REGMIX-D: Dynamic Data Mixing via Proxy Training Trajectories

Kaiyan Zhao[1,2], Zhongtao Miao[1], Akiko Aizawa[2], Yoshimasa Tsuruoka[1]
[1]The University of Tokyo, [2]National Institute of Informatics
{kaiyan1006, miao, tsuruoka}@logos.t.u-tokyo.ac.jp, aizawa@nii.ac.jp

Abstract

Data mixture selection is critical for Large Language Model pretraining. Existing methods such as RegMix select a single static mixture by fitting a regression model on small-scale proxy runs. We propose REGMIX-D, a simple extension of RegMix to dynamic mixing. Our key observation is that proxy runs produce not only endpoint losses, but also full loss trajectories, which can be used to further improve data mixture. By training regression model on these trajectories, we can predict optimal mixtures at multiple training stages. REGMIX-D supports two deployment modes: an *offline* variant that generates a complete mixture schedule before target training, and an *online* variant that adapts the mixture during training using observed loss. Experiments on 25B tokens of the Pile dataset with a 1B parameter target model show that REGMIX-D consistently improves over baselines across 13 downstream tasks, and the improvement carries over to a 7B target model on 100B tokens. REGMIX-D also remains proxy-efficient: it surpasses RegMix even with only 128 proxy models (25% of RegMix's proxy compute budget).

1 Introduction

Pretraining Large Language Models (LLMs) requires assembling massive corpora from diverse domains (Belenki et al., 2025; Chen et al., 2026). The mixture proportions among these domains substantially affect downstream performance, making data mixture selection a critical design choice (Feng et al., 2024). Recent work has proposed automated methods to optimize these proportions, including DoReMi (Xie et al., 2023), which uses group DRO (Oren et al., 2019) over a reference model, and RegMix (Liu et al., 2025), which trains a regression model on a set of small proxy models to predict loss as a function of mixture.

While effective, these methods share a common assumption: a single optimal mixture exists and can be used throughout pretraining. However, the optimal composition of different domains may shift during training (Mo et al., 2025). Under this view, static mixtures necessarily compromise across training, leading to suboptimal performance.

Several recent methods adopt dynamic mixing to address this limitation (Chen et al., 2025; Wang et al., 2025; Ma et al., 2025). However, these methods introduce additional overhead on top of standard training. See Section 2 for discussion.

To this end, we propose REGMIX-D, a simple extension of RegMix to dynamic mixing. Our key observation is that the loss trajectories collected during proxy runs already contain the information needed for dynamic prediction: each proxy run produces a full loss curve, not just an endpoint. We train regression models that take the current training state: (t, m, ℓ), where t is the proxy step, m is the current mixture, and ℓ is the current loss, to predict the loss at the next sampled step. This model can be deployed in two modes (Figure 1): an *offline* mode that recursively generates mixture schedules before target training begins, and an *online* mode that queries the regression model during training and adapts the mixture in place. Neither mode introduces additional optimization mechanisms.

We evaluate REGMIX-D on 25B tokens of the Pile dataset (Gao et al., 2020) with a 1B parameter target model, following the standard RegMix setup. Our contributions are as follows:

- We introduce REGMIX-D, a dynamic extension of RegMix with offline and online variants, requiring no additional training machinery beyond the existing regression framework.
- Across all tested granularities, REGMIX-D consistently outperforms RegMix on validation loss and on the average of 13 downstream tasks, with gains that are significant over 5 seeds and hold at 7B scale.

Findings of the Association for Computational Linguistics: IJCNLP-AACL 2026, pages 1698–1707
November 6-10, 2026 ©2026 Association for Computational Linguistics

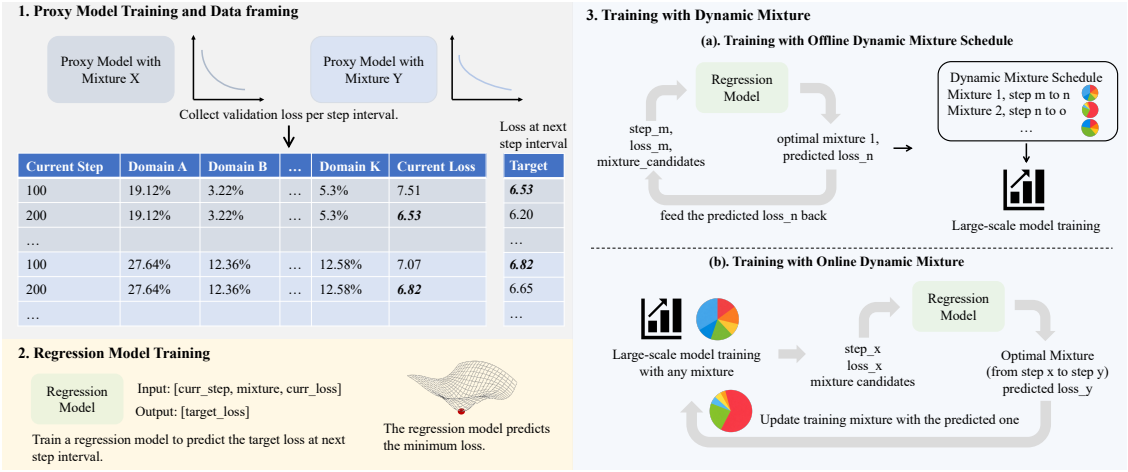


Figure 1: Overview of REGMIX-D. We train a regression model f on proxy loss trajectories (left), then deploy f in two modes (right): *Offline* recursively generates a complete mixture schedule before target training; *Online* queries f during target training using observed losses to adapt the mixture in place.

- REGMIX-D is proxy-efficient: 128 proxy models (25% of RegMix's compute budget) suffice to surpass RegMix with 512 proxy models.

2 Related Work

Static Data Mixing. Conventional methods select a single mixture for the whole training. DoReMi (Xie et al., 2023) uses group DRO with a reference model to upweight domains with high excess loss. DoGE (Fan et al., 2024) reweights domains based on gradient-level generalization contribution. RegMix (Liu et al., 2025) trains small proxy models with diverse mixtures and fits a regression model that predicts validation loss from mixture candidates, then selects the optimal mixture by search. An alternative line of work selects mixtures via weighted model merging rather than direct proxy training (Li et al., 2026; Wang et al., 2026). REGMIX-D builds directly on RegMix's regression framework, generalizing single-mixture selection to time-varying schedules.

Dynamic Data Mixing. Several recent methods adopt time-varying mixtures. Aioli (Chen et al., 2025) estimates mixing-law parameters online during target training and updates proportions via exponentiated gradient descent. TiKMiX (Wang et al., 2025) reweights domains using group gradient influence on a target distribution. AC-ODM (Ma et al., 2025) trains an actor-critic network that provides an online reweighting signal during target training. REGMIX-D differs from them along three axes: (i) *No additional machinery during target training*: existing dynamic methods introduce online optimization, gradient-influence computation, or RL queries on top of target training, whereas REGMIX-D confines all optimization to the proxy phase. (ii) *Reuses RegMix's signal*: we predict from proxy loss trajectories already produced in the RegMix pipeline, requiring no new instrumentation. (iii) *Two deployment modes*: a single regression model supports both offline and online adaptation.

3 REGMIX-D

3.1 Preliminaries

RegMix (Liu et al., 2025) selects a single static data mixture by training M small *proxy* models (one per sampled mixture) for a fixed number of steps T_p, recording the validation loss at the final step, and fitting a regression model that maps a mixture to its predicted validation loss. The optimal mixture is then identified by searching over candidate mixtures and selecting those with the lowest predicted loss. The *target* model is subsequently trained from scratch with this single mixture for the entire run. This discards two sources of information in proxy runs: (i) intermediate losses along each proxy trajectory, and (ii) how the optimal mixture might evolve across training.

3.2 Trajectory-Conditioned Regression

We log validation loss at multiple step intervals during training instead of only the endpoint loss. Given a desired number of switch points N (at which the mixture would change), we select $N+$

1 intervals $\{t_1, t_2, \ldots, t_{N+1}\}$ from the proxy run. For each proxy run with mixture $m^{(i)}$, this yields a trajectory $\{\ell_1^{(i)}, \ldots, \ell_{N+1}^{(i)}\}$ of validation losses.

We construct training pairs for a new regression model f_N over consecutive intervals as shown in Figure 1, Part 1. For each proxy i and each adjacent pair $(t_\ell, t_{\ell+1})$, we form one training example:

$$\underbrace{\left(t_\ell,\ m^{(i)},\ \ell_\ell^{(i)}\right)}_{\text{input}} \longrightarrow \underbrace{\ell_{\ell+1}^{(i)}}_{\text{target}}. \tag{1}$$

That is, f_N learns to predict the validation loss at the next step interval, conditioned on (a) the current proxy step t_ℓ, (b) the current mixture $m^{(i)}$, and (c) the current observed loss $\ell_\ell^{(i)}$. This produces $M \times N$ training pairs in total. Crucially, f_N predicts a *local* transition $(t_\ell, m, \ell_\ell) \to \ell_{\ell+1}$ rather than a global mixture-to-endpoint map. This locality is what enables dynamic scheduling: at any training state, we can ask f_N which mixture is expected to drive the loss lowest in the next interval. Note that f_N is specific to a given N, since the training pairs depend on the interval numbers.

3.3 Offline Mixture Schedule (Algorithm 1)

Given the regression model f_N trained at the desired N, we can generate mixture schedules with N switch points $\{s_1, s_2, \ldots, s_N\}$, corresponding to the step intervals used to train f_N. The schedule divides target training into $N+1$ segments. The first segment uses the human-curated Pile prior m_0^{human}, reflecting the assumption that no prior knowledge about the target corpus is available at the start of training, and each subsequent segment uses a regression-predicted mixture.

Starting from an initial loss ℓ_0, we iterate over switch points $s_1, \ldots, s_N$. At each switch point s_j, we search for the mixture m_j^* minimizing $f_N(s_j, m, \ell_{j-1})$ over Dirichlet-sampled candidates, record (s_j, m_j^*) as one schedule entry, and update $\ell_j \leftarrow f_N(s_j, m_j^*, \ell_{j-1})$ as the input loss for the next iteration. The novelty lies in how f_N is queried across the N switch points: predictions are made *recursively*, with each prediction's output loss fed into the next prediction as input. The final schedule $\{(s_j, m_j^*)\}_{j=1}^N$ is mapped to target training via $s_j^{\text{target}} = s_j \cdot (T_{\text{target}}/T_p)$, where T_{target} stands for the training steps of the target model.

3.4 Online Dynamic Mixture (Algorithm 2)

The recursive use of predictions in offline mode may iteratively accumulate error. The online variant replaces predicted losses with *observed* target-model losses at each switch point, grounding subsequent mixture decisions during target training.

Concretely, the same regression model f_N trained on proxy trajectories is loaded as a frozen predictor at the start of target training. At each switch point $S_j = s_j \cdot (T_{\text{target}}/T_p)$, we query f_N using the validation loss $\hat{\ell}_{S_j}$ measured on the target model, then select the next segment's mixture via the same Dirichlet-sampling and top-k averaging used in the offline variant.

Cross-scale loss correction. Because f_N is trained on proxy-scale losses (proxy model size P_p) while target losses come from a much larger model (size P_t), the two are not directly comparable. We apply a power-law correction to map target losses to proxy scale before invoking f_N (Kaplan et al., 2020):

$$\tilde{\ell}_{S_j} \;=\; \hat{\ell}_{S_j} \cdot \left(P_t/P_p\right)^{\beta}, \tag{2}$$

where β is a small positive constant. In our setup $P_t/P_p = 1000$, and we use $\beta = 0.05$, selected from a sweep over $\beta \in \{0.01, 0.02, 0.05, 0.10\}$; final downstream performance varies by at most 0.25 across this range (Table 2), indicating that online adaptation is robust to the precise choice of β. The query at switch point S_j thus becomes $f_N(s_j,\ m,\ \tilde{\ell}_{S_j})$.

4 Experiments

4.1 Experimental Setup

Following Liu et al. (2025), we use the 1B TinyLlama (Zhang et al., 2024) trained on 25B Pile (Gao et al., 2020) dataset (17 domains) as the target model and a 1M parameter TinyLlama variant as the proxy model in the main experiments. We compare against three static baselines: Human, DoReMi (Xie et al., 2023), and RegMix (Liu et al., 2025), and one dynamic baseline, AC-ODM (Ma et al., 2025), reproduced with the authors' official code under our exact setting. We report **RegMix-D (128)** and **RegMix-D (512)** and two deployment variants for each: **offline** and **online** with $N = 5$ switch points in main experiments. Average results for 13 tasks from `lm-eval-harness`[1] (Gao et al., 2024) are reported. More details can be found in Appendix A.1.

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

Task	Human	DoReMi	AC-ODM†	RegMix	RegMix-D (128)		RegMix-D (512)	
					Offline	Online	Offline	Online
HellaSwag	37.93	43.01	43.58	43.78±0.4	**45.76**±0.4	45.69±0.3	45.11±0.3	45.74±0.2
PIQA	64.99	68.05	68.74	68.71±0.2	69.00±0.5	69.50±0.4	69.20±0.4	**69.80**±0.3
OpenBookQA	29.17	30.70	31.12	30.80±0.7	30.33±0.4	31.26±0.4	**31.99**±0.1	31.87±0.3
LAMBADA	27.09	31.58	32.45	32.78±0.2	31.94±0.6	**34.55**±0.2	34.43±0.2	34.49±0.4
SciQ	79.67	81.42	81.87	81.20±0.2	83.11±0.2	**83.58**±0.3	82.58±0.1	83.40±0.3
ARC-Easy	49.06	50.36	51.70	50.84±0.5	53.62±0.1	53.99±0.5	53.56±0.2	**54.19**±0.2
ARC-Challenge	25.61	25.78	26.76	26.20±0.3	**27.64**±0.2	26.98±0.6	26.72±0.3	27.49±0.2
COPA	65.00	67.50	69.00	68.00±0.4	70.00±0.4	69.50±0.3	71.00±0.1	**71.31**±0.3
RACE	29.87	31.07	**32.74**	30.60±0.1	31.08±0.3	30.87±0.2	31.88±0.2	32.10±0.2
LogiQA	25.47	26.93	**27.85**	26.71±0.6	26.42±0.3	25.88±0.3	25.55±0.3	25.02±0.5
QQP	42.48	50.50	51.94	**52.23**±0.3	50.96±0.6	51.69±0.3	50.11±0.4	51.06±0.3
WinoGrande	51.87	52.09	53.29	52.71±0.4	**53.77**±0.4	52.08±0.5	52.30±0.5	52.33±0.4
MultiRC	54.15	53.31	**54.21**	52.07±0.4	50.43±0.3	51.77±0.6	52.15±0.2	51.98±0.3
Avg	44.80	47.10	48.09	47.43±0.3	48.00±0.2*	48.25±0.1*	48.19±0.3*	**48.52**±0.2*

Table 1: Downstream performance on 13 tasks, averaged over 0-5 shot settings. All methods train a 1B target model on 25B tokens. RegMix and RegMix-D report mean±std over 5 seeds; RegMix-D variants use $N = 5$ switch points. *: significantly better than RegMix on Avg (paired t-test over 5 seeds, $p < 0.05$). †: reproduced with the official code under our setup, single run. **Bold** and underline indicate the best/second best result per row. RegMix-D (128) uses 128 proxy models, approximately 25% of RegMix's 512-proxy compute budget.

4.2 Main Results

Table 1 reports downstream performance, where REGMIX-D consistently outperforms baselines at both proxy budgets. All four REGMIX-D variants improve over RegMix on average, and each improvement is statistically significant over 5 seeds. Notably, even with only 128 proxies, 25% of the proxy compute used by RegMix, both REGMIX-D variants still surpass RegMix in average score. Figure 2 shows the pile-cc validation loss across training: all four REGMIX-D variants achieve lower loss than RegMix after the initial Human-mixture segment. The corresponding dynamic mixtures differ substantially from RegMix's static mixture, and we visualize the full 17-domain schedules in Appendix A.4.

Comparison with a dynamic baseline. AC-ODM improves over RegMix, confirming that dynamic mixing is beneficial in this setting. REGMIX-D (512) online still attains the highest average, and REGMIX-D (128) online also exceeds AC-ODM, while requiring only lightweight regression queries at switch points, whereas AC-ODM maintains a learned actor-critic controller during target training.

Online generally outperforms offline. Across both proxy budgets, the online variant achieves higher average accuracy than the offline variant. We attribute this to the gap between proxy and target dynamics: offline schedules are generated entirely from proxy loss trajectories, while online predictions condition on the actual target model's observed loss at each switch point. This grounding in real target dynamics provides a more accurate signal for downstream performances.

Improvements are broad rather than concentrated. REGMIX-D variants achieve top-1 or top-2 performance on 10 of 13 tasks. We do not observe a single task category (e.g., commonsense, reading comprehension) where REGMIX-D systematically wins or loses, suggesting the gains stem from a generally better mixture rather than implicit specialization toward particular tasks.

Compute scaling helps modestly. Increasing proxies from 128 to 512 yields a smaller average gain. This suggests that the dynamic-vs-static gap dominates the proxy-count gap in our setup: RegMix-D extracts more signal from each proxy run than RegMix does, by exploiting the full loss trajectory rather than a single endpoint.

4.3 Scaling to 7B

To test whether the gains persist at larger scale, we train a 7B target model on 100B Pile tokens (7× the model size and 4× the token budget of the main setting; single run) using RegMix-D (128) with the same proxies, regression model, and β as in the 1B setting. Both variants again outperform RegMix on average (55.91 → 56.70 offline, 56.94 online), with the online variant outperforming RegMix on 12 of 13 tasks (Table 5, Appendix). The absolute gain (+1.03 online) is comparable to that at 1B (+0.82 for the same configuration), suggesting that the benefit of dynamic scheduling does not diminish with scale in this range.

(a) Number of switch points N

	128 proxies		512 proxies	
N	Off.	On.	Off.	On.
3	47.47	48.13	47.55	48.01
5	**47.93**	**48.15**	**48.09**	**48.44**
7	47.78	47.74	47.84	47.98
9	47.58	47.82	48.01	48.40

(b) Scaling factor β

β	128	512
0.01	48.08	48.24
0.02	**48.24**	48.42
0.05	48.15	**48.44**
0.10	47.99	48.27

(c) Proxy size

Size	Offline	Online
1M	47.93	48.15
60M	**47.97**	48.14
120M	47.93	**48.19**

Table 2: Ablations on REGMIX-D (downstream Avg, %, single-seed run). All configurations exceed the RegMix baseline (47.41). **(a)** Number of switch points N at the 128- / 512-proxy setting. **(b)** Cross-scale loss correction factor β for the Online variant; maximum variation is 0.25. We use β=0.05 in main experiments as default. **(c)** Proxy model size analysis at the 128-proxy setting. **Bold**: best per column.

4.4 Ablation Study

Table 2 reports ablations on the three main design choices of REGMIX-D. Ablations are single-seed runs and are therefore compared against the single-seed RegMix run. **All 26 configurations across the three studies outperform the RegMix baseline (47.41)**, indicating that the benefit of dynamic mixing is robust to specific hyperparameter choices rather than reliant on careful tuning. Three more specific findings follow. First, the number of switch points N has a moderate but consistent effect: N=5 is best in all four proxy/variant settings, while N=3, 7, 9 also reliably surpass RegMix. Second, the cross-scale loss correction factor β has limited influence on the Online variant, with maximum Avg variation of 0.25. This robustness justifies treating β as a fixed constant rather than a tuned hyperparameter. Third, increasing the proxy model size from 1M to 120M yields virtually identical performance (within 0.05 Avg for both variants), a consistent finding with Liu et al. (2025).

Is a single switch enough? To separate the effect of introducing *any* adaptation from that of *multi-point* scheduling, we train online variants with a single switch point ($N = 1$) (Table 3). With a single switch, performance falls below RegMix regardless of where the switch is placed, because training starts from the relatively suboptimal Human mixture. More importantly, even when the first segment uses the RegMix-optimized static mixture, which removes any disadvantage from a poor initial distribution, one switch yields only a marginal gain (+0.36), whereas $N = 5$ yields +1.09. This attributes the improvement to multi-point scheduling rather than to one-off adaptation. Note that the RegMix-initialized variant violates our working assumption of no prior knowledge of domain quality and is included only as an analysis anchor.

Method	Avg
RegMix (static)	47.43
RegMix-D, $N = 1$, switch at 50%	45.98
RegMix-D, $N = 1$, switch at 10%	46.67
RegMix-D, $N = 1$, switch at 50%, RegMix init.	47.79
RegMix-D, $N = 5$ (default)	**48.52**

Table 3: Single-switch ablation (online, 512 proxies, 1B target). "RegMix init." starts from the RegMix static mixture instead of the Human mixture. RegMix and $N = 5$ are 5-seed means; $N = 1$ rows are single runs.

We discuss the overhead of REGMIX-D in Appendix A.2.

5 Conclusions

We present REGMIX-D, a simple regression-based extension of REGMIX that generates dynamic data mixture ratios, with both offline and online variants and no additional optimization machinery. On the Pile dataset with a 1B-parameter target model, REGMIX-D improves over REGMIX, DoReMi, and AC-ODM across 13 downstream tasks, and surpasses REGMIX even with 25% of its proxy compute. The improvements over REGMIX are significant across 5 seeds and persist at 7B scale. Ablation studies show the gains stem from multi-point scheduling rather than a single mixture switch, and are robust to switch-point count ($N \geq 3$), scaling factor, and proxy size, indicating that effective dynamic mixing is achievable at low cost.

Limitations

REGMIX-D has several limitations that motivate future work. First, our experiments use a single target-domain validation loss (pile-cc) as the optimization signal, following RegMix's setup. Extending REGMIX-D to multi-target or domain-agnostic objectives requires a non-trivial extension of the regression framework and is left to future work. Sec-

ond, among recent dynamic data mixing methods, we compare against AC-ODM, which we reproduced with its official code under our exact setup. Other methods such as TiKMiX report strong results in their respective settings, but they use different corpora (e.g., SlimPajama vs. Pile), domain partitions, target model scales, and benchmark suites, and to our knowledge no official implementation is publicly available, so we could not include them in a controlled comparison. Third, due to compute constraints, the AC-ODM reproduction and the 7B/100B-token experiment are single runs, and we have not tested target models beyond 7B or corpora other than the Pile.

Ethical Statements

This work studies data mixture selection for language model pretraining. We use the Pile (Gao et al., 2020), a publicly released corpus, and follow its existing domain partitioning. REGMIX-D does not introduce new training data, generation capabilities, or downstream applications beyond those already present in standard pretraining pipelines, and we therefore do not foresee direct ethical risks specific to our method. We use LLMs only for language and grammar polishing.

Acknowledgment

Kaiyan Zhao was supported by JST SPRING, Grant Number JPMJSP2108.

References

Lior Belenki, Alekh Agarwal, Tianze Shi, and Kristina Toutanova. 2025. Optimizing pre-training data mixtures with mixtures of data expert models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 32570–32587, Vienna, Austria. Association for Computational Linguistics.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*.

Mayee F Chen, Michael Y. Hu, Nicholas Lourie, Kyunghyun Cho, and Christopher Re. 2025. Aioli: A unified optimization framework for language model data mixing. In *The Thirteenth International Conference on Learning Representations*.

Zhuo Chen, Yuxuan Miao, and Deyi Xiong. 2026. Data mixing for large language models pretraining: A survey and outlook. *Data Intelligence*, 8(1):15–55.

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

Simin Fan, Matteo Pagliardini, and Martin Jaggi. 2024. DOGE: Domain reweighting with generalization estimation. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 12895–12915. PMLR.

Steven Feng, Shrimai Prabhumoye, Kezhi Kong, Dan Su, Mostofa Patwary, Mohammad Shoeybi, and Bryan Catanzaro. 2024. Maximize your data's potential: Enhancing llm accuracy with two-phase pretraining. *Preprint*, arXiv:2412.15285.

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

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, and 5 others. 2024. The language model evaluation harness.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *Preprint*, arXiv:2001.08361.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: a highly efficient gradient boosting decision tree. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 3149–3157, Red Hook, NY, USA. Curran Associates Inc.

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. 2018. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 252–262, New Orleans, Louisiana. Association for Computational Linguistics.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. RACE: Large-scale ReAding comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794, Copenhagen, Denmark. Association for Computational Linguistics.

Shengrui Li, Fei Zhao, Kaiyan Zhao, Jieying Ye, Haifeng Liu, Fangcheng Shi, Zheyong Xie, Yao Hu, and Shaosheng Cao. 2026. Decouple searching from training: Scaling data mixing via model merging for large language model pre-training. *arXiv preprint arXiv:2602.00747*.

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

Qian Liu, Xiaosen Zheng, Niklas Muennighoff, Guangtao Zeng, Longxu Dou, Tianyu Pang, Jing Jiang, and Min Lin. 2025. Regmix: Data mixture as regression for language model pre-training. In *The Thirteenth International Conference on Learning Representations*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Jing Ma, Chenhao Dang, and Mingjie Liao. 2025. Actor-critic based online data mixing for language model pre-training. *Preprint*, arXiv:2505.23878.

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

Kaixiang Mo, Yuxin Shi, Weiwei Weng, Zhiqiang Zhou, Shuman Liu, Haibo Zhang, and Anxiang Zeng. 2025. Mid-training of large language models: A survey. *Preprint*, arXiv:2510.06826.

Yonatan Oren, Shiori Sagawa, Tatsunori B. Hashimoto, and Percy Liang. 2019. Distributionally robust language modeling. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4227–4237, Hong Kong, China. Association for Computational Linguistics.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1525–1534, Berlin, Germany. Association for Computational Linguistics.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. Winogrande: An adversarial winograd schema challenge at scale. *arXiv preprint arXiv:1907.10641*.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. *Advances in neural information processing systems*, 32.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Jiapeng Wang, Changxin Tian, Kunlong Chen, Ziqi Liu, Jiaxin Mao, Wayne Xin Zhao, Zhiqiang Zhang, and Jun Zhou. 2026. Mergemix: Optimizing mid-training data mixtures via learnable model merging. *Preprint*, arXiv:2601.17858.

Yifan Wang, Binbin Liu, Fengze Liu, Yuanfan Guo, Jiyao Deng, Xuecheng Wu, Weidong Zhou, Xiaohuan Zhou, and Taifeng Wang. 2025. Tikmix: Take data influence into dynamic mixture for language model pre-training. *Preprint*, arXiv:2508.17677.

Johannes Welbl, Nelson F. Liu, and Matt Gardner. 2017. Crowdsourcing multiple choice science questions. In *NUT@EMNLP*.

Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifeng Lu, Percy Liang, Quoc V Le, Tengyu Ma, and Adams Wei Yu. 2023. Doremi: Optimizing data mixtures speeds up language model pretraining. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. 2024. Tinyllama: An open-source small language model. *Preprint*, arXiv:2401.02385.

A Appendix

A.1 Implementation Details

The hyperparameters we used are: AdamW optimizer (Loshchilov and Hutter, 2019) with weight decay 0.1, learning rate 4e-4, context length 2048, and global batch size 512 (achieved via gradient accumulation). Proxy models are trained on 1 H800 GPU for 1,000 steps (1M tokens per step) and the target model is trained on 8 GPUs for 25,000 steps, totaling 25B tokens. Human stands for the original Pile token distribution. The adjusted 17 domains are from Liu et al. (2025).

For all baselines (Human, DoReMi, and RegMix), we use the open-sourced mixtures released and train target models under our exact hyperparameter setup. This isolates the contribution of the mixture itself from any unrelated training-recipe

Algorithm 1 Offline Mixture Schedule Generation

Require: regression model f_N, switch points $\{s_1, \ldots, s_N\}$, initial loss $\ell_0 = \frac{1}{M}\sum_{i=1}^{M} \ell_1^{(i)}$, candidate count C, top-k, human prior m_0^{human}, concentration α
Ensure: schedule $\mathcal{S} = \{(s_j, m_j^*)\}_{j=1}^{N}$

$\mathcal{S} \leftarrow \emptyset$
for $j = 1, \ldots, N$ **do**
  Sample $\{m^{(c)}\}_{c=1}^{C} \sim \mathrm{Dir}(\alpha \cdot m_0^{\text{human}})$
  $\hat{\ell}^{(c)} \leftarrow f_N(s_j, m^{(c)}, \ell_{j-1})$ for $c = 1, \ldots, C$
  $\mathcal{I} \leftarrow \mathrm{TopK}_c\big(-\hat{\ell}^{(c)}\big)$ ▷ indices of k lowest losses
  $m_j^* \leftarrow \frac{1}{k}\sum_{c \in \mathcal{I}} m^{(c)}$
  $\ell_j \leftarrow f_N(s_j, m_j^*, \ell_{j-1})$ ▷ input loss for next iter
  $\mathcal{S} \leftarrow \mathcal{S} \cup \{(s_j, m_j^*)\}$
end for
return $\mathcal{S}$

Algorithm 2 Online Dynamic Mixture

Require: regression model f_N, target model $\mathcal{M}$, target-step switch points $\{S_1, \ldots, S_N\}$, candidate count C, top-k, scaling exponent β, size ratio P_t/P_p, human prior m_0^{human}, concentration α

Initialize $\mathcal{M}$ with mixture m_0^{human}
for each training step t **do**
  Take one training step of $\mathcal{M}$
  if $t \bmod T_{\text{eval}} = 0$ **then**
    $\hat{\ell}_t \leftarrow \mathrm{validate}(\mathcal{M})$ ▷ cached for online use
  end if
  if $t \in \{S_1, \ldots, S_N\}$, say $t = S_j$ **then**
    $\tilde{\ell}_{S_j} \leftarrow \hat{\ell}_{S_j} \cdot (P_t/P_p)^{\beta}$ ▷ scale to proxy
    $s_j \leftarrow t \cdot T_p/T_{\text{target}}$ ▷ proxy-step image
    Sample $\{m^{(c)}\}_{c=1}^{C} \sim \mathrm{Dir}(\alpha \cdot m_0^{\text{human}})$
    $\hat{\ell}^{(c)} \leftarrow f_N(s_j, m^{(c)}, \tilde{\ell}_{S_j})$ for $c = 1, \ldots, C$
    $\mathcal{I} \leftarrow \mathrm{TopK}_c\big(-\hat{\ell}^{(c)}\big)$
    $m_j^* \leftarrow \frac{1}{k}\sum_{c \in \mathcal{I}} m^{(c)}$
    Update $\mathcal{M}$'s training mixture to m_j^*
  end if
end for

differences. For RegMix and all REGMIX-D variants in Table 1, 1B target models are trained with 5 random seeds; significance is assessed with a paired t-test on the 13-task average over seeds. Ablations, the AC-ODM reproduction, and the 7B experiments use a single seed. We use LightGBM (Ke et al., 2017) as the regression model, matching the choice in RegMix.

We use the Pile-CC validation loss as the target predicted loss. Predictions over candidate mixtures are made by Dirichlet sampling 100K candidates and averaging the top-128 lowest predicted validation loss.

The full list of our evaluated tasks: HellaSwag (Zellers et al., 2019), PIQA (Bisk et al., 2020), OpenBookQA (Mihaylov et al., 2018), LAMBADA (Paperno et al., 2016), SciQ (Welbl et al., 2017), ARC-Easy (Clark et al., 2018), ARC-Challenge (Clark et al., 2018), COPA (Wang et al., 2019), RACE (Lai et al., 2017), LogiQA (Liu et al., 2020), QQP (Wang et al., 2018), WinoGrande (Sakaguchi et al., 2019), and MultiRC (Khashabi et al., 2018).

A.2 Overhead for REGMIX-D

REGMIX-D's online adaptation adds minimal cost to target training. Table 4 reports wall-clock training time on 8 GPUs (averaged over 3 runs): REGMIX-D Offline matches RegMix exactly (the schedule is generated before training), while REGMIX-D online incurs only $+0.37\%$ overhead from per-switch-point regression queries. This confirms that the additional flexibility of online adaptation comes at negligible computational cost.

Method	Time (hours)	Overhead
RegMix	21.77	—
RegMix-D (Offline)	21.76	$\approx 0\%$
RegMix-D (Online)	21.85	$+0.37\%$

Table 4: Target-model training wall-clock time on 8 GPUs (25,000 steps, averaged over 3 runs).

As for the proxy model training, one 1M proxy model requires about 0.5 hours on a single H800, reducing proxy number from 512 to 128 thus would reduce the total proxy training cost from approximately 256 to 64 H800 GPU-hours.

A.3 From Proxy Timescale to Target Training

REGMIX-D's regression model is trained entirely on proxy-scale trajectories (T_p steps), while target training is much longer ($T_{\text{target}} \gg T_p$). We assume that proxy and target training share enough loss-curve structure to make the proxy-predicted switch points transferable. Concretely, each proxy switch point s_j is linearly mapped to a target step

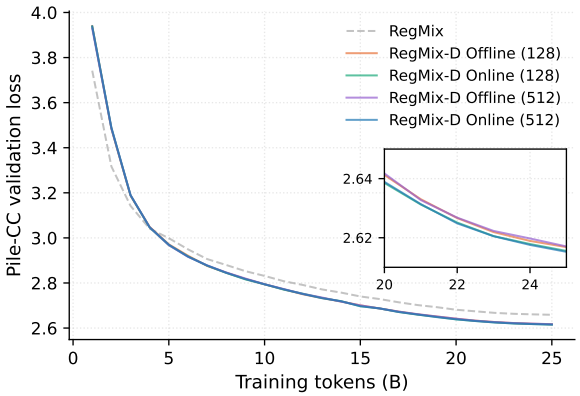


Figure 2: Pile-CC validation loss across training for RegMix and all four REGMIX-D variants (Offline/Online $\times$ 128/512 proxies), at N=5 switch points. All four REGMIX-D variants achieve consistently lower validation loss than RegMix after the initial segment. The inset zooms into the final 5B tokens, where RegMix-D Online (512) attains the lowest loss.

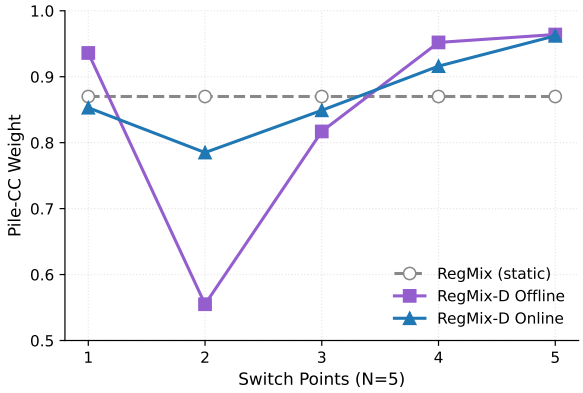


Figure 3: Pile-CC weight across training process for RegMix (static, dashed) and REGMIX-D (dynamic, N=5, 512 proxies). The REGMIX-D variants assign substantially different mixture weights than RegMix. This illustrates the dynamic schedules discovered by our method.

$s_j \cdot (T_{\text{target}}/T_p)$. This assumption underlies RegMix's original design as well: the static mixture selected from proxy losses is similarly assumed to transfer to target training. REGMIX-D inherits this assumption without strengthening it.

A.4 Visualization of the Dynamic Mixture

Figure 3 visualizes the pile-cc weights produced by REGMIX-D (Offline and Online) at the N=5, 512-proxy setting, compared with RegMix's static weight of 0.870. The dynamic schedules differ markedly from the static baseline: both variants assign substantially different pile-cc weights in 4 of 5 segments. The Offline variant exhibits a more pronounced dip in segment 2 (0.555), reflecting a stronger early redistribution of weight toward non-dominant domains, before concentrating back to high pile-cc weights in later segments. The Online variant follows a similar overall trajectory but with smaller fluctuations, likely because it conditions on actually observed target-model losses rather than recursively predicted ones.

Figure 4 shows the full schedules for all 17 domains, with colors indicating each weight relative to RegMix's static weight for the same domain (red: higher, blue: lower). Beyond pile-cc, REGMIX-D consistently reallocates weight among secondary domains (e.g., suppressing ubuntu_irc and up-weighting stackexchange throughout). Over time, both variants spread weight across a broader set of domains early in training (e.g., wikipedia and dm_mathematics) and concentrate on pile-cc in the final segments. The offline schedule makes sharper moves, while the online one changes more gradually. The two variants still share this overall shape despite differing on small domains, which suggests that the trajectory is not an artifact of error accumulation in the offline recursion.

A.5 Results at 7B Scale

We present the comparison between 7B-scale RegMix and both variants of REGMIX-D with 128 proxy models in Table 5. The gains persist at larger scale.

		RegMix-D (128)	
Task	RegMix	Offline	Online
HellaSwag	63.90	65.84	**66.20**
PIQA	74.78	75.69	**75.90**
OpenBookQA	36.55	37.03	**37.20**
LAMBADA	51.20	52.69	**53.44**
SciQ	91.80	**92.43**	92.39
ARC-Easy	66.00	67.48	**68.06**
ARC-Challenge	34.43	35.08	**35.41**
COPA	80.01	**82.22**	81.65
RACE	35.59	35.47	**35.61**
LogiQA	24.70	25.03	**25.98**
QQP	**55.78**	54.37	54.79
WinoGrande	61.10	**62.86**	62.57
MultiRC	50.87	50.85	**51.01**
Avg	55.91	56.70	**56.94**

Table 5: Downstream performance of a 7B target model trained on 100B Pile tokens (single run, $N = 5$, 128 proxies).

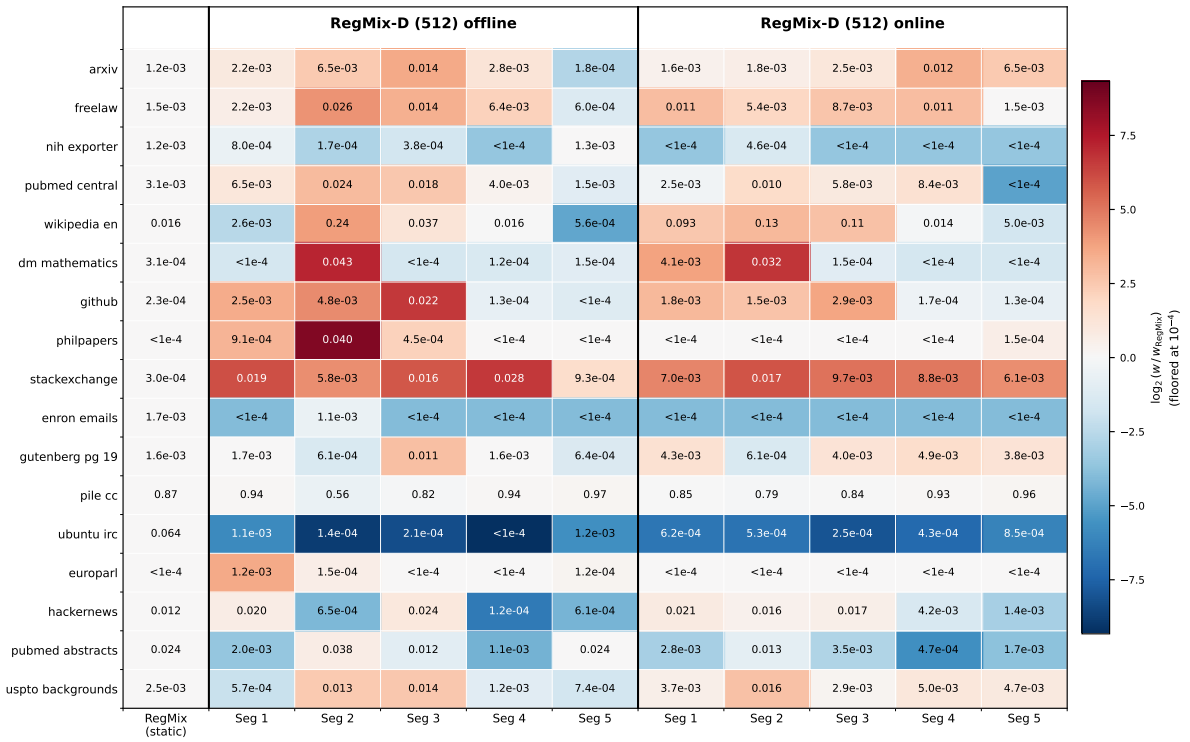


Figure 4: Full domain schedules of REGMIX-D (512 proxies, $N = 5$). Rows are the 17 Pile domains. The first column is RegMix's static mixture; the remaining columns are the mixtures selected at the five switch points for the offline (left) and online (right) variants (Seg j is the mixture used after the j-th switch point; the initial segment uses the Human mixture and is omitted). Numbers are absolute domain weights. Colors encode $\log_2(w/w_{\text{RegMix}})$, the log-ratio to RegMix's weight for the same domain (weights floored at 10^{-4}): red cells receive more weight than under RegMix, blue cells less.