R&B: Breaking the Data Mixing Bottleneck with Just 0.01% Overhead

Albert Ge [1] **Tzu-Heng Huang** [1] **John Cooper** [1] **Avi Trost** [1] **Ziyi Chu** [1] **Satya Sai Srinath Namburi GNVV** [1] **Ziyang Cai** [1] **Kendall Park** [1] **Nicholas Roberts** [1] **Frederic Sala** [1]

Abstract

Data mixing strategies have successfully reduced the costs involved in training language models. While promising, such methods suffer from two flaws. First, they rely on predetermined data domains (e.g., data sources, task types), which may fail to capture critical semantic nuances, leaving performance on the table. Second, these methods scale with the number of domains in a computationally prohibitive way. We address these challenges via R&B, a framework that repartitions training data based on semantic similarity (Regroup) to create finer-grained domains, and efficiently optimizes the data composition (Balance) by leveraging a Gram matrix induced by domain gradients obtained throughout training. Unlike prior works, it removes the need for additional compute to obtain evaluation information such as losses or gradients. We analyze this technique under standard regularity conditions and provide theoretical insights that justify R&B's effectiveness compared to non-adaptive mixing approaches. Empirically, we demonstrate the effectiveness of R&B on five diverse datasets ranging from natural language to reasoning and multimodal tasks. **With as little as 0.01% additional compute overhead,** R&B **matches or exceeds the performance** of state-of-the-art data mixing strategies.

1. Introduction

Large language models depend on vast, diverse datasets, but the shift to general-purpose foundation models has created a fundamental imbalance: potential training data vastly exceeds available computational resources. This has driven the development of data-efficient strategies that maximize performance while minimizing compute costs. Among these approaches, **data mixing** is particularly promising. By optimizing the composition of training data used—rather than simply increasing its volume—we can achieve comparable or superior performance with significantly fewer computational resources.

A wide variety of data mixture optimization techniques have been proposed (Fan et al., 2024b; Xie et al., 2023; Chen et al., 2023; 2024; Jiang et al., 2024). These adjust the relative proportions ("mixture") of training data from different predefined domains, also known as **skills**. Skills are often assigned to data based on human judgments or on source metadata (Wettig et al., 2025). We find, however, that these coarse human-defined categorizations fail to capture the optimal groupings for data mixing. That is, **human categorizations of skills are suboptimal when used for the development of LLM capabilities**.

Consider the Dolly-15k instruction-tuning dataset, which categorizes its data into general domains such as open-question answering, information extraction, and summarization (Conover et al., 2023). Rather than directly optimizing these coarse categories, our approach first re-partitions the data into finer-grained, semantically-grouped skills (Fig. 1, left). Optimizing the proportions across these semantically-clustered skills can significantly improve training performance over that of the general predefined domains. These improvements are even more pronounced when the number of skill partitions is optimized (Fig. 1, right).

However, the more granular semantic-based clustering approach has a *critical drawback*. Existing approaches typically require additional evaluations—either through forward passes over evaluation datasets for each skill or by computing per-skill gradient information derived from target tasks. To overcome this limitation, **we propose an efficient gradient-based approach that leverages information already computed during training, bypassing the need for these expensive evaluations.**

These insights motivate our approach, **Regroup & Balance** (R&B), a two-stage framework for efficient data mixture optimization. First, we repartition (Regroup) training data into semantically coherent clusters based on embedding similarity. Then, we dynamically optimize domain weights (Balance) to capture individual domain contributions and

[*]Equal contribution [1]Department of Computer Sciences, University of Wisconsin-Madison, Madison, WI. Correspondence to: Albert Ge <afge@wisc.edu>.

Published at Data in Generative Models Workshop: The Bad, the Ugly, and the Greats (DIG-BUGS) at ICML 2025, Vancouver, Canada. Copyright 2025 by the author(s).

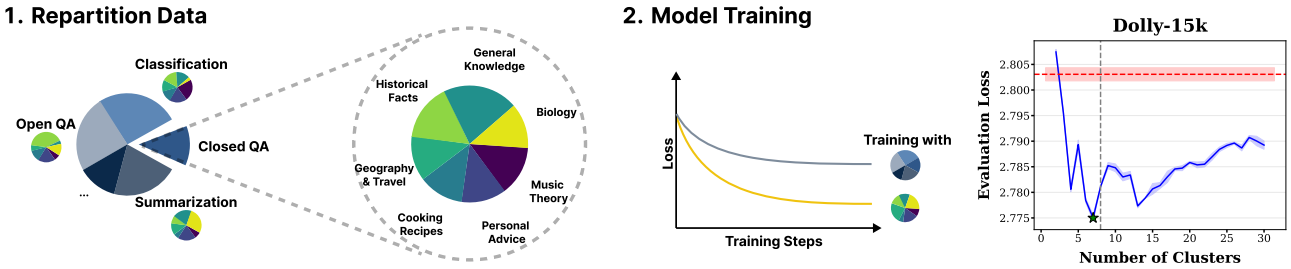


Figure 1. Instead of using pre-determined domains (e.g., by task type), we find that it is often better to first repartition the data into finer-grained, semantically related domains. Optimizing the proportions of these new semantic domains can significantly improve training performance.

cross-domain relationships, leveraging domain gradients computed throughout normal training. This produces the best of all worlds: it unlocks the performance gains in fine-grained skill clusters while dramatically reducing computational complexity.

Theoretically, we derive insights to characterize the importance of semantic-based clustering in data mixing strategies and the optimality of R&B. Empirically, we validate R&B across five diverse data settings, encompassing natural-language, instruction-following, reasoning, and multimodal tasks. **R&B only requires an additional 0.01% compute overhead, which cuts computational FLOPs by more than 99% relative to existing approaches, all while maintaining or improving model performance**.

We summarize our contributions as follows:

1. We establish that semantic-based categorizations of skills are superior to human-defined categories for foundation model data mixing algorithms (Section 2.1).
2. We introduce R&B, an efficient and theoretically justified two-stage framework that first repartitions data into semantically coherent clusters of skills, then dynamically reweights skill mixtures using their gradients (Sections 2, 2.1, and 2.3).
3. We theoretically and empirically demonstrate that R&B scales effectively with increased skill counts (Section 3).

2. R&B: Regroup and Balance Data Mixtures

We first provide some context and intuition. We will refer to skills and domains interchangeably throughout this paper. In data mixing, each data skill/domain is assigned a proportion weight from the probability simplex, and data is sampled according to this probability distribution. Formally, for m skills, we sample data according to the distribution $\boldsymbol{p} = [p_1, \ldots, p_m] \in \Delta^{m-1}$. We seek to answer two questions:

R1. How should we define domains for data mixing? Given a dataset, we wish to group the data into m partitions suitable for data mixing strategies. We define a mapping function $S : \mathcal{X} \to \{1, 2, \ldots, m\}$ which labels each data point x to its corresponding partition index. For a given dataset D, we have that $D_i = \{x \in D : S(x) = i\}$, and $\bigcup_{i=1}^{m} D_i = D$.

Intuitively, one would like to slice the data to minimize noise within each partition and reduce overlap between partitions. If perfect separation is possible, each partition would correspond to a distinct skill or capability domain, allowing for more targeted optimization of mixing weights.

R2. How do we efficiently reweight domains? Once D_{train} has been partitioned into m groups, our secondary objective is determining the optimal weight proportions $\boldsymbol{p}$ for each domain. Weights may change over time, so training is split into T rounds. At the end of each round t, we can reweight the skills $\boldsymbol{p}^{t+1} = [p_1^{t+1}, \ldots, p_m^{t+1}] \in \Delta^{m-1}$, and resume training according to the new proportions.

2.1. Problem Setup

We formulate our framework as a bilevel optimization problem for minimizing test loss. The lower-level optimization aims to find the best training proportions $\boldsymbol{p}^t \in \Delta^{m-1}$ for each training round $t \in 1, \ldots, T$. The upper-level problem seeks to find the best partitioning of a dataset D into m partitions $D_1, \ldots, D_m$ where $D = \bigcup_{i=1}^{m} D_m$. Let $D_{\text{eval},i} = \{x \in D_{\text{eval}} : S(x) = i\}$ be the set of evaluation points assigned to skill i by S. Let f_{θ_t} be the model parametrized by θ_t that is trained during round t, i.e. $f_{\theta_{t+1}}$ is obtained by training f_{θ_t} with proportions $\boldsymbol{p}^t$. Formally, we aim to solve the following problem

$$\min_{m \in \mathbb{Z}_+} \min_{\boldsymbol{p}^1, \ldots, \boldsymbol{p}^T \in \Delta^{m-1}} \mathcal{L}_{\text{eval}}(f_{\theta_{T+1}}), \tag{1}$$

where $\mathcal{L}_{\text{eval}}(f_{\theta_{T+1}})$ is the average evaluation loss for the partition $D_{\text{eval},i}$ after training model f_{θ_T} on mixture proportions $\boldsymbol{p}^T$ to obtain $f_{\theta_{T+1}}$.

Solving the full bilevel problem (1) is infeasible because for each candidate solution of the upper-level optimization prob-

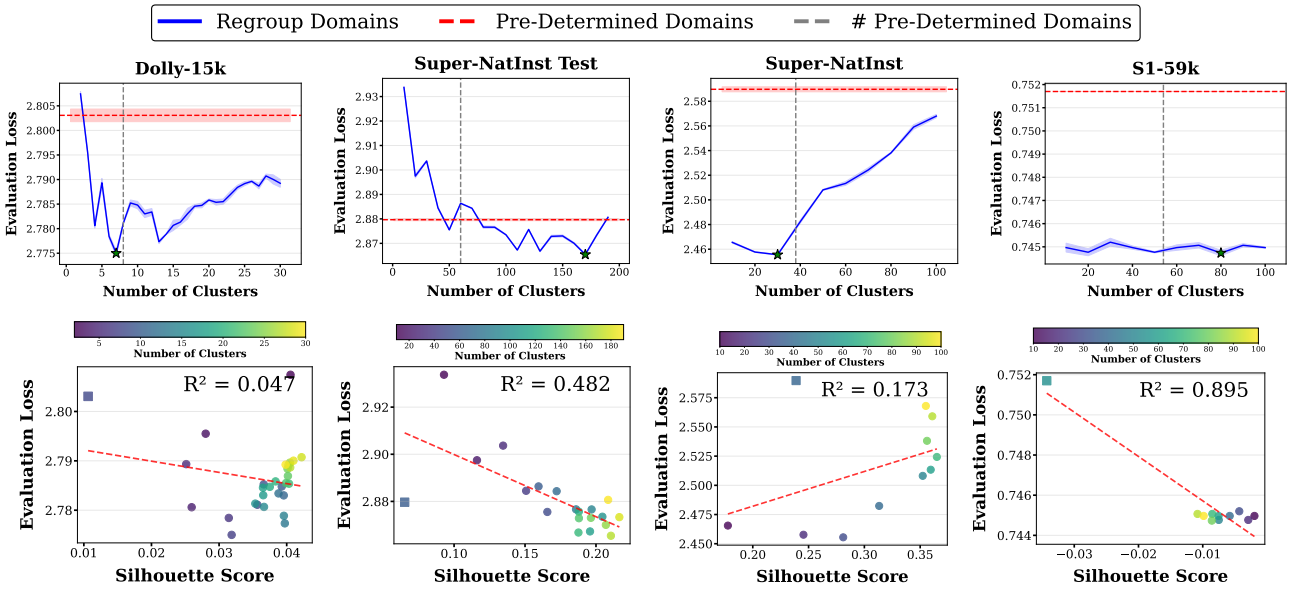


Figure 2. Top Row: Across various data settings, we find that there is a "sweet spot" in the number of domains used for data mixing, indicated by the green star. The optimal number of groups varies significantly with the dataset, which motivates the need for compute-efficient data mixing. *Bottom Row*: We find that silhouette score often correlates with model performance, suggesting that it is possible to predict data mixing performance based on clustering metrics.

lem, we must train a model for the lower-level optimization problem to obtain the loss. Thus, we propose decomposing Equation 1 into two:

$$S^*, m^* = \arg\min_{S,m} \mathcal{L}_{\text{clustering}}(S; f_{\text{Unif}}(D)), \tag{2}$$

$$\min_{\boldsymbol{p}^1,\ldots,\boldsymbol{p}^t \in \Delta^{m^*-1}} \mathcal{L}^*_{\text{eval}}(f_{\theta_{T+1}}). \tag{3}$$

In (2), we use f_{Unif}, which we denote as a model trained on fixed uniform proportions across skills to convergence (i.e. stratified sampling, $f_{\text{Unif}} = \arg\min_f \mathcal{L}_{\text{eval}}(f)$). This minimization is taken over a family of partitioning schemes S, such as the S found through k-means, on the gradients of trained model $\mathcal{L}_{\text{eval}}(f_{\text{Unif}})$. In (3), we use the optimal choice of m^* and partitioning scheme S^* found in the previous stage, and solve the optimization problem at every training round t.

2.2. Defining Domains

We investigate how to partition a given dataset to achieve optimal data mixing. Intuitively, data points that belong in a cluster should have a similar effect, i.e. gradients, during training. If gradients are not aligned, then swapping points with another cluster would reduce noise in both clusters. This leads to our first definition.

Definition 1. *A skill-assigning function $S : D \to [m]$ is stable in the direction $\nabla\mathcal{L}(\theta_t; \mathcal{D}_{\boldsymbol{p}})$ if for the skill $i = \arg\max_{i\in[m]} \nabla\mathcal{L}(\theta_t; D_i)^\top \nabla\mathcal{L}(\theta_t; \mathcal{D}_{\boldsymbol{p}})$ and any other $j \in [m]$, exchanging a pair $x_i \in D_i$, $x_j \in D_j$ does not improve $\nabla\mathcal{L}(\theta_t; D_i)^\top \nabla\mathcal{L}(\theta_t; \mathcal{D}_{\boldsymbol{p}})$.*

In other words, a clustering is said to be stable if no swapping of points improves alignment with the evaluation gradient. This definition provides a theoretical foundation for optimal data mixing, but it is still impractical to discover good groupings. The following lemma characterizes the maximum regret from swapping points between clusters.

Lemma 1. *Define the regret $R_S(i,j)$ under the skill- assigning function S for class j as:*

$$R_S(i,j) = \max_{\substack{\tilde{D}_i \subset D_i \cup D_j \\ |\tilde{D}_i| = |D_i|}} \nabla\mathcal{L}(\theta_t; \mathcal{D}_p)^\top \nabla\mathcal{L}(\theta_t, \tilde{D}_i) - \nabla\mathcal{L}(\theta_t; \mathcal{D}_p)^\top \nabla\mathcal{L}(\theta_t; D_i)$$

Let $i, j \in |m|$ with $|D_i| = |D_j|$ and

$$\nabla\mathcal{L}(\theta_t; D_i)^\top \nabla\mathcal{L}(\theta_t; \mathcal{D}_p) \geq \nabla\mathcal{L}(\theta_t; D_j)^\top \nabla\mathcal{L}(\theta_t; \mathcal{D}_p)$$

Define $r_i = \max_{x\in D_i} |\nabla\mathcal{L}(\theta_t, x)^\top \nabla\mathcal{L}(\theta_t; \mathcal{D}_p) - \nabla\mathcal{L}(\theta_t; D_i)^\top \nabla\mathcal{L}(\theta_t; \mathcal{D}_p)|$. Then:

$$R_S(i,j) \leq \max\Big\{0, \frac{1}{2}\Big(r_i + r_j - \Big(\nabla\mathcal{L}(\theta_t; D_i)^\top \nabla\mathcal{L}(\theta_t; \mathcal{D}_p) - \nabla\mathcal{L}(\theta_t; D_j)^\top \nabla\mathcal{L}(\theta_t; \mathcal{D}_p)\Big)\Big)\Big\}$$

Algorithm 1 R&B: Online Domain Data Selection

Input: Domain datasets $\{D^*_{\text{train},i}, D^*_{\text{eval},i}\}_{i=1}^{m^*}$, model $\theta \in \mathbb{R}^n$, training rounds T, steps per round K, evaluation weights $\boldsymbol{p} \in \mathbb{R}^{m^*}$
Initialize sampling distribution $\boldsymbol{p}^0 \leftarrow \text{Uniform}(m^*)$
for $t = 0$ to $T-1$ **do**
  Initialize gradient accumulators $\nabla\mathcal{L}(\theta; D_i) \leftarrow \mathbf{0}_n$ and sample sets $\mathcal{S}_i \leftarrow \emptyset$ for all i
  for $k = 0$ to $K-1$ **do**
    Sample mini-batch $\mathcal{B}$ from $D^*_{\text{train},i}$ using domain probabilities $\boldsymbol{p}^t$
    Update model: $\theta \leftarrow \theta - \eta\nabla\mathcal{L}(\theta; \mathcal{B})$
    for each domain i present in $\mathcal{B}$ **do**
      Accumulate final-layer gradients: $\nabla\mathcal{L}(\theta; D_i) \mathrel{+}= \nabla\mathcal{L}(\theta; \mathcal{B} \cap D^*_{\text{train},i})$
      Add samples to set: $\mathcal{S}_i \leftarrow \mathcal{S}_i \cup (\mathcal{B} \cap D^*_{\text{train},i})$
    end for
  end for
  Compute similarity matrix G with $G_{ij} = \frac{1}{|\mathcal{S}_i||\mathcal{S}_j|}\nabla\mathcal{L}(\theta; D_i)^\top \nabla\mathcal{L}(\theta; D_j)$
  Update distribution: $\boldsymbol{p}^{t+1} \leftarrow \text{softmax}(\lambda G\boldsymbol{p}/\|G\boldsymbol{p}\|_2)$
end for
Return: Final model parameters θ

Proof: See Appendix B.1.2.

This bound reveals additional structure that enables us to determine an effective clustering: clusters should have minimal radii while maintaining sufficient separation between their centroids. In light of these theoretical findings, we seek to empirically validate our claims by clustering real data and train using fixed proportions $\boldsymbol{p}$. Our hypothesis is that well-clustered data can result in better overall training performance. To keep our investigation tractable, we focus on k-means clustering, and sweep over k. We first embed our examples with ModernBERT-embed (Nussbaum et al., 2024), a state-of-the-art embedding model that supports long-context inputs. Then, we apply k-means and train a model using a uniform proportion of k clusters. We evaluate our setup across four settings: Dolly-15k (Conover et al., 2023), Super Natural Instructions (Super-NatInst) & Super Natural Instruction Test (Super-NatInst Test) (Wang et al., 2022), and S1-Reasoning (Muennighoff et al., 2025), and across 3 seeds. Appendix F lists the full experimental details.

The top row of Figure 2 shows that **training on the resulting clusters often results in significantly better performance** compared to pre-determined partitions. On 3 of the 4 datasets, there is a U-shaped pattern in the number of clusters versus evaluation loss. Thus in many cases, there is an optimal choice of k—but it varies significantly between datasets.

Furthermore, we often find that optimal clusters are compact and well-separated, as our theory suggests. We plot the silhouette score (Rousseeuw, 1987) of each cluster group against the final evaluation loss of the model. The bottom row of Figure 2 shows that on 3 of the 4 datasets, there is moderate to strong correlation between the clusters' silhouette score and model performance. **These results validate our theoretical insights that clusters which are well-separated result in better data mixing performance**. Furthermore, this suggests that it is possible to choose the optimal k without training a model, which would lead to further cost savings.

2.3. Proposed Method

The R&B algorithm (Algorithm 1) performs adaptive data selection for efficient model training on partitioned datasets. Starting with a uniform sampling distribution across clusters, it iteratively refines this distribution to focus computational resources on the most relevant partitions.

During each training round, R&B accumulates final-layer gradients from sampled batches and tracks which clusters contribute to model updates. Then it constructs a gradient similarity matrix that captures how gradients from different clusters relate to each other. This similarity information is combined with predefined evaluation proportions to produce an updated sampling distribution through a softmax operation. As training progresses, the algorithm adaptively shifts sampling probability toward clusters that contain the most valuable training examples, improving efficiency while maintaining performance across all partitions. See Appendix B.1 for derivation details.

3. Experiments

We study the effectiveness of R&B empirically across a diverse range of datasets and tasks. Our experiments aim to validate the following claims:

C1. R&B can *match or improve training performance* on natural language tasks while *significantly reducing computational overhead* compared to existing methods.

C2. R&B can improve training performance *beyond natural language modalities*.

Method	SUP-NATINST ($m = 38$)		SUP-NATINST test ($m = 60$)		Dolly-15k ($m = 8$)	
	Loss	% Overhead (↓)	Loss	% Overhead (↓)	Loss	% Overhead (↓)
Stratified	2.591	0	2.877	0	2.788	0
Skill-It	2.632	595.5	2.911	6×10^7	2.786	14.46
Aioli	2.622	1336.5	2.883	7×10^6	2.779	62.5
DGA	2.591	1.723	2.893	1601	2.787	0.41
R&B	($m^* = 30$)		($m^* = 100$)		($m^* = 7$)	
	2.381	**0.009**	**2.859**	**0.1**	**2.765**	**0.0006**

Table 1. Across three datasets, R&B **significantly reduces** the compute overhead for evaluation compared to existing methods, while **matching or exceeding performance**.

Method	SUP-NATINST		SUP-NATINST test		Dolly-15k	
	Original	Regroup	Original	Regroup	Original	Regroup
m	38	30	60	100	8	7
Stratified	2.591	2.454	2.877	2.871	2.788	2.761
Skill-It	2.632	2.812	2.911	2.881	2.786	2.778
Aioli	2.622	2.488	2.883	2.947	2.779	**2.760**
DGA	2.591	2.453	2.893	2.871	2.787	2.761
Balance	2.520	**2.381**	-	**2.859**	2.783	2.765

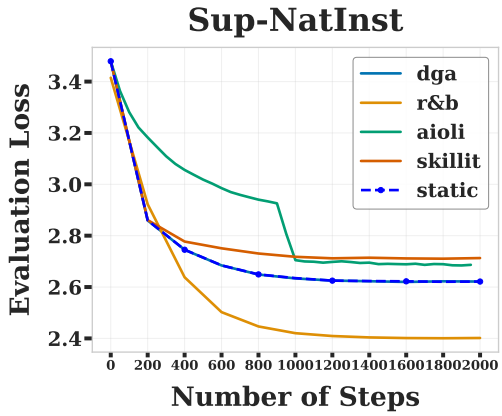


Figure 3. Left: Regrouping skills before applying data mixing strategies can yield substantial improvements. Underlined values indicate where regrouping beats the original grouping for that method and dataset. **Highlighted** values (with brown background) indicate the best overall performance for each dataset. Note that we do not apply Balance to the original categorization of SUP-NATINST test, as we assume that training data and validation data are bucketed into the same m groups. Right: Loss curve on the Sup-NatInst dataset.

3.1. Data Mixing on Natural Language Tasks

Setup. We compare R&B against four existing baselines: stratified sampling, Skill-It (Chen et al., 2023), Aioli (Chen et al., 2024), and DGA (Fan et al., 2024a). We compare each method across three distinct three natural-language data settings: Dolly-15k (Conover et al., 2023), NaturalInstructions In-domain SUP-NATINST (Wang et al., 2022), and NaturalInstructions Test SUP-NATINST. For all experiments, we train 125M GPT-Neo models (Black et al., 2021); full experimental details are listed in Appendix F. We report the final evaluation loss and the relative compute overhead (over standard training) incurred from re-estimating proportions. The formulas for the relative compute overhead are derived in Appendix D.

Results. Table 1 shows R&B's strong performance across all datasets. On SUP-NATINST, R&B achieves the best performance (loss: 2.381) with minimal computational overhead (0.009%). On SUP-NATINST test, R&B outperforms all methods (loss: 2.859) with only 0.1% overhead versus Skill-It (6×10^7%) and Aioli ((7×10^6%). On Dolly-15k, R&B performs competitively (loss: 2.765) compared to Aioli (2.779), with significantly smaller overhead (0.0006% versus 62.5%). As expected from C1, **R&B consistently delivers strong results with orders of magnitude better computational efficiency** than other data mixing approaches.

Ablations. Fig. 3 ablates semantic regrouping across data mixing strategies. In SUP-NATINST, **regrouping improves most methods** (5.3-8.1% gains) except Skill-It. On SUP-NATINST test, regrouping yields modest improvements for all methods except Aioli. Dolly-15k shows strong improvements with regrouping across all methods. While in many cases regrouping does help, it is not universally beneficial as evidenced by Skill-It's performance degradation on SUP-NATINST and Aioli's slight decline on SUP-NATINST test. Notably, the Balance method combined with regrouping achieves the best overall performance on both SUP-NATINST datasets, while Aioli with regrouping performs

Method	S1-59K	
	Original ($m = 54$)	Regroup ($m = 10$)
Stratified	0.7517	0.7449
R&B	-	0.7449

Table 2. Performance comparison of Stratified and R&B methods on the S1-59K dataset.

best on Dolly-15k.

Even without regrouping, **our gradient-based method shows strong performance while maintaining minimal overhead**. On original SUP-NATINST, Balance achieves a loss of 2.520, significantly outperforming other data-mixing methods. On Dolly-15k Original, it reaches a competitive 2.783 loss.

The convergence plot in Figure 3 demonstrates R&B's efficiency. R&B reaches convergence with only 20% of the training steps needed by other methods while achieving lower final loss values. Its smooth, monotonic descent contrasts with methods like Aioli, suggesting R&B's gradient-based domain weighting enables more consistent optimization.

3.2. Beyond Language Modeling

We next explore additional tasks our approach was not specifically designed for, including reasoning and multimodal setups.

Reasoning Setup. We investigate whether optimizing data mixtures can boost model performance in reasoning tasks. We use S1-Reasoning dataset (Muennighoff et al., 2025), which comprises reasoning traces from challenging math problems, drawn from 54 distinct sources. We use Qwen2-0.5B model (Yang et al., 2024) as an illustrative example.

Reasoning Results. Table 2 shows that regrouping improves performance compared to using original domains, with the optimal number of groups being 10. Specifically, the evaluation loss decreased from 0.7517 to 0.7449 when using our regrouping approach instead of predetermined domains. This suggests that while clustering generally improves the model performance, applying data mixing techniques such as R&B does not improve model performance further. We believe it is an open question as to whether data mixing can still be applied to such reasoning traces.

Multimodal Setup. We extend our setup to include multimodal tasks. We train CLIP models (Radford et al., 2021) from scratch using the small-scale DataComp dataset (Ilharco et al., 2021; Gadre et al., 2023). Our dataset comprises approximately 10 million image-caption pairs sourced from the web.[1] To ensure dataset quality, we select the top 30% of samples based on CLIP Score (Gadre et al., 2023), retaining 3.8 million high-quality pairs. We extract image embeddings for both the filtered training dataset and DataComp's evaluation benchmark, which spans 38 diverse downstream tasks. We apply k-means clustering to repartition data into varying numbers of groups.

m	Method	ImageNet	Dist. shift	VTAB	Retrieval	Avg (38)
10	Stratified	0.034	0.044	0.157	0.104	**0.146**
	R&B	0.033	0.040	0.153	0.104	0.141
20	Stratified	0.036	0.044	0.153	0.106	0.145
	R&B	0.031	0.042	0.163	0.103	**0.148**
50	Stratified	0.042	0.047	0.170	0.107	0.153
	R&B	0.042	0.047	0.177	0.108	**0.158**
100	Stratified	0.034	0.043	0.152	0.107	0.139
	R&B	0.041	0.047	0.151	0.104	**0.145**
150	Stratified	0.034	0.043	0.165	0.109	0.143
	R&B	0.039	0.050	0.164	0.109	**0.153**

Table 3. R&B performs better than stratified sampling on image-text modalities.

Results. We presented CLIP models' performance in Table 3. R&B outperforms stratified sampling when the number of domains exceeds 10. With 50 domains, **R&B achieves a 3.27% relative improvement** over the stratified sampling baseline. These findings highlight the effectiveness of R&B when the number of underlying domains is high and validate its extensibility to other modalities.

4. Conclusion

We introduced Regroup & Balance (R&B), a two-stage framework that breaks free from two fundamental constraints found in state-of-the-art data mixing strategies: the limitations of predetermined domains and the computational bottleneck of per-skill evaluations. Empirically, R&B matched or exceeded state-of-the-art data mixing methods while requiring two orders of magnitude less compute overhead, charting a more efficient path forward for foundation model training.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

[1] Some URLs provided by DataComp are now broken. See here for details.

References

Black, S., Gao, L., Wang, P., Leahy, C., and Biderman, S. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow, March 2021. URL https://doi.org/10.5281/zenodo.5297715.

Chen, M. F., Roberts, N., Bhatia, K., Wang, J., Zhang, C., Sala, F., and Ré, C. Skill-it! A Data-Driven Skills Framework for Understanding and Training Language Models, July 2023. URL http://arxiv.org/abs/2307.14430. arXiv:2307.14430 [cs].

Chen, M. F., Hu, M. Y., Lourie, N., Cho, K., and Ré, C. Aioli: A Unified Optimization Framework for Language Model Data Mixing, November 2024. URL http://arxiv.org/abs/2411.05735. arXiv:2411.05735.

Conover, M., Hayes, M., Mathur, A., Xie, J., Wan, J., Shah, S., Ghodsi, A., Wendell, P., Zaharia, M., and Xin, R. Free Dolly: Introducing the World's First Truly Open Instruction-Tuned LLM, 2023. URL https://www.databricks.com/blog/2023/04/12/dolly-first-open-commercially-viable-instruction-tuned-llm.

Fan, S., Grangier, D., and Ablin, P. Dynamic Gradient Alignment for Online Data Mixing, October 2024a. URL http://arxiv.org/abs/2410.02498. arXiv:2410.02498 [cs].

Fan, S., Pagliardini, M., and Jaggi, M. DoGE: Domain Reweighting with Generalization Estimation, February 2024b. URL http://arxiv.org/abs/2310.15393. arXiv:2310.15393.

Gadre, S. Y., Ilharco, G., Fang, A., Hayase, J., Smyrnis, G., Nguyen, T., Marten, R., Wortsman, M., Ghosh, D., Zhang, J., and others. Datacomp: In search of the next generation of multimodal datasets. *Advances in Neural Information Processing Systems*, 36:27092–27112, 2023.

Goodfellow, I. Efficient Per-Example Gradient Computations, October 2015. URL http://arxiv.org/abs/1510.01799. arXiv:1510.01799 [stat].

Hobbhahn, M. What's the Backward-Forward FLOP Ratio for Neural Networks?, December 2021. URL https://epoch.ai/blog/backward-forward-FLOP-ratio.

Ilharco, G., Wortsman, M., Wightman, R., Gordon, C., Carlini, N., Taori, R., Dave, A., Shankar, V., Namkoong, H., Miller, J., Hajishirzi, H., Farhadi, A., and Schmidt, L. OpenCLIP, July 2021. URL https://doi.org/10.5281/zenodo.5143773.

Jiang, Y., Zhou, A., Feng, Z., Malladi, S., and Kolter, J. Z. Adaptive Data Optimization: Dynamic Sample Selection with Scaling Laws, October 2024. URL http://arxiv.org/abs/2410.11820. arXiv:2410.11820.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling Laws for Neural Language Models, January 2020. URL http://arxiv.org/abs/2001.08361. arXiv:2001.08361 [cs].

Muennighoff, N., Yang, Z., Shi, W., Li, X. L., Fei-Fei, L., Hajishirzi, H., Zettlemoyer, L., Liang, P., Candès, E., and Hashimoto, T. s1: Simple test-time scaling, 2025. URL https://arxiv.org/abs/2501.19393. _eprint: 2501.19393.

Nussbaum, Z., Morris, J. X., Duderstadt, B., and Mulyar, A. Nomic Embed: Training a Reproducible Long Context Text Embedder, 2024. _eprint: 2402.01613.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., and others. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PmLR, 2021.

Rousseeuw, P. J. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987. ISSN 0377-0427. doi: https://doi.org/10.1016/0377-0427(87)90125-7. URL https://www.sciencedirect.com/science/article/pii/0377042787901257.

Wang, J. T., Wu, T., Song, D., Mittal, P., and Jia, R. GREATS: Online Selection of High-Quality Data for LLM Training in Every Iteration. November 2024. URL https://openreview.net/forum?id=232VcN8tSx&referrer=%5Bthe%20profile%20of%20Ruoxi%20Jia%5D(%2Fprofile%3Fid%3D~Ruoxi_Jia1).

Wang, Y., Mishra, S., Alipoormolabashi, P., Kordi, Y., Mirzaei, A., Arunkumar, A., Ashok, A., Dhanasekaran, A. S., Naik, A., Stap, D., Pathak, E., Karamanolakis, G., Lai, H. G., Purohit, I., Mondal, I., Anderson, J., Kuznia, K., Doshi, K., Patel, M., Pal, K. K., Moradshahi, M., Parmar, M., Purohit, M., Varshney, N., Kaza, P. R., Verma, P., Puri, R. S., Karia, R., Sampat, S. K., Doshi, S., Mishra, S., Reddy, S., Patro, S., Dixit, T., Shen, X., Baral, C., Choi, Y., Smith, N. A., Hajishirzi, H., and Khashabi, D. Super-NaturalInstructions: Generalization via Declarative Instructions on 1600+ NLP Tasks, October 2022. URL http://arxiv.org/abs/2204.07705. arXiv:2204.07705 [cs].

Wettig, A., Lo, K., Min, S., Hajishirzi, H., Chen, D., and Soldaini, L. Organize the Web: Constructing Domains Enhances Pre-Training Data Curation, February 2025. URL http://arxiv.org/abs/2502.10341. arXiv:2502.10341 [cs].

Xie, S. M., Pham, H., Dong, X., Du, N., Liu, H., Lu, Y., Liang, P., Le, Q. V., Ma, T., and Yu, A. W. DoReMi: Optimizing Data Mixtures Speeds Up Language Model Pretraining, November 2023. URL http://arxiv.org/abs/2305.10429. arXiv:2305.10429 [cs].

Yang, A., Yang, B., Hui, B., Zheng, B., Yu, B., Zhou, C., Li, C., Li, C., Liu, D., Huang, F., Dong, G., Wei, H., Lin, H., Tang, J., Wang, J., Yang, J., Tu, J., Zhang, J., Ma, J., Xu, J., Zhou, J., Bai, J., He, J., Lin, J., Dang, K., Lu, K., Chen, K., Yang, K., Li, M., Xue, M., Ni, N., Zhang, P., Wang, P., Peng, R., Men, R., Gao, R., Lin, R., Wang, S., Bai, S., Tan, S., Zhu, T., Li, T., Liu, T., Ge, W., Deng, X., Zhou, X., Ren, X., Zhang, X., Wei, X., Ren, X., Fan, Y., Yao, Y., Zhang, Y., Wan, Y., Chu, Y., Liu, Y., Cui, Z., Zhang, Z., and Fan, Z. Qwen2 Technical Report. *arXiv preprint arXiv:2407.10671*, 2024.

The appendix is structured as follows. Appendix A introduces our notation, followed by theoretical insights and proofs in Appendix B. Algorithmic details are provided in Appendix C. We then analyze the computational cost of existing data mixing methods in Appendix D. Implementation specifics and experimental setups are detailed in Appendix E and Appendix F, respectively. Appendix G presents the results of our ablation studies. Lastly, we give interpretation for determined domains in Appendix H.

A. Notation

Symbol	Meaning
d	The dimensionality of each skill (toy theory)
m	The number of skills
$[\cdot]$	The sequence of numbers $1, 2, \ldots, n$
Δ^{m-1}	The $m-1$ dimensional simplex
$\mathcal{D}$	A mixture data distribution
$\mathcal{D}_i$	The distribution for skill i
$\mathcal{D}_{\boldsymbol{p}}$	The mixture of $\mathcal{D}_i$ according to $\boldsymbol{p}$
D_i	A sample from the distribution $\mathcal{D}_i$
G	The inner product between gradients of different skills $\nabla\mathcal{L}(\theta;\mathcal{D}_i)$
$\mathcal{L}(\theta;\mathcal{D})$	The expected loss of θ over $\mathcal{D}_{\boldsymbol{p}}$
$\nabla\mathcal{L}(\theta;\mathcal{D}_i)$	The skill gradient for skill i
$\boldsymbol{p}$	The evaluation data proportions
$\boldsymbol{p}'$	The chosen training data proportions
S	A skill-assigning function $S : \mathcal{X} \to [m]$

B. Theoretical Results

The goal is to find the best data mixture throughout training. With the many degrees of freedom such an algorithm can take, a few assumptions are made. First, an update is restricted to being performed by SGD with the following update rule:

$$\theta_{t+1} = \theta_t - \eta\nabla\mathcal{L}(\theta_t;\mathcal{D}_{\boldsymbol{p}'}).$$

The modification from standard SGD is the ability to change $\boldsymbol{p}'$ which allows for a different sampling mixture.

B.1. Method Derivation

The design of the proportion finding algorithm described here comes from one core assumption: the gradient update works roughly linearly. Specifically, we assume, for a small enough ball around θ_t,

$$\mathcal{L}(\theta;\mathcal{D}_{\boldsymbol{p}}) \approx \mathcal{L}(\theta_t;\mathcal{D}_{\boldsymbol{p}}) + \nabla\mathcal{L}(\theta_t;\mathcal{D}_{\boldsymbol{p}})^\top(\theta - \theta_t).$$

Inputting the SGD update rule,

$$\mathcal{L}(\theta_{t+1};\mathcal{D}_{\boldsymbol{p}}) \approx \mathcal{L}(\theta_t;\mathcal{D}_{\boldsymbol{p}}) - \eta\nabla\mathcal{L}(\theta_t;\mathcal{D}_{\boldsymbol{p}})^\top\nabla\mathcal{L}(\theta_t;\mathcal{D}_{\boldsymbol{p}'}).$$

Since the gradient is linear, we can treat $\nabla\mathcal{L}(\theta_t;\mathcal{D}_{\boldsymbol{p}})$ as a weighted sum of $\nabla\mathcal{L}(\theta_t;\mathcal{D}_i)$, i.e. the individual skill gradients, based on the proportions $\boldsymbol{p}_i$. Define the Gram matrix G as

$$G_{ij}^{(t)} = \nabla\mathcal{L}(\theta_t;\mathcal{D}_i)^\top\nabla\mathcal{L}(\theta_t;\mathcal{D}_j).$$

Note that $G^{(t)}$ is dependent on the iteration, but this will often be written only as G for notational clarity. This matrix also has a clear interpretation as a neural tangent kernel (NTK) aggregated over each skill rather than over each data point as is commonly used. With this matrix and treating $\boldsymbol{p}$ and $\boldsymbol{p}'$ as vectors, our assumption simplifies to

$$\mathcal{L}(\theta_{t+1};\mathcal{D}_{\boldsymbol{p}}) \approx \mathcal{L}(\theta_t;\mathcal{D}_{\boldsymbol{p}}) - \eta\boldsymbol{p}^\top G\boldsymbol{p}'.$$

Since the aim is to decrease loss as much as possible, we want to maximize $\eta\boldsymbol{p}^\top G\boldsymbol{p}'$ over $\boldsymbol{p}' \in \Delta^{m-1}$. This objective is linear over the simplex, so optima will only be found at the corners of the simplex. However, this would imply only one class is sampled from. This causes a few issues. First, the per-skill gradient computation method requires samples from skill

i to find the gradient for skill i. No information about other skills will be gathered through training. Second, the gradient as a function of time will be discontinuous. As the maximum skill of the optimization changes, the gradient will instantly change to the skill gradient for the new maximal skill. We impose a common regularity to restrict $\boldsymbol{p}'$ to the simplex by solving for the following:

$$\boldsymbol{p}' = \underset{\boldsymbol{p}' \in \Delta^{m-1}}{\arg\max}\, \eta \boldsymbol{p}^\top G \boldsymbol{p}' - \lambda_1 \sum_{i=1}^{m} \boldsymbol{p}'_i \log \boldsymbol{p}'_i + \lambda_2 \sum_{i=1}^{m} \boldsymbol{p}'_i.$$

Equivalently, we can multiply the λ_1 and λ_2 terms by a constant without affecting the optimum. This constant $\|\eta G p\|_2$ is chosen to make the parameter λ_1 have a roughly equal effect between the cross entropy term of the base objective, regardless of η, G, and $\boldsymbol{p}$. This also means that as the model trains and the gradients decrease in magnitude, the effects of the regularization term will decrease. Otherwise, proportions will tend towards uniform. Now, solve for:

$$\boldsymbol{p}' = \underset{\boldsymbol{p}' \in \Delta^{m-1}}{\arg\max}\, \eta \boldsymbol{p}^\top G \boldsymbol{p}' - \lambda_1 \|\eta G \boldsymbol{p}\| \sum_{i=1}^{m} \boldsymbol{p}'_i \log \boldsymbol{p}'_i + \lambda_2 \|\eta G \boldsymbol{p}\| \sum_{i=1}^{m} \boldsymbol{p}'_i.$$

The parameter λ_1 acts as a normal hyper-parameter, but λ_2 is a Lagrange multiplier enforcing $\boldsymbol{p}'$ lay on the simplex. Taking a gradient and setting to 0,

$$\begin{aligned}
0 &= \eta G \boldsymbol{p} - \lambda_1 \|\eta G \boldsymbol{p}\| \log \boldsymbol{p}' + (\lambda_1 + \lambda_2) \|\eta G \boldsymbol{p}\| \mathbf{1}_m, \\
\log \boldsymbol{p}' &= \frac{\eta}{\lambda_1 \|\eta G \boldsymbol{p}\|} G \boldsymbol{p} + \frac{\lambda_1 + \lambda_2}{\lambda_1} \mathbf{1}_m, \\
\boldsymbol{p}' &= \exp\left(\frac{\eta}{\lambda_1 \|\eta G \boldsymbol{p}\|} G \boldsymbol{p}\right) \exp\left(\frac{\lambda_1 + \lambda_2}{\lambda_1}\right) \\
&= \exp\left(\frac{1}{\lambda_1 \|G \boldsymbol{p}\|} G \boldsymbol{p}\right) \exp\left(\frac{\lambda_1 + \lambda_2}{\lambda_1}\right).
\end{aligned}$$

Here, $\mathbf{1}_m$ is the vector of all 1s of dimension m. Also, λ_2 will take on a value to make $\boldsymbol{p}'$ sum to 1. Thus, setting $Z = \sum_{i=1}^{m} \exp\left(\frac{1}{\lambda_1 \|G \boldsymbol{p}\|} G \boldsymbol{p}\right)_i$ and letting $\lambda = 1/\lambda_1$, we have

$$\begin{aligned}
\boldsymbol{p}' &= \frac{1}{Z} \exp\left(\frac{\lambda}{\|G \boldsymbol{p}\|} G \boldsymbol{p}\right) \\
&= \mathrm{softmax}\left(\frac{\lambda}{\|G \boldsymbol{p}\|} G \boldsymbol{p}\right).
\end{aligned}$$

Note that in this parametrization, a large λ indicates a small penalty from the entropy term. This solution aligns with the unconstrained solution described above from finding the maximal corner of the simplex, except now the solution is smooth.

B.1.1. Max can be Better than Static proportions

This method, for a small enough learning rate, will result in a reduction of a smooth loss greater than any other fixed proportions.

Lemma 2. *Let ℓ be an L-smooth loss function and f_θ is learned with SGD on datasets $\mathcal{D}_i$ with associated sampling prior $\boldsymbol{p}_i$, and let $\boldsymbol{p}''$ be some other fixed distribution of datasets. If the learning rate η satisfies $\eta \leq \frac{1}{L}\left(\frac{\max_i \nabla\mathcal{L}(\theta_0;\mathcal{D}_i)^\top \nabla\mathcal{L}(\theta_0;\mathcal{D}_{\boldsymbol{p}}) - (\nabla\mathcal{L}(\theta_0;\mathcal{D}_{\boldsymbol{p}''}))^\top \nabla\mathcal{L}(\theta_0;\mathcal{D}_{\boldsymbol{p}})}{\max_i \|\nabla\mathcal{L}(\theta_0;\mathcal{D}_i)\|^2 + \|\nabla\mathcal{L}(\theta_0;\mathcal{D}_{\boldsymbol{p}''})\|^2}\right)$, then a gradient descent step with $\boldsymbol{p}' = \delta_{\arg\max_i (G\boldsymbol{p})_i}$ results in a greater (or equal) decrease in the loss than $\boldsymbol{p}''$.*

Proof. Let θ_0 be some base parameter, $\boldsymbol{p}' = \delta_i$ for $i = \arg\max_i (G\boldsymbol{p})_i$, and let

$$\begin{aligned}
\theta_{\boldsymbol{p}'} &= \theta_0 - \eta \nabla\mathcal{L}(\theta_0; \mathcal{D}_{\boldsymbol{p}'}), \\
\theta_{\boldsymbol{p}''} &= \theta_0 - \eta \nabla\mathcal{L}(\theta_0; \mathcal{D}_{\boldsymbol{p}''}).
\end{aligned}$$

Assume that $\mathcal{L}$ is L-smooth. Now consider

$$\begin{aligned}\mathcal{L}(\theta_{\boldsymbol{p}''};\mathcal{D}_{\boldsymbol{p}}) - \mathcal{L}(\theta_{\boldsymbol{p}'};\mathcal{D}_{\boldsymbol{p}}) &= (\mathcal{L}(\theta_{\boldsymbol{p}''};\mathcal{D}_{\boldsymbol{p}}) - \mathcal{L}(\theta_0;\mathcal{D}_{\boldsymbol{p}})) - (\mathcal{L}(\theta_{\boldsymbol{p}'};\mathcal{D}_{\boldsymbol{p}}) - \mathcal{L}(\theta_0;\mathcal{D}_{\boldsymbol{p}})) \\ &\geq -\eta \boldsymbol{p}^\top G\boldsymbol{p}'' + \eta \boldsymbol{p}^\top G\boldsymbol{p}' - \eta^2 L\boldsymbol{p}'^\top G\boldsymbol{p}' - \eta^2 L p''^\top G\boldsymbol{p}'' \\ &= \eta(\max_i (G\boldsymbol{p})_i - \boldsymbol{p}^\top G\boldsymbol{p}'' - \eta L(\boldsymbol{p}'^\top G\boldsymbol{p}' + \boldsymbol{p}''^\top G\boldsymbol{p}'')).\end{aligned}$$

Now let $i = \arg\max_i (G\boldsymbol{p})_i$ and let $\eta \leq \frac{1}{L}\left(\frac{(G\boldsymbol{p})_i - \boldsymbol{p}^\top G\boldsymbol{p}''}{G_{ii} + \boldsymbol{p}''^\top G\boldsymbol{p}''}\right)$. This quantity is positive (and therefore well defined for $\eta \geq 0$ since $\boldsymbol{p}^\top G\boldsymbol{p}''$ is the convex combination of values all less than (or equal to) $(G\boldsymbol{p})_i$. Thus,

$$\begin{aligned}\mathcal{L}(\theta_{\boldsymbol{p}''};\mathcal{D}_{\boldsymbol{p}}) - \mathcal{L}(\theta_{\boldsymbol{p}'};\mathcal{D}_{\boldsymbol{p}}) &\geq \eta((G\boldsymbol{p})_i - \boldsymbol{p}^\top G\boldsymbol{p}'' - \eta L(G_{ii} + \boldsymbol{p}''^\top G\boldsymbol{p}'')) \\ &\geq \eta((G\boldsymbol{p})_i - \boldsymbol{p}^\top G\boldsymbol{p}'' - \frac{1}{L}\left(\frac{(G\boldsymbol{p})_i - \boldsymbol{p}^\top G\boldsymbol{p}''}{G_{ii} + \boldsymbol{p}''^\top G\boldsymbol{p}''}\right) L(G_{ii} + \boldsymbol{p}''^\top G\boldsymbol{p}'')) \\ &= \eta((G\boldsymbol{p})_i - \boldsymbol{p}^\top G\boldsymbol{p}'' - ((G\boldsymbol{p})_i - \boldsymbol{p}^\top G\boldsymbol{p}'')) \\ &= 0.\end{aligned}$$

Therefore, for a small enough learning rate, the choosing the gradient with the largest skill results in a larger decrease in the loss than using priors proportional to the evaluation data. The learning rate can also be bounded using gradient notation and taking a maximum over G_{ii}:

$$\eta \leq \frac{1}{L}\left(\frac{\max_i \nabla\mathcal{L}(\theta_{\theta_0};\mathcal{D}_i)^\top \nabla\mathcal{L}(\theta_{\theta_0};\mathcal{D}_{\boldsymbol{p}}) - (\nabla\mathcal{L}(\theta_{\theta_0};\mathcal{D}_{\boldsymbol{p}''}))^\top \nabla\mathcal{L}(\theta_{\theta_0};\mathcal{D}_{\boldsymbol{p}})}{\max_i \|\nabla\mathcal{L}(\theta_{\theta_0};\mathcal{D}_i)\|^2 + \|\nabla\mathcal{L}(\theta_0;\mathcal{D}_{\boldsymbol{p}''})\|^2}\right).$$

□

B.1.2. Clustering

One major phenomenon observed here is that clustering the data points works well for domain mixing, and sometimes even outperforms the provided labels for the different skills. These clusters are taken via the embeddings of some other model, which are assumed to mimic the gradients of the model being learned.

When $\eta \approx 0$, the change in the loss can be well-approximated by its first order Taylor expansion:

$$\mathcal{L}(\theta_t;\mathcal{D}_{\boldsymbol{p}}) - \mathcal{L}(\theta_{t+1};\mathcal{D}_{\boldsymbol{p}}) \approx \eta\nabla\mathcal{L}(\theta_t;\mathcal{D}_{\boldsymbol{p}})^\top \nabla\mathcal{L}(\theta_t;\mathcal{D}_{\boldsymbol{p}'}).$$

For the following sections, let $D_i = \{x \in D : S(x) = i\}$ for some skill-assigning function S.

Definition 2. *A skill-assigning function $S : D \to [m]$ is <u>stable in the direction $\nabla\mathcal{L}(\theta_t;\mathcal{D}_{\boldsymbol{p}})$</u> if for the skill $i = \arg\max_{i\in[m]} \nabla\mathcal{L}(\theta_t;D_i)^\top \nabla\mathcal{L}(\theta_t;\mathcal{D}_{\boldsymbol{p}})$ and any other $j \in [m]$, exchanging a pair $x_i \in D_i$, $x_j \in D_j$ does not improve $\nabla\mathcal{L}(\theta_t;D_i)^\top \nabla\mathcal{L}(\theta_t;\mathcal{D}_{\boldsymbol{p}})$.*

Intuitively, a stable clustering is one which doesn't improve $\nabla\mathcal{L}(\theta_t;D_i)^\top \nabla\mathcal{L}(\theta_t;\mathcal{D}_{\boldsymbol{p}})$ by exchanging points between classes.

Lemma 3. *Let S be some $\{0,1\}$ skill-assigning function on D with $\nabla\mathcal{L}(\theta_t;D_0)^\top \nabla\mathcal{L}(\theta_t;\mathcal{D}_{\boldsymbol{p}}) \geq \nabla\mathcal{L}(\theta_t;D_1)^\top \nabla\mathcal{L}(\theta_t;\mathcal{D}_{\boldsymbol{p}})$ and let $x_0, x_1 \in D$ with $S(x_0) = 0$ and $S(x_1) = 1$, and let $\tilde{C}$ be the clustering identical to S except on x_0, x_1 where each is assigned to the opposite class. Then, $\nabla\mathcal{L}(\theta_t;D_{\tilde{S},\boldsymbol{p}'})^\top \nabla\mathcal{L}(\theta_t;\mathcal{D}_{\boldsymbol{p}})^\top > \nabla\mathcal{L}(\theta_t;D_{S,\boldsymbol{p}'})^\top \nabla\mathcal{L}(\theta_t;\mathcal{D}_{\boldsymbol{p}})$ if $\nabla\mathcal{L}(\theta_t,x_1)^\top \nabla\mathcal{L}(\theta_t;\mathcal{D}_{\boldsymbol{p}}) > \nabla\mathcal{L}(\theta_t,x_0)^\top \nabla\mathcal{L}(\theta_t;\mathcal{D}_{\boldsymbol{p}})$.*

Proof. Let $D_i = \{x \in D : S(x) = i\}$, $\tilde{D}_i = \{x \in D : \tilde{S}(x) = i\}$, $n_i = |D_i|$, and $\boldsymbol{p}'$ be the proportions that put mass only on the maximal value of $A\nabla\mathcal{L}(\theta_t;\mathcal{D}_{\boldsymbol{p}})$. If we define

$$A = \begin{bmatrix}\nabla\mathcal{L}(\theta_t;D_0) \\ \nabla\mathcal{L}(\theta_t;D_1)\end{bmatrix},$$

then

$$\nabla\mathcal{L}(\theta_t; D_S)^\top \nabla\mathcal{L}(\theta_t; \mathcal{D}_{\boldsymbol{p}}) = \boldsymbol{p}'^\top A \nabla\mathcal{L}(\theta_t; \mathcal{D}_{\boldsymbol{p}}),$$
$$\nabla\mathcal{L}(\theta_t; D_{\tilde{S}})^\top \nabla\mathcal{L}(\theta_t; \mathcal{D}_{\boldsymbol{p}}) = \boldsymbol{p}'^\top \tilde{A} \nabla\mathcal{L}(\theta_t; \mathcal{D}_{\boldsymbol{p}})$$
$$= \boldsymbol{p}'^\top A \nabla\mathcal{L}(\theta_t; \mathcal{D}_{\boldsymbol{p}}) + \boldsymbol{p}'^\top \begin{bmatrix} \frac{1}{n_1}\nabla\mathcal{L}(\theta_t, x_1)^\top \nabla\mathcal{L}(\theta_t; \mathcal{D}_{\boldsymbol{p}}) - \frac{1}{n_1}\nabla\mathcal{L}(\theta_t, x_0)^\top \nabla\mathcal{L}(\theta_t; \mathcal{D}_{\boldsymbol{p}}) \\ \frac{1}{n_0}\nabla\mathcal{L}(\theta_t, x_0)^\top \nabla\mathcal{L}(\theta_t; \mathcal{D}_{\boldsymbol{p}}) - \frac{1}{n_0}\nabla\mathcal{L}(\theta_t, x_1)^\top \nabla\mathcal{L}(\theta_t; \mathcal{D}_{\boldsymbol{p}}) \end{bmatrix}.$$

Therefore swapping the classes of x_0 and x_1 results in an improvement for $\tilde{S}$ over S if $\nabla\mathcal{L}(\theta_t, x_1)^\top \nabla\mathcal{L}(\theta_t; \mathcal{D}_{\boldsymbol{p}}) > \nabla\mathcal{L}(\theta_t, x_0)^\top \nabla\mathcal{L}(\theta_t; \mathcal{D}_{\boldsymbol{p}})$. □

We immediately have the following corollary:

Lemma 4. *A skill-assigning function $S : D \to [m]$ is stable in the direction $\nabla\mathcal{L}(\theta_t; \mathcal{D}_{\boldsymbol{p}})$ if for the skill $i = \arg\max_{i\in[m]} \nabla\mathcal{L}(\theta_t; D_i)^\top \nabla\mathcal{L}(\theta_t; \mathcal{D}_{\boldsymbol{p}})$ and any other $j \in [m]$, for all $x_i \in D_i$, $x_j \in D_j$,*

$$\nabla\mathcal{L}(\theta_t; \mathcal{D}_{\boldsymbol{p}})^\top \nabla\mathcal{L}(\theta_t, x_i) \geq \nabla\mathcal{L}(\theta_t; \mathcal{D}_{\boldsymbol{p}})^\top \nabla\mathcal{L}(\theta_t, x_j).$$

An important fact to note is that the evaluation gradient $\nabla\mathcal{L}(\theta_t; \mathcal{D}_{\boldsymbol{p}})$ can be arbitrary, especially if the evaluation and training data come from different distributions. To reduce the benefit of swapping points between classes, a good clustering will be stable in as many directions as possible.

A simple but noisy choice is to take all points in the convex hull to be in unique clusters, and all interior points to make up another cluster. While this clustering is stable in every direction, the classes are very small and therefore likely to be noisy, especially as training progresses and the gradient landscape shifts. A better alternative is clustering points if they can be linearly separated from the others.

Assume D can be partitioned into D_0 and D_1 such that $f(x) = \text{sign}(\boldsymbol{v}^\top x + b)$ is a perfect classifier, so in the direction $\boldsymbol{v}$, S which labels the data based on the partition is stable. If f has a large margin, then many other $\boldsymbol{v}$ also a linear separators, and therefore also have S stable.

This still may be too restrictive though in general settings where data points are more spread apart. Instead, it may be good to compare the regret of skill-labeling with a sub-optimal labeling.

Definition 3. *The regret $R_S(i, j)$ under the skill-assigning function S for class j is the difference between $\max_{\tilde{D}_i \subset D_i \cup D_j, |\tilde{D}_i| = |D_i|} \nabla\mathcal{L}(\theta_t; \mathcal{D}_{\boldsymbol{p}})^\top \nabla\mathcal{L}(\theta_t, \tilde{D}_i)$ and $\nabla\mathcal{L}(\theta_t; \mathcal{D}_{\boldsymbol{p}})^\top \nabla\mathcal{L}(\theta_t; D_i)$.*

This regret is exactly difference between the first-order loss decrease using $\tilde{D}_i$ as compared to D_i, where new elements in $\tilde{D}_i$ come from D_j.

Lemma 5. *Let $i, j \in |m|$ and assume $|D_i| = |D_j|$. Assume $\nabla\mathcal{L}(\theta_t; D_i)^\top \nabla\mathcal{L}(\theta_t; \mathcal{D}_{\boldsymbol{p}}) \geq \nabla\mathcal{L}(\theta_t; D_j)^\top \nabla\mathcal{L}(\theta_t; \mathcal{D}_{\boldsymbol{p}})$, and let $r_i = \max_{x\in D_i} |\nabla\mathcal{L}(\theta_t, x)^\top \nabla\mathcal{L}(\theta_t; \mathcal{D}_{\boldsymbol{p}}) - \nabla\mathcal{L}(\theta_t; D_i)^\top \nabla\mathcal{L}(\theta_t; \mathcal{D}_{\boldsymbol{p}})|$ and similarly define r_j. Then*

$$R_S(i, j) \leq \max\{0, \frac{1}{2}(r_i + r_j - (\nabla\mathcal{L}(\theta_t; D_i)^\top \nabla\mathcal{L}(\theta_t; \mathcal{D}_{\boldsymbol{p}}) - \nabla\mathcal{L}(\theta_t; D_j)^\top \nabla\mathcal{L}(\theta_t; \mathcal{D}_{\boldsymbol{p}})))\}.$$

Proof. Let $R_i = \{\nabla\mathcal{L}(\theta_t, x)^\top \nabla\mathcal{L}(\theta_t; \mathcal{D}_{\boldsymbol{p}}) | x \in D_i\}$ and $R_j = \{\nabla\mathcal{L}(\theta_t, x)^\top \nabla\mathcal{L}(\theta_t; \mathcal{D}_{\boldsymbol{p}}) | x \in D_j\}$. Also, in this notation, $r_i = \max_{x\in R_i} |x - \mathbb{E}_{x\sim R_i}[x]|$, and define $\delta = \mathbb{E}_{x\in R_i}[x] - \mathbb{E}_{x\in R_j}[x]$. The problem then reduces to

$$R_S(i, j) \leq \max\{0, \frac{1}{2}(r_i + r_j - \delta)\}.$$

Also, $R_S(i, j)$ in this one dimensional case becomes the largest over $m \in [|R_i|]$ of the difference between the m largest values of R_j and the m smallest values of R_i. This is maximized over all possible R_i and R_j when half of R_i is $\mathbb{E}_{x\in R_i}[x] - r_i$ and the other half is $\mathbb{E}_{x\in R_i}[x] + r_i$, and similarly for R_j. The difference between the max R_j and the min R_i is $r_i + r_j - \delta$, and only half of these data points attain these max and min values, so $R_S(i, j) \leq \max\{0, \frac{1}{2}(r_i + r_j - \delta)\}$ as desired. □

This extends the case where skills i and j are linearly separable in the direction $\nabla\mathcal{L}(\theta_t; \mathcal{D}_{\boldsymbol{p}})$. It further provides insight in how to pick skills. To reduce any pairwise regret, the radii r_i and r_j of the clusters from their mean should be as small as possible in every direction.

B.1.3. OOD EVALUATION CLUSTERS

When performing k-means clustering, there is a choice in clustering the training points and then assigning the evaluation points, or clustering the evaluation points and then assigning the training points. The latter choice has a major problem: evaluation clusters may have no training points near them. This causes a major dilemma for the training procedure attempting to sample from a distribution that lacks any data points.

We adopt the former method of clustering based on the training points to circumvent this issue. However, a new issues arises: evaluation data may be OOD and have no representatives in the training data. The result is the label provided to those OOD points is the same as the closest training points. These can be quite distant and therefore not a strong representation of their gradient. However, this still is the optimal choice, as all other training points are a greater distance away and therefore have a weaker similarity. This label assignment also adds more weight to the class that is most aligned with the OOD evaluation data, increasing its sample rate to learn both the ID and OOD data for that class.

C. Algorithm details

We fully outline our algorithms for solving the optimization problems specified in Equation 2 and Equation 3, respectively. And we provide our Algorithm details in Algorithm 2 and Algorithm 3.

Algorithm 2 R&B Skill Partitioning

Input: Training data D_{train}, Evaluation data D_{eval}, Embedding model $\psi : \mathcal{X} \to \mathbb{R}^d$,
Clustering algorithm $cluster : \mathcal{P}(\mathbb{R}^d) \times \mathcal{K} \to \mathbb{N} \times (\mathbb{R}^d \to \mathbb{N})$,
Clustering metric $metric : (\mathbb{R}^d \to \mathbb{N}) \times \mathcal{P}(\mathbb{R}^d) \to \mathbb{R}$,
Range of clustering hyperparameters $K \in \mathcal{K}$
Output: Optimal number of clusters m^*, Optimal mapping function f^*, Partitioned datasets $\{D^*_{\text{train},i}\}_{i=1}^{m^*}$, $\{D^*_{\text{eval},i}\}_{i=1}^{m^*}$
$D_{\text{train}} \leftarrow \{\psi(x) : x \in D_{\text{train}}\}$ ▷ Collect embeddings for training data
$D_{\text{eval}} \leftarrow \{\psi(x) : x \in D_{\text{eval}}\}$ ▷ Collect embeddings for eval data
for $k \in K$ **do**
  $m, f = cluster(D_{\text{train}}, k)$
  $score = metric(f, D_{\text{train}})$
  $m^*, f^*, score^* = \arg\max_{m,f,score}(score, score^*)$
end for
for $i = 1$ to m^* **do**
  $D^*_{\text{train},i} \leftarrow \{x \in D_{\text{train}} : f^*(x) = i\}$ ▷ Partition training data
  $D^*_{\text{eval},i} \leftarrow \{y \in D_{\text{eval}} : f^*(y) = i\}$ ▷ Partition evaluation data
end for
Return $m^*, f^*, \{D^*_{\text{train},i}\}_{i=1}^{m^*}, \{D^*_{\text{eval},i}\}_{i=1}^{m^*}$

D. Compute cost models for online data mixing

We formalize a cost model for estimating the amount of compute required for several data mixing methods. Table 4 reports cost in terms of FLOPs, or number of floating point operations required to perform each method.

Following (Kaplan et al., 2020), we will use the estimate for the compute cost $C = C_{\text{forward}} + C_{\text{backward}} \approx 2ND + 4ND$, where N is the number of model parameters, and D is the number of training tokens. Here, we also make use of the empirical observation that the amount of compute for a backward pass is roughly twice that of the amount for a forward pass (Hobbhahn, 2021).

For all methods analyzed, we make the following assumptions:

- Each method trains on D_t tokens across m domains,
- Each method has access to an evaluation dataset with D_e tokens,
- Training is divided into T rounds with domain reweighting between rounds,
- Each method uses some fraction of the training dataset, $\delta \cdot D_t$ (where $\delta < 1$), to perform their reweighting procedure.

For our analysis, it is necessary to split the forward and backward compute costs because the data mixing algorithms we study involve a domain-reweighting mechanism that requires model evaluation on a hold-out dataset. Model evaluation

Algorithm 3 R&B Online Data Selection

Input: Partitioned datasets $\{D^*_{\text{train},i}\}_{i=1}^{m^*}$, $\{D^*_{\text{eval},i}\}_{i=1}^{m^*}$, model parameters $\theta \in \mathbb{R}^n$, training rounds T, steps per round K, evaluation proportions $\boldsymbol{p}' \in \mathbb{R}^{m^*}$
Initialize sampling distribution $\boldsymbol{p}^0 = \text{Uniform}(m^*)$
for $t = 0, \ldots, T-1$ **do**
  for $i \in [m^*]$ **do**
    $\nabla\mathcal{L}(\theta; D_i) \leftarrow \mathbf{0}_n$ ▷ Initialize gradient accumulator for domain i
    $\mathcal{S}_i \leftarrow \emptyset$ ▷ Initialize set of samples from domain i
  end for
  for $k = 0, \ldots, K-1$ **do**
    Sample batch $\mathcal{D} = \{x_j\}_{j=1}^B$ where $x_j \sim D^*_{\text{train},i}$ with $i \sim \boldsymbol{p}^t$ for each j
    $\theta_{tK+k+1} \leftarrow \theta_{tK+k} - \eta\nabla\mathcal{L}(\theta_{tK+k}; \mathcal{D})$
    for $i \in \{f^*(d) : d \in \mathcal{D}\}$ **do**
      $\nabla\mathcal{L}(\theta; D_i) \leftarrow \nabla\mathcal{L}(\theta; D_i) + \nabla\mathcal{L}(\theta_{tK+k}; \mathcal{D} \cap D^*_{\text{train},i})$
      $\mathcal{S}_i \leftarrow \mathcal{S}_i \cup (\mathcal{D} \cap D^*_{\text{train},i})$
    end for
  end for
  Construct $G \in \mathbb{R}^{m^* \times m^*}$ where $G_{ij} = \frac{1}{|\mathcal{S}_i||\mathcal{S}_j|}\nabla\mathcal{L}(\theta; D_i)^T\nabla\mathcal{L}(\theta; D_j)$
  $\boldsymbol{p}^{t+1} \leftarrow \text{softmax}(\eta\lambda G\boldsymbol{p}')$
end for
Return

only requires a forward pass, whereas model training requires both a forward and backward pass. To illustrate this point, let D_{train} be the number of tokens in the training dataset, while D_{eval} is the number of tokens in the evaluation dataset. Training a model on all available training data has a total cost of $6ND_{train}$, while computing model evaluation once has a cost of $2ND_{eval}$.

D.1. Skill-It

Skill-It (Chen et al., 2023) has two stages in its data-mixing procedure: estimating a graph A which is used as part of its domain reweighting procedure, and training itself.

For learning A, a model is trained on each of m domains for some fraction of D_{train}, then evaluated on D_{eval}. For comparative purposes, we will assume that δD_{train} training tokens are used in this process of constructing A, for $\delta < 1$. Furthermore, we assume these tokens are divided evenly among each of m domains. Then the compute cost for learning A is $6(\delta D_{train})N + 2(mD_{eval})N$. Training is split into T rounds, and after each round, the model is re-evaluated on D_{eval} to update the domain proportions. The compute cost for training, then, is $6(D_{train})N + 2TD_{eval}N$. This brings the total compute cost to

$$6(1+\delta)D_{train}N + 2(T+m)D_{eval}N.$$

D.2. Aioli

Similar to Skill-It, Aioli (Chen et al., 2024) also includes two stages for learning A and training, but incorporates both directly into the training process. At a high level, training is also split into T rounds, where each round dedicates some fraction $\delta < 1$ to learning A. When learning A, the model is trained on each of m domains sequentially, and re-evaluating the resulting model on the evaluation dataset. Consequently, the training compute cost for learning A is simply absorbed into the overall cost for training, but the model still must be evaluated on D_{eval} for each domain. Within a round, this process repeats for the number of sweeps k, but here we will set $k = 1$ to simplify the analysis. Therefore, the total compute cost for training improves to $6(D_{train})N$, but the compute cost for evaluation increases to $2(Tm)D_{eval}N$. This brings the total compute cost to

$$6D_{train}N + 2(Tm)D_{eval}N.$$

Method	Total Compute Cost (FLOPs)	Relative Compute Overhead (vs. Standard Training)
Standard Training	$6D_tN$	0
Skill-It (Chen et al., 2023)	$6(1+m\delta)D_tN + 2(T+m)D_eN$	$m\delta + \frac{(T+m)D_e}{3D_t}$
Aioli (Chen et al., 2024)	$6D_tN + 2(Tm)D_eN$	$\frac{TmD_e}{3D_t}$
DGA (Fan et al., 2024a)	$6(1+m\delta)D_tN + 6T(\delta D_e)N$	$m\delta + T\delta\frac{D_e}{D_t}$
R&B (Ours)	$6D_tN + Tm^2N$	$\frac{Tm^2}{6D_t}$

Table 4. Computational cost comparison of data mixing methods. We report (1) total cost of training, given under the table Total Compute Cost, and relative compute overhead over standard training. Standard training requires no additional compute overhead since its proportions are fixed. In the common setting where the number of skills m is much smaller than that of evaluation tokens D_e and training tokens D_t, R&B enjoys superior computational efficiency.

D.3. DGA

Dynamic Gradient Alignment (Fan et al., 2024a) instead uses gradient information to reweight the domain proportions. Their method splits training into T rounds, and reweights proportions after each round. Their procedure involves sampling a batch from each domain, and then performing a forward and backward pass to obtain gradients respective of each domain. They then obtain gradients for a batch on a specific dataset D_{spe} (which for consistency of analysis we will simply refer to as D_{eval}), and computes the inner product between the gradients of each domain and that of D_{eval}. In order to equalize model performance with Skill-It and Aioli, we will assume that a batch from each training domain contains $\frac{\delta}{m}D_{train}$ tokens, and a batch from the specific dataset contains D_{eval} tokens. Then, computing each domain's gradient has a cost of $6(\frac{\delta}{m})D_{train}N$, and computing the specific dataset's gradient has a cost of $6D_{eval}N$. We assume that computing the inner product between two model gradients is linear in N so there is an additional mN compute overhead. Therefore, the total compute cost is

$$6(1+\delta)D_{train}N + 6T(D_{eval}+m)N.$$

D.4. R&B (ours)

Similar to all above methods, we split training into T rounds, and reweight domain proportions at the end of each round. Like Dynamic Gradient Alignment, we opt to use gradient inner product information to inform our reweighting procedure. Crucially, however, we make two observations: (1) gradients per domain can be collected on the fly during normal backpropagation, and (2) our optimization problem only requires knowledge about the respective proportions of D_{eval}, and does not use gradient or loss information about D_{eval}. Instead, we simply compute the equivalent of matrix A which is a Gram matrix comprised of the inner products between the gradients of each respective domain. As a result, the compute cost of training our method is simply $6(D_{train})N$, and the compute cost of evaluation is just m^2N. Therefore, the total compute cost is

$$6(D_{train})N + m^2N.$$

Efficiency Analysis. Under typical conditions where the number of skills is much smaller than the size of the evaluation dataset, **R&B demonstrates superior computational efficiency**. Its evaluation overhead scales only with m^2 rather than with D_e, making it particularly advantageous for scenarios with large evaluation datasets but a moderate number of domains.

When comparing specifically with DGA, R&B's advantage depends on the relationship between the number of domains and evaluation data size. R&B is more efficient when $m < \sqrt{D_e}$, which holds in most practical settings. Even when m approaches or exceeds $\sqrt{D_e}$, R&B maintains partial efficiency benefits through its 6× lower coefficient on the evaluation term, and by avoiding the additional δ fraction for computing gradients.

E. Implementation Details

We start with an explanation of gradient computations.

E.1. Efficient Gradient Computation

Standard training pipelines provide per-batch gradients, but we need per-example gradients in order to aggregate per-skill gradients for our method. We perform a gradient decomposition similar to the method introduced in (Goodfellow, 2015) to efficiently circumvent this. A simple application of the chain-rule means we can exactly recover per-example gradients of a linear layer in a mini-batch with just one backwards pass by multiplying an example's input with that mini-batch's gradient.

Adopting the notational convention from (Wang et al., 2024), let $\mathbf{s} = \mathbf{a}\mathbf{W}$ be a linear layer where $\mathbf{W} \in \mathbb{R}^{d_1 \times d_2}$ is a weight matrix, $\mathbf{a} = (\mathbf{a}^{(1)}, \ldots, \mathbf{a}^{(B)})^\top \in \mathbb{R}^{B \times d_1}$ is the input to the mini-batch, and $\mathbf{s} = (\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(B)})^\top \in \mathbb{R}^{B \times d_2}$ is the layer's pre-activation output. Denote by $\ell^{(i)}$ the loss on the ith example in the mini-batch. Let ℓ denote the summed loss of the mini-batch. It follows from the chain rule that the gradient of $\ell^{(i)}$ with respect to $\mathbf{W}$ can be expressed as

$$\frac{\partial \ell^{(i)}}{\partial \mathbf{W}} = \frac{\partial \ell^{(i)}}{\partial s^{(i)}} \frac{\partial s^{(i)}}{\partial \mathbf{W}} = \frac{\partial \ell^{(i)}}{\partial s^{(i)}} a^{(i)} = \frac{\partial \ell}{\partial s^{(i)}} a^{(i)},$$

where the last equality follows from the fact that the $\frac{\partial \ell^{(j)}}{\partial s^{(i)}}$ terms disappear when $i \neq j$. Notably, the $\frac{\partial \ell}{\partial s^{(i)}}$ term is available through standard training, and $a^{(i)}$ can be easily tracked. We aggregate per-example gradients into their respective skills, allowing for efficient per-skill gradient computation.

F. Experimental Details

We evaluate our method, R&B, against four baseline data mixing methods: Stratified sampling, Skill-It, Aioli, and DGA (Dynamic Gradient Alignment). We conducted experiments on three datasets of varying sizes and characteristics.

F.1. Datasets

- DOLLY-15K: An instruction follow-up dataset consisting of 15,000 examples with eight original skill categories.
- SUP-NATINST (Natural Instructions In-Domain): A 285k dataset created from Natural Instructions by selecting 100 tasks out of 876 available tasks containing 38 original skill categories.
- SUP-NATINST-Test (Natural Instructions Out-of-Domain): A 3.56M dataset created from Natural Instructions with questions and answers from domains not seen SUP-NATINST, containing 60 original skill categories.

For in-distribution datasets (NI-ID and Dolly-15k), we use 90% of the total dataset for training and 5% for testing. For regrouping experiments, we generate embeddings using ModernBERT with a dimension of 786 and cluster the datasets using k-means.

F.2. Experimental Configurations

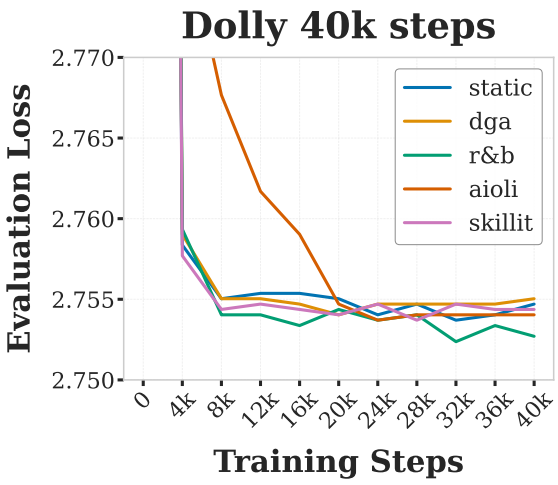
Table 5. Experimental Settings Across Different Datasets

Parameter	**Dolly-15k, NI-ID, NI-OOD**	**S1-59k**
Model	GPT-Neo 125M	Qwen2-0.5B
Training batch size	16	4
Evaluation batch size	16	16
Context Length	512	8192
Learning rate	5e-5	1e-5
Optimizer	AdamW	AdamW

G. Extended Training Results

In this appendix, we provide additional experimental results for training on the Dolly-15k dataset for an extended period of 40,000 steps. This allows us to understand the long-term behavior of R&B compared to different data mixing methods. Figure 4 shows the training loss curves for different data mixing methods over the full 40,000 steps (left) alongside the final evaluation loss after 40,000 steps (right). R&B maintains consistent performance over other data mixing methods, demonstrating the stability of our approach and achieving the best performance with a loss of 2.723.

For this extended training experiment, we focus on the original category partitioning (rather than our regrouping approach)



Method	Evaluation Loss
Stratified	2.733
DGA	2.733
Aioli	2.724
Skill-It	2.728
R&B (ours)	**2.723**

Figure 4. Left: Training loss curves for Dolly-15k trained for 40,000 steps with different data mixing methods using the original category partitioning. Right: Average test loss on Dolly-15k after 40,000 training steps using original category partitioning. **Highlighted** values (with brown background) indicate the best overall performance.

to demonstrate R&B's effectiveness even with pre-defined categories when given sufficient training time. We observe that all data mixing methods eventually converge to similar performance levels after sufficient training, but R&B maintains a consistent advantage throughout the training process. This suggests that our gradient-based approach effectively captures the optimal training dynamics from early stages, leading to more efficient parameter updates throughout the training process.

Furthermore, we study how R&B reweights proportions over time. As illustrated in Figure 5, our method employs a dynamic approach to domain importance throughout the training process. Initially, domain weights fluctuate significantly as the model explores the contribution of each domain to overall performance. By the midpoint of training (around steps 40-60), a clear pattern emerges with Domains 1 and 5 receiving substantially higher weights (reaching approximately 0.225) compared to other domains. Notably, while these weights gradually trend toward the evaluation distribution proportions shown in Figure 6, they never completely converge to match the actual evaluation proportions. For instance, Domain 1 maintains a training weight of around 0.225 even though its evaluation proportion is higher at 53.6%, and Domain 5 stabilizes at approximately 0.200 despite its 29.3% evaluation proportion. This deliberate partial convergence suggests that optimal performance requires a strategic balance—influenced by but not identical to the evaluation distribution.

H. Clustering Interpretation

In this section, we provide some interpretation about the groups discovered via clustering.

Table 8 shows the difference in groups before and after clustering on the Dolly-15k dataset. The left column displays the initial eight categories used to organize the text dataset during collection: brainstorming, classification, closed QA, generation, information extraction, open QA, summarization, and creative writing. The right column shows the seven distinct clusters that emerged when applying our clustering algorithm to the entire corpus. Interestingly, rather than following the original task-based boundaries, these clusters primarily organized around content domains, and subject matter, and subject length. This suggests that semantic content features may be more salient for learning features than the original task-based categorization framework, potentially offering new insights into how language models naturally organize information.

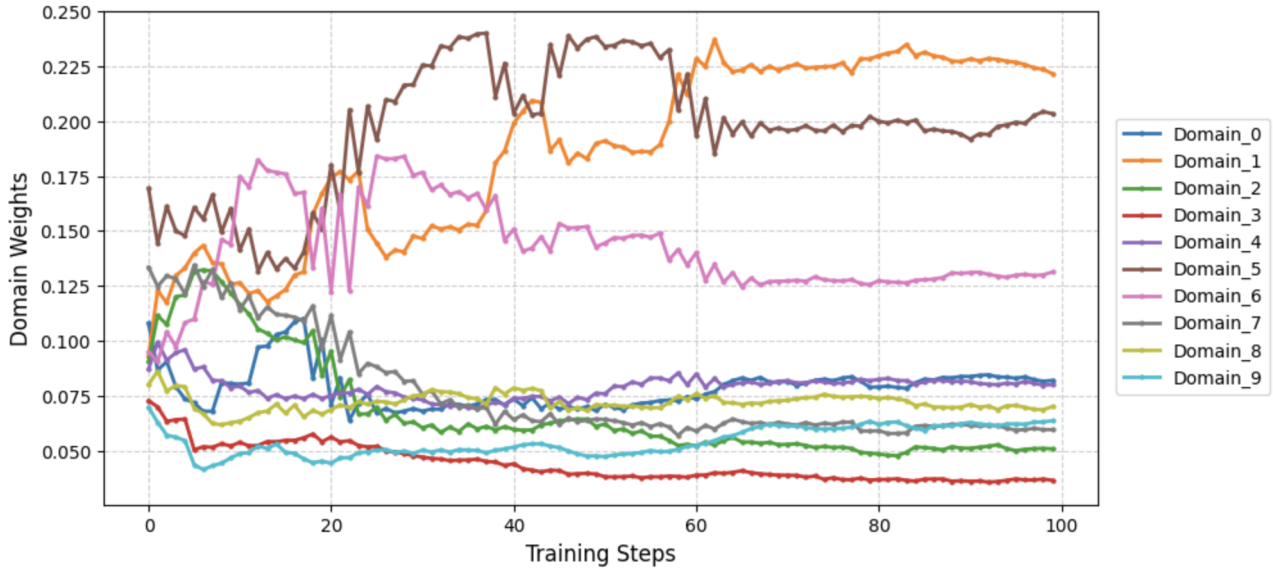


Figure 5. Domain weight evolution during training. Our method dynamically adjusts the importance of each domain throughout the training process, with Domains 1 and 5 eventually receiving the highest weights while Domains 0, 2, 3, 7, 8, and 9 are downweighted over time.

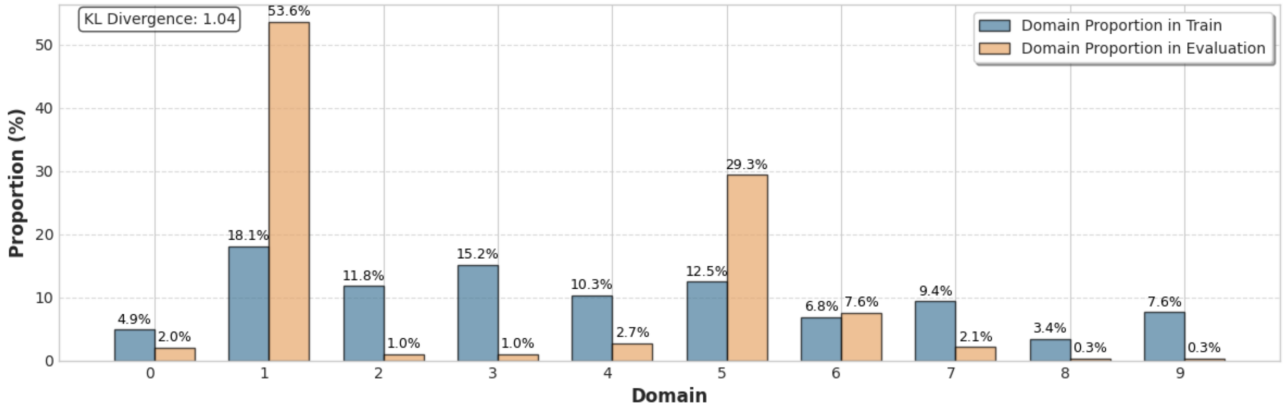


Figure 6. Comparison between domain proportions in training versus evaluation data (KL Divergence: 1.04). Our method strategically reweights domain distributions during training to optimize performance, notably increasing the representation of Domains 1 and 5 while reducing emphasis on Domains 2, 3, 4, 7, 8, and 9 compared to their evaluation proportions.

Table 6. Training budget allocations and experimental settings for all datasets and methods.

Dataset	Domains	Method	Training Steps	Method-Specific Settings
Dolly-15k	Original (8)	Stratified	2000	full eval dataset
		Skill-It	2000	200 steps for graph estimation, full eval dataset
		Aioli	2000	rounds=2, sweeps=1, full eval dataset
		DGA	2000	full eval dataset
		R&B	2000	full eval dataset
	Regrouped (7)	Stratified	2000	full eval dataset
		Skill-It	2000	200 steps for graph estimation, full eval dataset
		Aioli	2000	rounds=2, sweeps=1, full eval dataset
		DGA	2000	full eval dataset
		R&B	2000	full eval dataset
NI-ID	Original (38)	Stratified	2000	full eval dataset
		Skill-It	2000	200 steps for graph estimation, full eval dataset
		Aioli	2000	rounds=2, sweeps=1, full eval dataset
		DGA	2000	full eval dataset
		R&B	2000	full eval dataset
	Regrouped (30)	Stratified	2000	full eval dataset
		Skill-It	2000	200 steps for graph estimation, full eval dataset
		Aioli	2000	rounds=2, sweeps=1, full eval dataset
		DGA	2000	full eval dataset
		R&B	2000	full eval dataset
NI-OOD	Original (60)	Stratified	2000	full eval dataset
		Skill-It	2000	200 steps for graph, full eval dataset
		Aioli	2000	rounds=1, sweeps=1, 50k eval samples
		DGA	2000	full eval dataset
		R&B	-	-
	Regrouped (100)	Stratified	2000	full eval dataset
		Skill-It	1000	25 steps for graph, 10k eval samples
		Aioli	1000	rounds=1, sweeps=1, 50k eval samples
		DGA	2000	full eval dataset
		R&B	2000	full eval dataset

* Note: There is no result for R&B in the NI-OOD original column because the method requires finding training skills in the evaluation dataset. In out-of-domain settings, test skills and train skills are different, causing the Gp_norm in R&B to be NaN.

Table 7. Training budget allocations and experimental settings for S1-59K dataset.

Dataset	Domains	Method	Training Steps	Method-Specific Settings
S1-59K	Original (6)	Stratified	500	full eval dataset
		R&B	500	num_layers_to_track=1, lamda=3, full eval dataset
	Regrouped (10)	Stratified	500	full eval dataset
		R&B	500	num_layers_to_track=1, lamda=3, full eval dataset

Original Category	Regrouped Cluster
brainstorming	Cluster 0: General knowledge and open-ended questions covering a wide range of topics from science, technology, to basic concepts
classification	Cluster 1: Music-related queries focusing on instrument classification, musical theory, and instrument comparisons
closed QA	Cluster 2: Information extraction and summarization tasks about various topics including companies, historical figures, and specific domains
generation	Cluster 3: Classification tasks primarily involving animals, colors, household items, and biological categorizations
information extraction	Cluster 4: Sports-related queries spanning multiple disciplines including golf, F1 racing, Olympics, and team sports
open QA	Cluster 5: Entertainment and pop culture queries about movies, TV shows, musicians, artists, and historical personalities
summarization	Cluster 6: Lifestyle and creative brainstorming queries covering diverse topics from home improvements to personal recommendations
creative writing	

Table 8. Mapping between original categories and regrouped clusters