$ is the Hadamard product.

Proof. We derive an expression for $\nabla^2_{r,p_1} F(r,p_1)$. We write r as $[\rho, b_2]$, where $b_2 := (1-\rho)p_2$, and define $s_i = A_{i,1}^\top \rho p_1 + A_{i,2}^\top b_2$ to be the expression inside the exp. First, we have that $\nabla_{p_1} F(r,p_1)$ is

$$\nabla_{p_1} F(r,p_1) = \frac{1}{n}\sum_{i=1}^{n} \exp(s_i)\rho A_{i,1}.$$

Then, we can compute $\nabla_{\rho,p_1} F(r,p_1)$ and $\nabla_{b_2,p_1} F(r,p_1)$:

$$\nabla_{\rho,p_1} F(r,p_1) = \nabla_\rho\left(\frac{1}{n}\sum_{i=1}^{n}\exp(s_i)\rho A_{i,1}\right) = \frac{1}{n}\sum_{i=1}^{n}(\rho\exp(s_i)\langle A_{i,1},p_1\rangle + \exp(s_i))A_{i,1}^\top$$
$$\nabla_{b_2,p_1} F(r,p_1) = \nabla_{b_2}\left(\frac{1}{n}\sum_{i=1}^{n}\exp(s_i)\rho A_{i,1}\right) = \frac{1}{n}\sum_{i=1}^{n}\rho\exp(s_i)A_{i,2}A_{i,1}^\top$$

We can consolidate these into

$$\nabla^2_{r,p_1} F(r,p_1) = \frac{1}{n}\sum_{i=1}^{n}\exp(s_i)(\rho a_i + e_1)A_{i,1}^\top,$$

where $a_i = \begin{bmatrix}\langle A_{i,1},p_1\rangle \\ A_{i,2}\end{bmatrix} \in \mathbb{R}^{|\mathcal{D}_2'|+1}$ and e_1 is the standard basis vector $\begin{bmatrix}1\\0\\ \vdots \\ 0\end{bmatrix} \in \mathbb{R}^{|\mathcal{D}_2'|+1}$. Therefore, the gradient $\nabla^2_{r,p_1} F(r,p_1)$ at $r^\star(p_1'), p_1^t$ is

$$\nabla^2_{r,p_1} F(r^\star(p_1'),p_1^t) = \frac{1}{n}\sum_{i=1}^{n}\exp(A_{i,1}^\top\rho^\star(p_1')p_1^t + A_{i,2}^\top(1-\rho^\star(p_1'))p_2^\star(p_1'))(\rho^\star(p_1')a_i + e_1)A_{i,1}^\top$$

where a_i is now $\begin{bmatrix}\langle A_{i,1},p_1^t\rangle \\ A_{i,2}\end{bmatrix}$. Now, we would like to upper bound $\sup_{t\in[0,1]}\|\nabla^2_{r,p_1}F(r^\star(p_1'),p_1^t)\|$. Using Cauchy-Schwarz inequality, we have

$$\|\nabla^2_{r,p_1} F(r^\star(p_1'),p_1^t)\| \le \frac{1}{n}\sum_{i=1}^{n}\exp(A_{i,1}^\top\rho^\star(p_1')p_1^t + A_{i,2}^\top(1-\rho^\star(p_1'))p_2^\star(p_1'))\|(\rho^\star(p_1')a_i + e_1)A_{i,1}^\top\|$$
$$\le \frac{1}{n}\sqrt{\sum_{i=1}^{n}\exp\big(2(A_{i,1}^\top\rho^\star(p_1')p_1^t + A_{i,2}^\top(1-\rho^\star(p_1'))p_2^\star(p_1'))\big)}\sqrt{\sum_{i=1}^{n}\|(\rho^\star(p_1')a_i + e_1)A_{i,1}^\top\|^2} \qquad (15)$$

We simplify the first summation in (15) by adding and subtracting p_1' to p_1^t:

$$\sum_{i=1}^{n}\exp\big(2(A_{i,1}^\top\rho^\star(p_1')p_1^t + A_{i,2}^\top(1-\rho^\star(p_1'))p_2^\star(p_1'))\big)$$
$$= \sum_{i=1}^{n}\exp\big(2A_{i,1}^\top\rho^\star(p_1')(p_1' + p_1^t - p_1') + 2A_{i,2}^\top(1-\rho^\star(p_1'))p_2^\star(p_1')\big)$$
$$= \sum_{i=1}^{n}\exp\big(2(A_{i,1}^\top\rho^\star(p_1')p_1' + A_{i,2}^\top(1-\rho^\star(p_1'))p_2^\star(p_1')) + 2A_{i,1}^\top\rho^\star(p_1')(p_1^t - p_1')\big)$$
$$\le \max_i\{\exp(2A_{i,1}^\top\rho^\star(p_1')(p_1^t - p_1'))\}\left(\sum_{i=1}^{n}\exp\big(A_{i,1}^\top\rho^\star(p_1')p_1' + A_{i,2}^\top(1-\rho^\star(p_1'))p_2^\star(p_1')\big)\right)^2$$

We note that $\sum_{i=1}^{n}\exp\big(A_{i,1}^\top\rho^\star(p_1')p_1' + A_{i,2}^\top(1-\rho^\star(p_1'))p_2^\star(p_1')\big)$ is $n\cdot\bar{F}(q^\star(p_1'))$. Moreover, using the fact that $\rho^\star(p_1') \le 1$ and $\|p_1^t - p_1'\| \le \|\Delta\|$, we can bound

$\max_i\{\exp(2A_{i,1}^\top \rho^\star(p_1')(p_1^t-p_1'))\} \leq \max_i\{\exp(2\|A_{i,1}\|\|p_1^t-p_1'\|)\} \leq \exp(2\|\Delta\|\max_i\{\|A_{i,1}\|\})$. Let $a_{\max}=\max_i\{\|A_{i,1}\|\}$, the largest norm of $A_{i,1}$ across all tasks. Therefore, we have:

$$\frac{1}{n}\sqrt{\sum_{i=1}^n \exp\big(2(A_{i,1}^\top\rho^\star(p_1')p_1^t + A_{i,2}^\top(1-\rho^\star(p_1'))p_2^\star(p_1'))\big)} \leq \frac{1}{n}\sqrt{\exp(2a_{\max}\|\Delta\|)\cdot n^2\cdot F(q^\star(p_1'))^2}$$
$$= \exp(a_{\max}\|\Delta\|)\cdot F(q^\star(p_1'))$$

Next, we simplify the second summation in (15), $\sum_{i=1}^n\|(\rho^\star(p_1')a_i+e_1)A_{i,1}^\top\|^2$, using Cauchy-Schwarz inequality, the definition of the norm, and the fact that $\rho^\star(p_1')\leq 1$:

$$\sum_{i=1}^n\|(\rho^\star(p_1')a_i+e_1)A_{i,1}^\top\|^2 \leq \sum_{i=1}^n\|\rho^\star(p_1')a_i+e_1\|^2\|A_{i,1}\|^2$$
$$= \sum_{i=1}^n((1+\rho^\star(p_1')\langle A_{i,1},p_1^t\rangle)^2+\rho^\star(p_1')^2\|A_{i,2}\|^2)\|A_{i,1}\|^2$$
$$\leq \sum_{i=1}^n((1+\rho^\star(p_1')\langle A_{i,1},p_1^t\rangle)^2+\|A_{i,2}\|^2)\|A_{i,1}\|^2$$

Note that $\langle A_{i,1},\ p_1^t\rangle$ is bound by $\langle A_{i,1},\ p_1'\rangle$ and $\langle A_{i,1},\ p_1\rangle$. Therefore, it holds that $(1+\rho^\star(p_1')\langle A_{i,1},\ p_1^t\rangle)^2 \leq (1+\max\{|\langle A_{i,1},\ p_1\rangle|,\ |\langle A_{i,1},\ p_1'\rangle|\})^2$. We denote $b_{i,\max}=\max\{|\langle A_{i,1},p_1\rangle|,|\langle A_{i,1},p_1'\rangle|\}$ as the larger of the two dot products per task, and then we can write:

$$\sqrt{\sum_{i=1}^n\|(\rho^\star(p_1')a_i+e_1)A_{i,1}^\top\|^2} \leq \sqrt{\sum_{i=1}^n((1+b_{i,\max})^2+\|A_{i,2}\|^2)\|A_{i,1}\|^2}$$
$$\leq \sqrt{\sum_{i=1}^n(1+b_{i,\max}+\|A_{i,2}\|)^2\|A_{i,1}\|^2}$$
$$= \big\|(\mathbf{1}+b_{\max}+\alpha_{\text{comp}})\odot\alpha_{\text{fix}}\big\|$$

where addition is elementwise, $\odot$ is the Hadamard product, and $\alpha_{\text{fix}},\alpha_{\text{comp}}\in\mathbb{R}^n$ are constructed where $\alpha_{i,1}=\|A_{i,1}\|$ and $\alpha_{i,2}=\|A_{i,2}\|$. Putting everything together, our gradient norm bound in (15) becomes

$$\|\nabla^2_{r,p_1}F(r^\star(p_1'),p_1^t)\| \leq \exp(a_{\max}\|\Delta\|)\cdot\bar{F}(q^\star(p_1'))\cdot\big\|(\mathbf{1}+b_{\max}+\alpha_{\text{comp}})\odot\alpha_{\text{fix}}\big\|$$

□

E.2.3 THEOREM 4.1 (FORMAL)

Theorem E.10. *Define $a_{\max}=\max_i\{\|A_{i,1}\|\}$. The performance gap between using $q^\star$ and $q^\star(\tilde{p}_{\mathcal{D}_{fix}})$ is bounded by:*

$$F(q^\star(\tilde{p}_{\mathcal{D}_{fix}}))-F(q^\star)\leq C\|\tilde{p}_{\mathcal{D}_{fix}}-q^\star_{\mathcal{D}_{fix}}\|,$$

where

$$C=\bar{F}(q^\star)\exp(a_{\max}\|\tilde{p}_{\mathcal{D}_{fix}}-q^\star_{\mathcal{D}_{fix}}\|)\left(\frac{\bar{F}(q^\star(\tilde{p}_{\mathcal{D}_{fix}}))}{\mu}\cdot\|\alpha_{fix}+\alpha_{comp}\|\cdot\kappa(\alpha_{fix},\alpha_{comp})+\|\alpha_{fix}\|\right)$$

Proof. Recall our simplified notation $p_1=\tilde{p}_{\mathcal{D}_{\text{fix}}}$, $p_1'=q^\star_{\mathcal{D}_{\text{fix}}}$. Applying Lemma E.5 to Lemma E.6, we can bound the performance gap to be linear in $\|p_1-p_1'\|$:

$$F(r^\star(p_1),p_1)-F(r^\star(p_1'),p_1')\leq C\|\Delta\|$$

where C is a term that consists of three gradient norms:

$$C=\frac{1}{\mu}\sup_{t\in[0,1]}\big(\|\nabla^2_{r,p_1}F(r^\star(p_1'),p_1^t)\|\big)\|\nabla_r F(r^\star(p_1),p_1)\|+\sup_{t\in[0,1]}\|\nabla_{p_1}F(r^\star(p_1'),p_1^t)\|$$

and $p_1^t = tp_1' + (1-t)p_1$. We can apply the gradient norm bounds from Lemmas E.7, E.8, E.9:

$$C \leq \frac{1}{\mu}\exp(a_{\max}\|\Delta\|)\cdot\bar{F}(q^\star(p_1'))\cdot\left\|(\mathbf{1}+b_{\max}+\alpha_{\text{comp}})\odot\alpha_{\text{fix}}\right\|\cdot\bar{F}(q^\star(p_1))\left\||A_1p_1|+\alpha_{\text{comp}}\right\|$$
$$+\bar{F}(q^\star(p_1'))\left\|\exp(\alpha_{\text{fix}}\|\triangle\|)\odot\alpha_{\text{fix}}\right\|$$

We simplify this bound a bit further. First, note that the elements of $|A_1p_1|$ are $|\langle A_{i,1},p_1\rangle| \leq \|A_{i,1}\|\|p_1\| \leq \|A_{i,1}\|$. Therefore, $|A_1p_1| \preceq \alpha_{\text{fix}}$.

Second, note that $b_{i,\max} = \max\{|\langle A_{i,1},p_1\rangle|, |\langle A_{i,1},p_1'\rangle|\} \leq \|A_{i,1}\|$, so $b_{\max} \preceq \alpha_{\text{fix}}$ as well.

Lastly, $\left\|\exp(\alpha_{\text{fix}}\|\triangle\|)\odot\alpha_{\text{fix}}\right\|$ can be bounded by $\exp(a_{\max}\|\Delta\|)\cdot\left\|\alpha_{\text{fix}}\right\|$.

Then, we can bound C as

$$C \leq \bar{F}(q^\star)\exp(a_{\max}\|\Delta\|)\left(\frac{\bar{F}(q^\star(\tilde{p}_{\mathcal{D}_{\text{fix}}}))}{\mu}\|(\mathbf{1}+\alpha_{\text{fix}}+\alpha_{\text{comp}})\odot\alpha_{\text{fix}}\|\cdot\|\alpha_{\text{fix}}+\alpha_{\text{comp}}\|+\|\alpha_{\text{fix}}\|\right)$$

Replacing $\|(\mathbf{1}+\alpha_{\text{fix}}+\alpha_{\text{comp}})\odot\alpha_{\text{fix}}\|$ with $\kappa(\alpha_{\text{fix}},\alpha_{\text{comp}})$ completes our proof.

□

E.3 Proof for Theorem 4.2

When domains are being added and $\tilde{p}$ is the optimal mix on $\mathcal{D}$, $\tilde{p}$ is equal to $\tilde{p}_{\mathcal{D}_{\text{fix}}}$ is the solution to the following optimization problem:

$$\begin{aligned}&\text{minimize}_{p\in\triangle^{m-1}}\frac{1}{n}\sum_{i=1}^{m} f_i(\text{LM}(S,R,p))\\ &\text{s.t.}\quad p_j \leq \frac{kN_j}{R}\quad \forall j\in[m]\end{aligned} \tag{16}$$

Assumption E.11. We make the following assumptions beyond Assumption E.1.

1. We assume that $F(r,p_1)$ is μ_1-strongly convex in p_1.
2. mutual feasibility
3. Let $\mathcal{S}(r)$ be the feasible set of (2) defined by the repetition constraints on p_1:

   $$p_j \leq \frac{kN_j}{R\rho}\,\forall j\in\mathcal{D}_{\text{fix}}$$

   We assume **mutual feasibility** of $\tilde{p}_{\mathcal{D}_{\text{fix}}}$ and $q^\star_{\mathcal{D}_{\text{fix}}}$: that $\tilde{p}_{\mathcal{D}_{\text{fix}}} \in \mathcal{S}(r^\star)$, and $q^\star_{\mathcal{D}_{\text{fix}}} \in \mathcal{S}(r_0)$, where $r_0 = [1,\mathbf{0}]$ corresponds to the mix before new domains are added. That is, we assume that $\tilde{p}_j \leq \frac{kN_j}{R\rho^\star}$ and $q_j^\star \leq \frac{kN_j}{R}$ for $j\in\mathcal{D}_{\text{fix}}$, meaning that the existing mix cannot have active repetition constraints.

First, we present a general lemma that allows us to bound any $\|p_1 - p_1'\|$.

Lemma E.12 (Argmin stability with mutual feasibility)**.** *Let $\mathcal{S},\mathcal{S}' \subset \triangle^{\|D_1\|-1}$ be convex sets. Let $h : \mathcal{S}\to\mathbb{R}$ be differentiable and μ_1-strongly convex on $\mathcal{S}$. Let $h' : \mathcal{S}'\to\mathbb{R}$ be a differentiable function. Define*

$$p_1 = \arg\min_{p_1\in\mathcal{S}} h(p_1), \qquad p_1' = \arg\min_{p_1\in\mathcal{S}'} h'(p_1)$$

We assume mutual feasibility; that is, $p_1\in\mathcal{S}'$ and $p_1'\in\mathcal{S}$. Then,

$$\|p_1 - p_1'\| \leq \frac{1}{\mu_1}\|\nabla h(p_1') - \nabla h'(p_1')\|.$$

Proof. Applying our mutual feasibility assumption, the first order condition at p_1 for h and p_1' for h yields

$$\langle \nabla h(p_1), p_1' - p_1 \rangle \geq 0$$
$$\langle \nabla h'(p_1'), p_1 - p_1' \rangle \geq 0$$

Adding these together and adding and subtracting $\nabla h(p_1')$, we have

$$\begin{aligned}
&\langle \nabla h(p_1) - \nabla h'(p_1'), p_1' - p_1 \rangle \geq 0 \\
\Rightarrow &\langle \nabla h(p_1) - \nabla h(p_1') + \nabla h(p_1') - \nabla h'(p_1'), p_1' - p_1 \rangle \geq 0 \\
\Rightarrow &\langle \nabla h(p_1) - \nabla h(p_1'), p_1 - p_1' \rangle \leq \langle \nabla h(p_1') - \nabla h'(p_1'), p_1' - p_1 \rangle
\end{aligned}$$

Next, due to μ_1 strong convexity of h, we have

$$h(p_1) \geq h(p_1') + \langle \nabla h(p_1'), p_1 - p_1' \rangle + \frac{\mu_1}{2} \|p_1 - p_1'\|^2$$
$$h(p_1') \geq h(p_1) + \langle \nabla h(p_1), p_1' - p_1 \rangle + \frac{\mu_1}{2} \|p_1 - p_1'\|^2$$

Adding these together and applying the bound from the first order conditions, we get

$$\begin{aligned}
\mu \|p_1 - p_1'\|^2 &\leq \langle \nabla h(p_1) - \nabla h(p_1'), p_1 - p_1' \rangle \\
&\leq \langle \nabla h(p_1') - \nabla h'(p_1'), p_1' - p_1 \rangle \\
&\leq \|\nabla h(p_1') - \nabla h'(p_1')\| \cdot \|p_1' - p_1\|
\end{aligned}$$

Dividing both sides by $\mu \|p_1' - p_1\|$ completes our proof. ☐

Lemma E.13 (When $\tilde{p}_{\mathcal{D}_{\text{fix}}} = q^\star_{\mathcal{D}_{\text{fix}}}$ for adding)**.** *Consider the* adding *update, where $\mathcal{D}_2 = \emptyset$ and $\mathcal{D}' = [\mathcal{D}_1 \mathcal{D}_2']$ with $\mathcal{D}_2' \neq \emptyset$. Let $\tilde{p} \in \triangle^{m-1}$ be an optimal solution to the mixing problem on the pre-update domain set $\mathcal{D} = \mathcal{D}_1$ (equivalently, $\tilde{p}_{\mathcal{D}_1} = \tilde{p}$), and let $q^\star$ be the optimal solution to the post-update mixing problem on $\mathcal{D}'$. If $\rho^\star = 1$, then*

$$\tilde{p}_{\mathcal{D}_{\mathit{fix}}} = q^\star_{\mathcal{D}_{\mathit{fix}}}.$$

Proof. Since $\mathcal{D}_2 = \emptyset$, the pre-update problem over $\mathcal{D} = \mathcal{D}_1$ is equivalent to the post-update problem restricted to mixtures that assign zero mass to all added domains:

$$\begin{aligned}
&\text{minimize}_{q \in \triangle^{m'-1}} \frac{1}{n} \sum_{i=1}^{n} f_i(\text{LM}(S, R, q)) \\
&\text{s.t.} \quad q_j \leq \frac{kN_j'}{R} \quad \forall j \in \mathcal{D}_1 \cup \mathcal{D}_2' \\
&\qquad\;\; q_j = 0 \quad \forall j \in \mathcal{D}_2'.
\end{aligned}$$

If $\rho^\star = 1$, then $q_j^\star = 0$ for all $j \in \mathcal{D}_2'$. This means that the additional constraints $q_j = 0$ on the post-update problem do not change the optimal value, since the restricted feasible set still contains the optimal solution of $q_j^\star = 0$. Therefore, if $\rho^\star = 1$, the restricted problem above and the unrestricted post-update problem have the same optimum, and their optimizers coincide on $\mathcal{D}_1$; in particular, $q^\star_{\mathcal{D}_1} = \tilde{p}_{\mathcal{D}_1}$. Replacing $\mathcal{D}_1$ with $\mathcal{D}_{\text{fix}}$ completes our proof. ☐

Lemma E.14. *Assume mutual feasibility between p_1 and p_1'. Then, when the domain update involves adding domains,*

$$\|p_1 - p_1'\| \leq \frac{2(1-\rho^\star)}{\mu_1} \| \sup_{t \in [0,1]} \|\nabla^2_{p_1, r} F(r^t, p_1')\| \qquad r^t = tr_0 + (1-t) r^\star$$

Proof. Define $r_0 = [1,\mathbf{0}]$ where $\rho_0 = 1$. Then, $p_1 = \arg\min_{p_1 \in \mathcal{S}(r_0)} F(r_0,p_1)$. We can write p_1' similarly as $p_1' = \operatorname{argmin}_{p_1 \in \mathcal{S}(r^\star)} F(r^\star,p_1)$.

If we assume mutual feasibility and define $h(p_1) = F(r_0,p_1), h'(p_1) = F(r^\star,p_1)$, then applying Lemma E.12 gives us

$$\|p_1 - p_1'\| \le \frac{1}{\mu_1} \|\nabla_{p_1} F(r_0,p_1') - F(r^\star,p_1')\|$$

Define $r^t = tr_0 + (1-t)r^\star$, where $t \in [0,1]$. By the fundamental theorem of calculus, it holds that

$$\nabla_{p_1} F(r_0,p_1') - \nabla_{p_1} F(r^\star,p_1') = \int_0^1 \nabla^2_{p_1,r} F(r^t,p_1')(r_0 - r^\star)dt$$

Therefore,

$$\|p_1 - p_1'\| \le \frac{1}{\mu_1} \|r_0 - r^\star\| \sup_{t \in [0,1]} \|\nabla^2_{p_1,r} F(r^t,p_1')\|$$

Lastly, we bound $\|r_0 - r^\star\|$:

$$\|r_0 - r^\star\| = \|[1,\mathbf{0}] - [\rho^\star,(1-\rho^\star)q^\star_{\mathcal{D}_2'}]\| = \|(1-\rho^\star)[1,-q^\star_{\mathcal{D}_2'}]\| \le 2(1-\rho^\star)$$

This completes our proof. □

Lemma E.15. *Define $r^t = tr_0 + (1-t)r^\star(p_1')$, where $r_0 = [1,\mathbf{0}]$. The gradient norm $\|\nabla^2_{r,p_1} F(r^t,p_1')\|$ can be bounded by*

$$\|\nabla^2_{r,p_1} F(r^t,p_1')\| \le \bar{F}(q^\star)\exp(c_{\max}(1-\rho^\star))\|(1+\alpha_{\mathit{fix}}+\alpha_{\mathit{comp}}) \odot \alpha_{\mathit{fix}}\|$$

where $c_{\max} = max_i\{\|A_{i,1}\| + \|A_{i,2}\|\}$ and $\odot$ is the Hadamard product.

Proof. First, we can write r^t as

$$\begin{aligned} r^t &= t[1,\mathbf{0}] + (1-t)[\rho^\star(p_1'),(1-\rho^\star(p_1'))p_2^\star(p_1')] \\ &= [t + (1-t)\rho^\star(p_1'),(1-t)(1-\rho^\star(p_1'))p_2^\star(p_1')] \\ &= [t + (1-t)\rho^\star(p_1'),(1-(t+(1-t)\rho^\star(p_1')))p_2^\star(p_1')] \end{aligned}$$

Therefore, we can express $r^t = [\rho^t,(1-\rho^t)p_2^t]$ where $\rho^t := t + (1-t)\rho^\star(p_1')$, and $p_2^t = p_2^\star(p_1')$.

Applying Lemma E.9, we can write the gradient as

$$\nabla^2_{r,p_1} F(r^t,p_1') = \frac{1}{n}\sum_{i=1}^n \exp(A_{i,1}^\top \rho^t p_1' + A_{i,2}^\top (1-\rho^t)p_2^t)(\rho^t a_i + e_1)A_{i,1}^\top$$

where $a_i = \begin{bmatrix} \langle A_{i,1},p_1' \rangle \\ A_{i,2} \end{bmatrix} \in \mathbb{R}^{|\mathcal{D}_2'|+1}$ and e_1 is the standard basis vector $\begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \in \mathbb{R}^{|\mathcal{D}_2'|+1}$. Then, using Cauchy-Schwarz inequality, the gradient norm can be bounded as

$$\|\nabla^2_{r,p_1} F(r^t,p_1')\| \le \frac{1}{n}\sum_{i=1}^n \exp(A_{i,1}^\top \rho^t p_1' + A_{i,2}^\top (1-\rho^t)p_2^t)\|(\rho^t a_i + e_1)A_{i,1}^\top\| \tag{17}$$

$$\le \frac{1}{n}\sqrt{\sum_{i=1}^n \exp(2(A_{i,1}^\top \rho^t p_1' + A_{i,2}^\top (1-\rho^t)p_2^t))}\sqrt{\sum_{i=1}^n \|(\rho^t a_i + e_1)A_{i,1}^\top\|^2}$$

We bound the first summation of (17) by adding and subtracting $\rho^\star(p_1')$:

$$\begin{aligned}
&\sum_{i=1}^{n}\exp(2(A_{i,1}^\top\rho^t p_1'+A_{i,2}^\top(1-\rho^t)p_2^t))\\
&\quad=\sum_{i=1}^{n}\exp(2(A_{i,1}^\top(\rho^t+\rho^\star(p_1')-\rho^\star(p_1'))p_1'+A_{i,2}^\top((1-\rho^t)+(1-\rho^\star(p_1'))-(1-\rho^\star(p_1')))p_2^t))\\
&\quad=\sum_{i=1}^{n}\exp(2(A_{i,1}^\top\rho^\star(p_1')p_1'+A_{i,2}^\top(1-\rho^\star(p_1'))p_2^t))\exp(2(A_{i,1}^\top(\rho^t-\rho^\star(p_1'))p_1'+A_{i,2}^\top(\rho^\star(p_1')-\rho^t)p_2^\star(p_1')))\\
&\quad=\sum_{i=1}^{n}\exp(2(A_{i,1}^\top\rho^\star(p_1')p_1'+A_{i,2}^\top(1-\rho^\star(p_1'))p_2^t))\exp(2t(1-\rho^\star(p_1'))(A_{i,1}^\top p_1'-A_{i,2}^\top p_2^\star(p_1')))
\end{aligned}$$

We can then replace some terms in (17) with $\bar F(r^\star(p_1'),p_1')$

$$\begin{aligned}
&\frac{1}{n}\sqrt{\sum_{i=1}^{n}\exp(2(A_{i,1}^\top\rho^t p_1'+A_{i,2}^\top(1-\rho^t)p_2^t))}\\
&\quad\le\frac{1}{n}\sum_{i=1}^{n}\exp(A_{i,1}^\top\rho^\star(p_1')p_1'+A_{i,2}^\top(1-\rho^\star(p_1'))p_2^t)\cdot\max_i\{\exp(t(1-\rho^\star(p_1'))(A_{i,1}^\top p_1'-A_{i,2}^\top p_2^\star(p_1')))\}\\
&\quad=\bar F(r^\star(p_1'),p_1')\max_i\{\exp(t(1-\rho^\star(p_1'))(A_{i,1}^\top p_1'-A_{i,2}^\top p_2^\star(p_1')))\}
\end{aligned}$$

We can bound $A_{i,1}^\top p_1' - A_{i,2}^\top p_2^\star(p_1') \le \|A_{i,1}\| + \|A_{i,2}\|$ using Cauchy-Schwarz. Let $c_{\max}=\max_i\{\|A_{i,1}\|+\|A_{i,2}\|\}$. Using the fact that $t\in[0,1]$, the first part of (17) can be bounded by

$$\frac{1}{n}\sqrt{\sum_{i=1}^{n}\exp(2(A_{i,1}^\top\rho^t p_1'+A_{i,2}^\top(1-\rho^t)p_2^t))}\le\bar F(r^\star(p_1'),p_1')\exp(c_{\max}(1-\rho^\star(p_1')))$$

The second summation of (17) can be bounded in the exact same way as the second summation of (15) in Lemma E.9. Therefore, we have that

$$\sqrt{\sum_{i=1}^{n}\|(\rho^t a_i+e_1)A_{i,1}^\top\|^2}\le\|(1+\alpha_{\text{fix}}+\alpha_{\text{comp}})\odot\alpha_{\text{fix}}\|$$

This completes our proof. □

E.3.1 THEOREM 4.2 (FORMAL)

Theorem E.16. *Define $\tilde p$ is the solution to* (2) *on $\mathcal{D}$ and suppose that new domains are added. Let $c_{\max}=\max_i\{\|A_{i,1}\|+\|A_{i,2}\|\}$. Then, the difference between $\tilde p_{\mathcal{D}_{fix}}$ and $q^\star_{\mathcal{D}_{fix}}$ is bounded by*

$$\|\tilde p_{\mathcal{D}_{fix}}-q^\star_{\mathcal{D}_{fix}}\|\le\frac{2\bar F(q^\star)\exp(c_{\max}(1-\rho^\star))}{\mu_1}\cdot\kappa(\alpha_{fix},\alpha_{comp})\cdot(1-\rho^\star)$$

Proof. Applying Lemma E.15 and Lemma E.14, we have

$$\begin{aligned}
\|p_1-p_1'\|&\le\frac{2(1-\rho^\star)}{\mu_1}\bar F(q^\star)\exp(c_{\max}(1-\rho^\star))\|(1+\alpha_{\text{fix}}+\alpha_{\text{comp}})\odot\alpha_{\text{fix}}\|\\
&=\frac{2\bar F(q^\star)\exp(c_{\max}(1-\rho^\star))}{\mu_1}\cdot\kappa(\alpha_{\text{fix}},\alpha_{\text{comp}})\cdot(1-\rho^\star)
\end{aligned}$$

□

E.4 Additional Theoretical Results

E.4.1 Removing domains

Our results for are symmetric to the addition update, but we state the full derivation for completeness.

Lemma E.17 (When $\tilde{p}_{\mathcal{D}_{\text{fix}}} = q^\star_{\mathcal{D}_{\text{fix}}}$ for removing)**.** *Consider the* removing *update, where $\mathcal{D}_2 \neq \emptyset$ and $\mathcal{D}' = \mathcal{D}_1$ (equivalently, $\mathcal{D}'_2 = \emptyset$). Let $\tilde{p} \in \triangle^{m-1}$ be an optimal solution to the mixing problem on the pre-update domain set $\mathcal{D} = [\mathcal{D}_1, \mathcal{D}_2]$, and let $q^\star$ be the optimal solution to the post-update mixing problem on $\mathcal{D}' = \mathcal{D}_1$ (equivalently, $q^\star_{\mathcal{D}_1} = q^\star$). If $\pi^\star = 1$, then*

$$\tilde{p}_{\mathcal{D}_{\mathit{fix}}} = q^\star_{\mathcal{D}_{\mathit{fix}}}.$$

Proof. Since $\mathcal{D}'_2 = \emptyset$, the post-update problem over $\mathcal{D}' = \mathcal{D}_1$ is equivalent to the pre-update problem restricted to mixtures that assign zero mass to all removed domains:

$$\begin{aligned}
&\text{minimize}_{p \in \triangle^{m-1}} \frac{1}{n} \sum_{i=1}^{n} f_i(\text{LM}(S, R, p)) \\
&\text{s.t.} \quad p_j \leq \frac{kN_j}{R} \quad \forall j \in \mathcal{D}_1 \cup \mathcal{D}_2 \\
&\qquad\; p_j = 0 \quad \forall j \in \mathcal{D}'_2.
\end{aligned}$$

If $\pi^\star = 1$, then $\tilde{p}_j = 0$ for all $j \in \mathcal{D}_2$. This means that the additional constraints $p_j = 0$ on the pre-update problem do not change the optimal value, since the restricted feasible set still contains the optimal solution of $\tilde{p}_j = 0$. Therefore, if $\pi^\star = 1$, the restricted problem above and the unrestricted pre-update problem have the same optimum, and their optimizers coincide on $\mathcal{D}_1$; in particular, $q^\star_{\mathcal{D}_1} = \tilde{p}_{\mathcal{D}_1}$. Replacing $\mathcal{D}_1$ with $\mathcal{D}_{\text{fix}}$ completes our proof. ☐

E.4.2 Partitioning a domain

Lemma E.18 (When $\tilde{p}_{\mathcal{D}_{\text{fix}}} = q^\star_{\mathcal{D}_{\text{fix}}}$ for partitioning)**.** *Consider the* partitioning *update operator, where $\mathcal{D} \setminus \mathcal{D}_1 = \{D\}$ consists of a single domain and $\mathcal{D}'_2 = \{D'_1, \ldots, D'_\ell\}$ are subdomains with $D = \bigcup_{j=1}^{\ell} D'_j$. Define the natural distribution $p_{\text{nat}} \in \triangle^{\ell-1}$ by*

$$[p_{\text{nat}}]_j = \frac{N'_j}{\sum_{t=1}^{\ell} N'_t} \qquad \forall j \in [\ell].$$

If $q^\star_{\mathcal{D}'_2} = p_{\text{nat}}$, then

$$\tilde{p}_{\mathcal{D}_{\mathit{fix}}} = q^\star_{\mathcal{D}_{\mathit{fix}}}.$$

Proof. Since $\mathcal{D}_2 = \{D\}$ is a single domain, sampling $(1-\pi)R$ tokens from D induces a distribution over the subdomains $\{D'_1, \ldots, D'_\ell\}$ that is proportional to token counts, i.e., p_{nat}. Therefore, the pre-update optimization problem can be written as the post-update optimization problem restricted to mixtures that preserve this conditional distribution within the partition:

$$\begin{aligned}
&\text{minimize}_{q \in \triangle^{m'-1}} \frac{1}{n} \sum_{i=1}^{n} f_i(\text{LM}(S, R, q)) \\
&\text{s.t.} \quad q_j \leq \frac{kN'_j}{R} \quad \forall j \in \mathcal{D}_1 \cup \mathcal{D}'_2 \\
&\qquad\; q_{\mathcal{D}'_2} = p_{\text{nat}},
\end{aligned}$$

If $q^\star_{\mathcal{D}'_2} = p_{\text{nat}}$, then $q^\star$ lies in the restricted feasible set above. The additional constraint $q_{\mathcal{D}'_2} = p_{\text{nat}}$ does not change the optimal value of the post-update problem, since the restricted feasible set still contains the optimal solution. Therefore, the optimizer on the unrestricted post-update problem must coincide with the optimizer of the restricted problem on the unaffected domains, implying $q^\star_{\mathcal{D}_1} = \tilde{p}_{\mathcal{D}_1}$. Replacing $\mathcal{D}_1$ with $\mathcal{D}_{\text{fix}}$ completes our proof. ☐

E.4.3 REVISING A DOMAIN

Lemma E.19 (Exactness for revising). *Consider the* revising *update operator, where $\mathcal{D}_2=\{D\}$ and $\mathcal{D}_2'=\{D'\}$ consists of a single domain whose contents are modified. Assume the log-linear model holds both pre- and post-update:*

$$F^{\text{old}}(q)=\frac{1}{n}\sum_{i=1}^{n}\exp\left((A_{i,1})^\top q_{\mathcal{D}_{\textit{fix}}}+(A_{i,2}^{\text{old}})^\top q_{\mathcal{D}_2}\right),\qquad F^{\text{new}}(q)=\frac{1}{n}\sum_{i=1}^{n}\exp\left((A_{i,1})^\top q_{\mathcal{D}_{\textit{fix}}}+(A_{i,2}^{\text{new}})^\top q_{\mathcal{D}_2}\right),$$

where $\mathcal{D}=[\mathcal{D}_1,\mathcal{D}_2]$ and $\mathcal{D}'=[\mathcal{D}_1,\mathcal{D}_2']$ have the same dimensionality, and the repetition constraints are unchanged.

Let $\tilde{p}$ be an optimal solution to the pre-update mixing problem with objective F^{old}, and let $q^\star$ be an optimal solution to the post-update mixing problem with objective F^{new}. If $A_{i,2}^{\text{new}}=A_{i,2}^{\text{old}}$ for all $i\in[n]$, then $\tilde{p}=q^\star$, and in particular $\tilde{p}_{\mathcal{D}_{\textit{fix}}}=q^\star_{\mathcal{D}_{\textit{fix}}}$.

Proof. If $A_{i,2}^{\text{new}}=A_{i,2}^{\text{old}}$ for all $i\in[n]$, then $F^{\text{new}}(q)=F^{\text{old}}(q)$ for all q. Since the feasible set (simplex and repetition constraints) is unchanged under revising, the pre-update and post-update optimization problems are identical. Therefore their optimizers coincide, so $\tilde{p}=q^\star$. □

F EMPIRICAL VALIDATION OF THEOREMS 4.1 AND 4.2

We empirically study the performance gap of FULLMIXTUREREUSE and PARTIALMIXTUREREUSE, where we measure the terms in Theorems 4.1 and 4.2 and see if they track how close mixture reuse's performance is to full recomputation.

F.1 SETUP

We consider two examples of an `Add` domain update:

- $\mathcal{D}_1$ =DCLM (Li et al., 2024) partitioned into 24 topic-based domains using WebOrganizer (Wettig et al., 2025), $\mathcal{D}_2'$ = Stack-Edu (Allal et al., 2025) partitioned into 15 programming languages.
- $\mathcal{D}_1$ =DCLM partitioned into 24 topic-based domains using WebOrganizer, $\mathcal{D}_2'$ = olmOCR Science PDFs (Olmo et al., 2025) partitioned into 21 topic-based domains using WebOrganizer.

F.2 RESULTS

Performance is correlated with the reuse gap. Mixture reuse yields $q^\star(\tilde{p}_{\mathcal{D}_{\text{fix}}})$ from Algorithm 2 given reused ratios $\tilde{p}_{\mathcal{D}_{\text{fix}}}$. Full recomputation yields $q^\star$ from directly solving (2) for the optimal mix on $\mathcal{D}'$. Theorem 4.1 shows that the performance gap of mixture reuse, $F(q^\star(\tilde{p}_{\mathcal{D}_{\text{fix}}}))-F(q^\star)$, where F is the average downstream task BPB, is bounded in terms of the reuse gap, $\|\tilde{p}_{\mathcal{D}_{\text{fix}}}-q^\star_{\mathcal{D}_{\text{fix}}}\|$, where $q^\star_{\mathcal{D}_{\text{fix}}}$ is $q^\star$ restricted to $\mathcal{D}_{\text{fix}}$ and normalized. We validate that the reuse gap predicts the performance gap.

To do this, we construct different values of $\tilde{p}_{\mathcal{D}_{\text{fix}}}$ with varying reuse gaps and measure the resulting performance gaps. We first approximate $q^\star_{\mathcal{D}_{\text{fix}}}$ and $F(q^\star)$ by running full recomputation on $\mathcal{D}'$ using OLMIXBASE and normalizing the resulting mix over $\mathcal{D}_{\text{fix}}$. We then construct two $\tilde{p}_{\mathcal{D}_{\text{fix}}}$:

- Weak mix: to construct a $\tilde{p}_{\mathcal{D}_{\text{fix}}}$ far from $q^\star_{\mathcal{D}_{\text{fix}}}$, we run OLMIXBASE on $\mathcal{D}'$ with a modified objective that maximizes BPB and then normalize the resulting mix over $\mathcal{D}_{\text{fix}}$.
- Intermediate mix: we average the weak mix with $q^\star_{\mathcal{D}_{\text{fix}}}$.

Figure 15 confirms the findings of Theorem 4.1: as the reuse gap increases from intermediate to weak, the performance gap increases across both settings.

The reuse gap is controlled by how much the domain update shifts the optimal mixture. When adding domains, Theorem 4.2 shows that the reuse gap (and consequently the performance gap) depends on how much total weight shifts to the affected domains. Specifically, the reuse gap depends

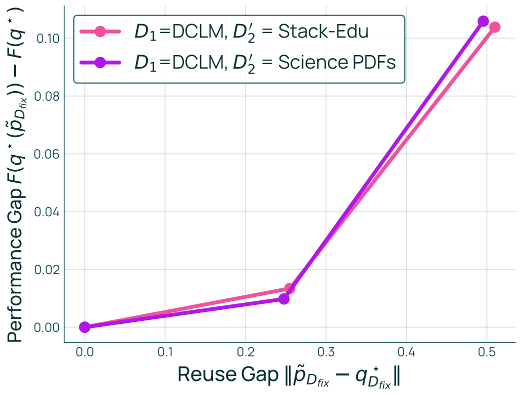

Figure 15: Performance vs Reuse Gap. The optimality of $\tilde{p}_{\mathcal{D}_{\text{fix}}}$ after the domain update is correlated with the success of mixture reuse. This holds true for both settings we consider, where either Stack-Edu or olmOCR Science PDF data is added to DCLM.

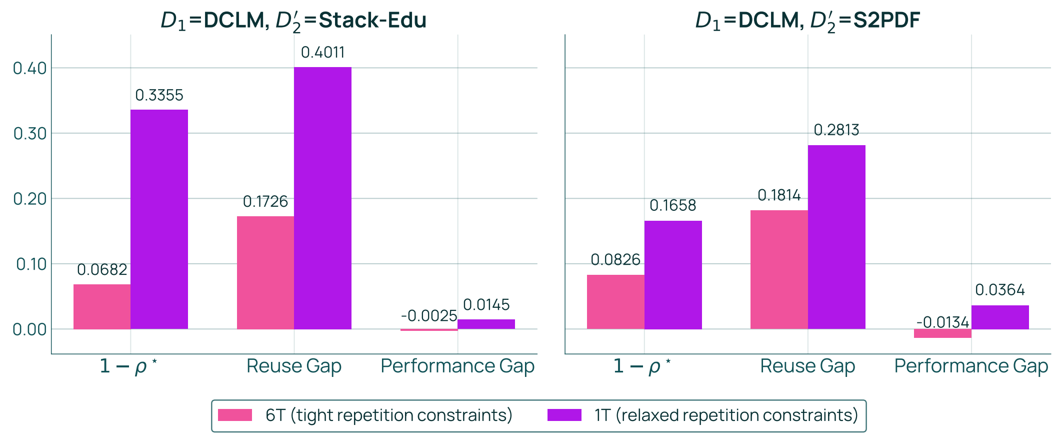

Figure 16: Performance gap vs $1-\rho^\star$ when adding Stack-Edu to DCLM (left) and olmOCR Science PDFs to DCLM (right). Varying R changes $\rho^\star$, and we observe that $1-\rho^\star$ propagates to both the reuse gap and performance gap, validating Theorem 4.2.

on $1-\rho^\star$, where $\rho^\star$ is the total weight on the reused domains $\mathcal{D}_1$ in the optimal mix $q^\star$. We validate that $1-\rho^\star$ predicts both the reuse gap and performance gap.

To do this, we construct settings with different $\rho^\star$ by varying the repetition constraint. A tight constraint (small k or large R in (2)) forces the mix to stay close to the natural distribution, yielding $\rho^\star \approx 1$ when $\mathcal{D}_1$ is much larger than the added domains. A relaxed constraint allows more weight to shift to new domains, potentially yielding smaller $\rho^\star$. We test two settings: $R = 1$T (relaxed) and $R = 6$T (tight). For each R, we 1) compute $\tilde{p}_{\mathcal{D}_{\text{fix}}}$ by running OLMIXBASE on $\mathcal{D}$; 2) apply FULLMIXTUREREUSE on $\mathcal{D}'$ using $\tilde{p}_{\mathcal{D}_{\text{fix}}}$; and 3) use full recomputation (run OLMIXBASE on $\mathcal{D}'$) to get an approximate $\rho^\star$, the reuse gap, and the performance gap.

Figure 16 shows that larger $1-\rho^\star$ correlates with both a larger reuse gap and a larger performance gap across both settings, validating that the extent to which the domain update shifts the optimal mixture directly impacts mixture reuse performance.

PARTIALMIXTUREREUSE reduces coupling and improves performance. Theorems 4.1 and 4.2 show that the coupling term κ controls the *rate* at which changes in optimality translate to the performance gap. The coupling term is large when both reused domains $\mathcal{D}_{\text{fix}}$ and recomputed domains $\mathcal{D}_{\text{comp}}$ impact similar downstream tasks. We validate that reducing coupling—by adjusting $\mathcal{D}_{\text{fix}}$ vs $\mathcal{D}_{\text{comp}}$ via PARTIALMIXTUREREUSE—improves both the reuse gap and performance gap.

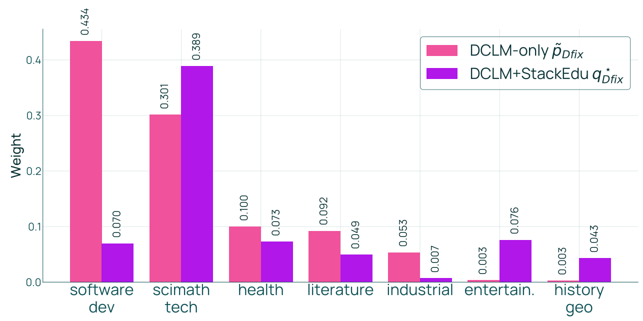

Figure 17: Comparison of $\tilde{p}_{\mathcal{D}_{\text{fix}}}$ (DCLM-only) versus $q^\star_{\mathcal{D}_{\text{fix}}}$ (DCLM+Stack-Edu) (top-7 domains according to $\tilde{p}_{\mathcal{D}_{\text{fix}}}$. The only domain that significantly differs in mixture weight is software development, suggesting that PARTIALMIXTUREREUSE can reduce coupling.

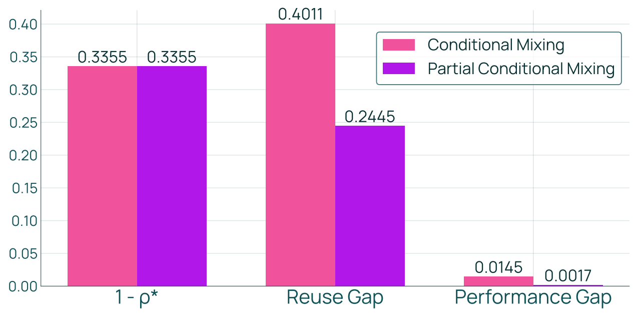

Figure 18: Performance of PARTIALMIXTUREREUSE (recompute DCLM:software_development) for $\mathcal{D}_1$=DCLM, $\mathcal{D}'_2$=Stack-Edu. Recomputing DCLM:software development along with Stack-Edu reduces both the reuse gap and the performance gap.

To do this, we analyze a scenario where coupling is high: adding Stack-Edu to DCLM topics when $R = 1$T (from Figure 16 left). Figure 17 compares the reused ratios $\tilde{p}_{\mathcal{D}_{\text{fix}}}$ against the optimal ratios $q^\star_{\mathcal{D}_{\text{fix}}}$ over DCLM topics. There is a stark difference in the weight on DCLM's software development topic, suggesting high coupling between software development and Stack-Edu. Based on this, we test whether PARTIALMIXTUREREUSE with $\mathcal{D}_{\text{comp}} = \text{Stack-Edu} \cup \{\text{DCLM:software\_development}\}$ reduces coupling and improves performance compared to FULLMIXTUREREUSE (which uses $\mathcal{D}_{\text{comp}} = \text{Stack-Edu}$).

Figure 18 shows that recomputing DCLM:software_development along with Stack-Edu reduces both the reuse gap and performance gap, even though $1 - \rho^\star$ remains the same—indicating that the coupling term has changed. To confirm this directly, we compute κ (see Appendix E.1 for definition) for FULLMIXTUREREUSE and κ for PARTIALMIXTUREREUSE with $\mathcal{D}_{\text{comp}} = \text{Stack-Edu} \cup \{\text{DCLM:topic}\}$ for each DCLM topic. Figure 19 shows that recomputing DCLM:software_development reduces κ far more than recomputing any other DCLM topic, validating that PARTIALMIXTUREREUSE reduces coupling and that this coupling reduction translates to better performance.

G EXPERIMENT DETAILS

Domains We provide a breakdown of the number of tokens per each domain in Table 8 and a description of each domain update in our simulated LM development workflow in Table 9.

Model We train 1B parameter decoder-only transformer models using Olmo 2 architecture (OLMo et al., 2024). In particular, we set n_layers=16, n_head=16, d_model=2048, head_dim=128.

We use a batch size of 512, a learning rate of 0.0018 with a cosine scheduler with warmup and linear decay, and a sequence length of 4096. We use the Dolma 2 tokenizer.

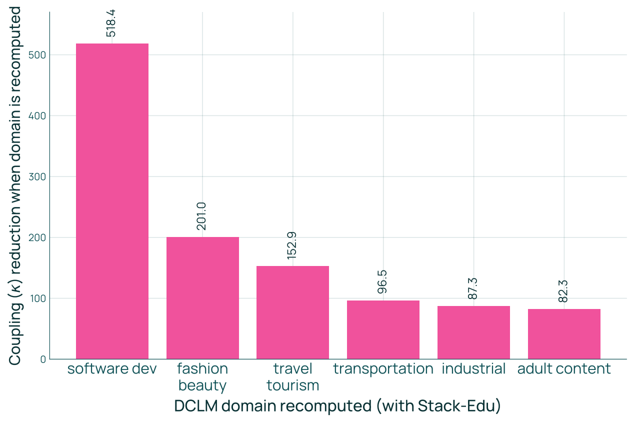

Figure 19: The top-6 DCLM domains for which recomputation results in the greatest reduction in κ for $\mathcal{D}$=DCLM, $\mathcal{D}'$=DCLM+Stack-Edu. The reduction from recomputing DCLM:software development via PARTIALMIXTUREREUSE is substantially higher than recomputing other DCLM topics.

All 1B models were trained on 32 NVIDIA H100s (80 GB), while proxy models were trained on one NVIDIA H100.

Evaluation Table 10 lists all the downstream tasks.

G.1 MIXING DETAILS

Across all swarm-based methods we evaluate on (OLMIXBASE, Swarm Reuse, FULLMIXTUREREUSE, PARTIALMIXTUREREUSE), we use $S_{\text{small}} = 30M$ parameters (config in Table 5) and train for 5x Chinchilla, with a batch size of 64, a learning rate of 0.007, and a sequence length of 2048.

For any m domains we want to recompute, we set K to be $c(m+1)$, rounded to the nearest power of 2 for $c = 1,2,3$. We solve all of our mixture optimization problems using CVXPY and a KL regularization term with $\lambda = 0.05$. We set $k = 4$ and $R = 1T$ for the experiments in §5.

For PARTIALMIXTUREREUSE, we freeze the DCLM topic ratios and the Stack-Edu topic ratios, while recomputing on source-level ratios after each domain update. When Stack-Edu is added, we also recompute on DCLM:software development, which was shown to be effective in Appendix F.2.

G.2 ADDITIONAL RESULTS

In Figure 20, we compare the performance versus cost of mixing strategies when $R = 6$T, a more data-constrained setting where the feasible set of mixes is smaller. For this setting, each swarm is of size $K \approx c(m+1)$ for $c = 1,2,3$ and we sweep over c for all strategies excluding the natural distribution. We exclude PARTIALMIXTUREREUSE since, only at $R = 1$T did our results in Appendix F show that FULLMIXTUREREUSE was insufficient.

At $c = 3$, we find that FULLMIXTUREREUSE achieves 99% of full recomputation's performance improvement (+6.94% versus +6.97%) while using 74% fewer total proxy runs (216 versus 832 runs). Swarm reuse achieves 93% of full recomputation's performance with 52 more runs than FULLMIXTUREREUSE. Similar trends hold for smaller c.

Domain	Token Count	Domain	Token Count
DCLM		**Stack-Edu**	
Adult Content	67,760,078,203	C	4,735,074,247
Art and Design	70,659,711,995	C#	7,204,525,343
Crime and Law	170,130,914,779	C++	12,530,445,761
Education and Jobs	184,690,792,861	Go	1,398,595,118
Electronics and Hardware	80,168,541,745	Java	31,347,725,888
Entertainment	441,768,061,760	JavaScript	8,886,972,357
Fashion and Beauty	37,256,539,512	Markdown	28,916,320,218
Finance and Business	310,313,927,581	PHP	7,395,033,318
Food and Dining	105,937,299,687	Python	18,017,832,560
Games	229,992,491,702	Ruby	1,386,775,805
Health	393,496,227,836	Rust	1,418,447,132
History and Geography	161,049,719,459	SQL	7,063,472,860
Home and Hobbies	126,910,777,314	Shell	2,542,637,875
Industrial	43,572,140,450	Swift	1,510,019,025
Literature	364,834,344,848	TypeScript	2,495,753,789
Politics	611,198,130,192		
Religion	277,776,929,208	**olmOCR Science PDFs**	
Science, Math and Technology	427,131,054,341	Adult	303,073,226
Social Life	218,731,841,124	Art and Design	6,833,185,034
Software	108,039,380,021	Crime and Law	42,538,674,743
Software Development	223,384,974,282	Education and Jobs	138,127,926,093
Sports and Fitness	196,759,999,355	Entertainment	6,069,602,783
Transportation	90,793,306,202	Fashion and Beauty	557,917,820
Travel and Tourism	57,642,815,530	Finance and Business	61,150,044,703
		Food and Dining	2,322,982,204
Single-Source Domains		Games	2,486,095,532
AlgebraicStack	11,818,955,329	Health	108,215,933,374
ArXiv	20,773,846,846	Home and Hobbies	3,924,579,643
FineMath-3+	34,057,973,953	Industrial	29,389,278,657
Pes2o	58,552,461,187	Literature	31,886,391,090
Wikipedia	10,067,758,073	Politics	39,234,116,889
		Religion	24,729,732,953
		Science and Technology	424,245,385,160
		Software	9,146,853,216
		Software Development	41,841,278,724
		Sports and Fitness	5,450,913,796
		Transportation	17,149,342,957
		Travel	2,102,425,717

Table 8: Token counts for all domains in the final domain set. Sources with topic-level partitions (DCLM, Stack-Edu, olmOCR Science PDFs) are shown with indented topics. Single-source domains are shown without subdivision.

Operation	Dataset(s)	Δm
Initial	DCLM (Li et al., 2024) partitioned into topical domains using WebOrganizer (Wettig et al., 2025))	24
`Add`	Stack-Edu (Allal et al., 2025), partitioned by programming language	+15
`Add`	ArXiv (Azerbayev et al., 2023), FineMath 3+ (Allal et al., 2025), olmOCR Science PDFs (Olmo et al., 2025), Dolma 1 Wikipedia (Soldaini et al., 2024), AlgebraicStack (Azerbayev et al., 2023), pes2o (Soldaini & Lo, 2023)	+6
`Revise`	olmOCR Science PDFs (Olmo et al., 2025) (reformatted tables and references)	0
`Remove`	AlgebraicStack (Azerbayev et al., 2023)	-1
`Partition`	olmOCR Science PDFs (Olmo et al., 2025), partitioned into topical domains using WebOrganizer (Wettig et al., 2025)	+20

Table 9: Full specification of datasets used for simulated domain updates, corresponding to Table 1 in the main text.

	Task	Capability	# ICL	Metric	# Subtasks
Math	Minerva MATH (2022)	Math Gen	4^{α}	BPB	7
	Subtasks: *Algebra, Counting and Probability, Geometry, Intermediate Algebra Number Theory, Prealgebra, Precalculus*				
Code	HumanEval (2021)	Code Gen	3	BPB	-
	MBPP (2021)	Code Gen	3	BPB	-
	MT MBPP (2022; 2025)	Code Gen	3	BPB	17
	Subtasks: *Bash, C, C++, C#, Go, Haskell, Java, JavaScript, MatLab, PHP Python, R, Ruby, Rust, Scala, Swift, TypeScript*				
QA	ARC (2018)	Science QA	5	BPB	2
	Subtasks: *ARC-Easy, ARC-Challenge*				
	MMLU STEM (2021)	General QA	5	BPB	19
	MMLU Humanities (2021)	General QA	5	BPB	13
	MMLU Social Sci. (2021)	General QA	5	BPB	12
	MMLU Other (2021)	General QA	5	BPB	14
	CSQA (2019)	Commonsense QA	5	BPB	-
	HellaSwag (2019)	Language Modeling	5	BPB	-
	WinoGrande (2020)	Language Modeling	5	BPB	-
	SocialIQA (2019)	Social QA	5	BPB	-
	PiQA (2020)	Physical QA	5	BPB	-
	CoQA (2019)	Conversation QA	$0^{\dagger}$	BPB	-
	DROP (2019)	Passage QA	5	BPB	-
	Jeopardy (2024)	Trivia QA	5	BPB	-
	NaturalQs (2019)	General QA	5	BPB	-
	SQuAD (2016)	General QA	5	BPB	-
	SciQ (2017)	Science QA	5	BPB	-
	Basic Skills (2025)	Basic QA	5	BPB	6
	Subtasks: *Basic Arithmetic, String Manipulation, Simple Coding Elementary Logical Reasoning, Basic Common Sense, Simple Pattern Recognition*				
	Lambada (2016)	Language Modeling	0	BPB	-
	MedMCQA (2022)	Medical QA	5	BPB	-

Table 10: **Details of the BPB evaluation suite**. We use the OLMOBASEEVAL easy evaluation suite (Olmo et al., 2025), formatted as bits-per-byte (BPB) over gold continuations. † = few-shot examples are built-in the task; $^{\alpha}$ = human-written few-shot examples. We treat subtasks as standalone tasks, except for MMLU, where we use averages over the four MMLU categories.

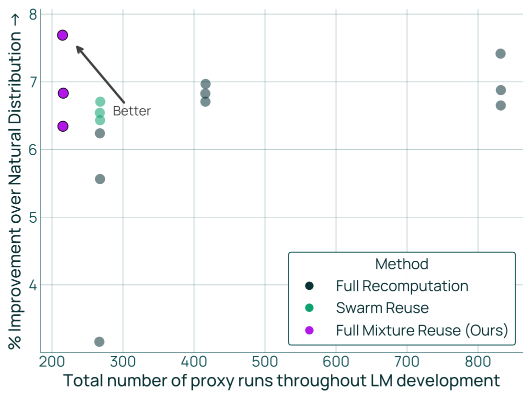

Figure 20: Performance improvement versus cost of mixing under evolving domains. FULLMIXTUREREUSE achieves 99% of the improvement of full recomputation while using 74% fewer proxy runs.

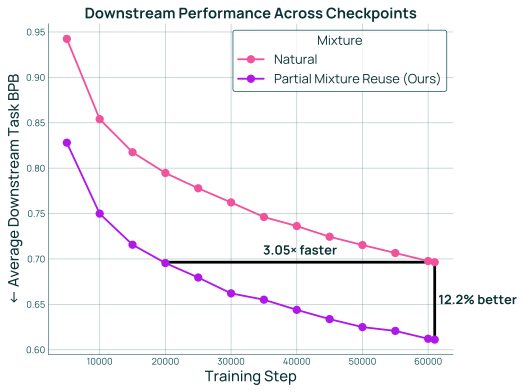

Figure 21: Downstream performance across training for PARTIALMIXTUREREUSE (best seed) versus the natural distribution. PARTIALMIXTUREREUSE reaches the natural distribution's final performance in 3.05× fewer steps.