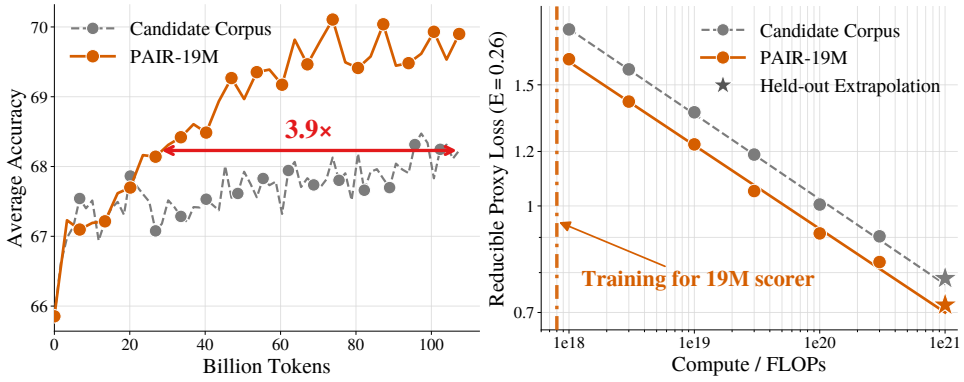
SMALL MODELS KNOW WHICH DATA SCALES

Anonymous authors
Paper under double-blind review

ABSTRACT

Large-scale pre-training requires scalable data selection that can both ***construct a high-quality training distribution*** and ***assess whether its advantage persists at scale***. Existing methods typically score documents independently, which gives little control over the distribution of the resulting corpus. They also validate selected data at a few isolated scales, using metrics that are neither comparable across data distributions nor extrapolable across compute. To address these challenges, we propose **PAIR** (**P**airwise **A**utoregressive **I**mportance **R**esampling), which uses paired lightweight language models to characterize the Candidate and high-quality Reference corpora and jointly calibrates resampling strength and retention rate to reconstruct the reference distribution under a finite data budget. To assess scaling value, we introduce the **Data Scaling Ladder**, which evaluates training distributions with benchmark-aligned proxy losses that follow predictable trends across compute and remain comparable across data distributions, enabling lower-compute experiments to forecast target-scale capability. With only a 19M dense scorer, PAIR achieves a **3.9× data-efficiency speedup** in continued pre-training of a 30B-A3B MoE model, closely matching the gain obtained with a 1.5B scorer. Across **1e18–1e21 FLOPs**, the two selected distributions follow nearly identical scaling trajectories and preserve their advantages on target-aligned capabilities, while lower-compute fits accurately predict the held-out 1e21-FLOP proxy loss. Together, these results show that training data for large-scale models can be effectively modeled, selected, and evaluated at a small fraction of the scale at which they are ultimately consumed.

1 INTRODUCTION



(a) CPT results (30B-A3B MoE)

(b) From scratch pre-training results

Figure 1: (a) Continued pre-training (CPT) accuracy across 17 tasks using a 30B-A3B MoE model pre-trained on 10T tokens. (b) From scratch pre-training results on our scaling ladder (1e18–1e21 FLOPs), in which we exclude the last point as held-out extrapolation and report the reducible proxy estimated from Candidate Corpus runs. In both settings, PAIR-19M uses 19M dense LMs as scorers to select data from the candidate corpus.

Large-scale pre-training relies not only on more data and compute, but also on the training distribution that determines what the model learns (Liu et al., 2025b; Magnusson et al., 2025; Shukor

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

et al., 2025; Li et al., 2026). Yet data decisions are typically made long before the final training run, when evaluating every candidate distribution at the target scale is prohibitively expensive. Scalable targeted data selection therefore requires answering two questions: ***Q1. How can we construct a target-aligned training distribution from a massive candidate corpus?*** and ***Q2. How can we determine whether its advantage will persist at much larger training scales?*** Together, they ask a broader question central to large-scale data construction: ***can data decisions made with small models remain valid for much larger training runs?***

For Q1, existing approaches typically assign each document an independent score, such as perplexity, quality, or estimated training utility, and then retain or resample high-scoring documents (Gu et al., 2025; Mizrahi et al., 2025). However, individually target-like documents need not collectively reproduce the coverage and relative composition of the target distribution. We therefore propose **PAIR** (**P**airwise **A**utoregressive **I**mportance **R**esampling), which reframes data selection from document-level valuation to corpus-level target-distribution reconstruction. PAIR uses lightweight language models trained on the candidate and target corpora to estimate their relative distributional preference, then jointly calibrates resampling strength and retention rate against the target corpus. Importantly, the estimated document score is only a means for constructing candidate subsets; the optimization target is the distribution induced by the selected corpus.

For Q2, performance measured at one or a few model scales provides only snapshots of a distribution's value and may not reveal whether its advantage persists with compute (Magnusson et al., 2025; The Microsoft AI Team, 2026). We therefore construct a **Data Scaling Ladder** that traces and extrapolates data value across scales. At each compute budget, we first identify and freeze an empirical effective-frontier training recipe, ensuring that training distribution is the only varying factor. We then evaluate each distribution using benchmark-aligned proxy losses that satisfy two requirements: *Scale Predictability*, so that proxy loss evolves regularly with compute, and *Cross-Distribution Comparability*, so that the same proxy loss carries a consistent capability meaning across training distributions. Together, these properties provide a stable compute-to-capability trajectory for estimating data value beyond the scales directly evaluated.

As shown in Figure 1, our experiments show that the scale required to make data decisions can be far smaller than the scale at which the resulting data are consumed. PAIR instantiated with a 19M dense scorer constructs a training distribution that achieves a 3.9× data-efficiency speedup when continued-pretraining a 30B-A3B MoE model, closely matching a 1.5B scorer despite an approximately 80× difference in scorer size. From-scratch experiments over 1e18-1e21 FLOPs further show that the two selected distributions follow nearly identical scaling trajectories and preserve their advantages on target-aligned capabilities. Moreover, fits using only lower-compute runs closely predict the held-out 1e21-FLOP proxy loss. These findings suggest that the compute required to construct and evaluate a training distribution need not scale with the compute required to train the model that consumes it.

Our main contributions are summarized as follows:

- We formulate targeted data selection as target-distribution reconstruction and propose **PAIR**, which jointly calibrates resampling strength and retention rate to construct a target-aligned training distribution using lightweight language models.
- We construct a Data Scaling Ladder that uses effective-frontier training configurations and benchmark-aligned proxy losses to measure and extrapolate the scaling value of training distributions.
- We show that data selected by a 19M scorer yield a 3.9× data-efficiency speedup for a 30B-A3B MoE model and exhibit scaling behavior comparable to a 1.5B scorer across 1e18-1e21 FLOPs.

2 Related Work

2.1 Targeted Pre-training Data Selection

Existing data-selection methods primarily estimate the utility or target affinity of individual documents. Perplexity- and quality-based methods rank data using language-model loss or learned quality signals (Marion et al., 2023; Wettig et al., 2024; Penedo et al., 2024; Li et al., 2024), while

model-aware approaches estimate document utility from training dynamics (Engstrom et al., 2024; Wang et al., 2025). Target-aware methods instead compare candidate data with a reference distribution: Moore–Lewis (Moore & Lewis, 2010) filtering and CoLoR-Filter (Brandfonbrener et al., 2024) use relative language-model losses. Despite different scoring signals, these methods largely determine how individual documents should be ranked or weighted. PAIR instead treats the distribution induced by the finite-budget selected corpus as the optimization target, using paired language models to parameterize target-aware resampling and jointly calibrating its sampling strength and retention rate against the reference distribution.

2.2 Evaluating Data Value Across Scale

Data-selection methods are typically evaluated by training on the selected corpus and comparing validation loss or downstream performance with alternative data distributions, often at only one or a few model scales (Fan et al., 2025; Dai et al., 2025). Such evaluations reveal whether a distribution is useful at the observed scale, but not whether its relative advantage persists at the substantially larger scale where the data will ultimately be consumed; indeed, multi-scale data ablations have shown that the ranking of data mixtures can change with compute (The Microsoft AI Team, 2026). Scaling-law studies establish predictable relationships between training compute and language-model loss (Kaplan et al., 2020; Hoffmann et al., 2022; Wang et al., 2026), while recent work shows that pretraining or task-conditioned losses can provide more regular indicators of downstream capability than benchmark scores alone (Du et al., 2024; Team et al., 2025). These results provide useful tools for forecasting, but do not directly measure the target-scale value of alternative training distributions. Our Data Scaling Ladder addresses this gap by testing whether a distribution's advantage persists with compute and extrapolating that advantage toward the intended training scale.

3 PAIR: Pairwise Autoregressive Importance Resampling

Given a Candidate Corpus $\mathcal{C}$ and a target reference corpus $\mathcal{T}$, our goal is to construct a finite subset $\mathcal{S} \subseteq \mathcal{C}$ whose training distribution matches $\mathcal{T}$ as closely as the support of $\mathcal{C}$ allows. PAIR separates this problem into two steps: paired language models identify how probability mass should shift from $\mathcal{C}$ toward $\mathcal{T}$, and target-side calibration determines how strongly to realize this shift under a finite data budget. Crucially, PAIR does not treat document scores as the final objective; they parameterize candidate subsets whose induced corpus distributions are explicitly matched to the target.

3.1 Pairwise Distribution Modeling

Autoregressive language models naturally model corpus distributions through next-token probabilities. We therefore train two lightweight LMs, $p_{\theta_\mathcal{C}}$ and $p_{\theta_\mathcal{T}}$, on the Candidate and target corpora, respectively. For a document x, let $\bar{\ell}_\mathcal{C}(x)$ and $\bar{\ell}_\mathcal{T}(x)$ denote its average token losses under the two models. PAIR defines the relative target preference

$$s(x) = \bar{\ell}_\mathcal{C}(x) - \bar{\ell}_\mathcal{T}(x). \tag{1}$$

This score is the document-level log-likelihood-ratio estimate induced by the paired autoregressive models: a larger $s(x)$ indicates that x is relatively more characteristic of the target distribution than of the Candidate distribution. Averaging over tokens places documents of different lengths on a common per-token scale.

The key role of $s(x)$ is to provide a redistribution direction, rather than an absolute notion of data quality. Considering that finite language models need not recover the exact density ratio, directly ranking by $s(x)$ or using its exponentiated value as a fixed sampling policy can over- or under-concentrate the selected corpus. PAIR therefore calibrates how this signal is converted into a finite training distribution.

3.2 Target-Calibrated Resampling

PAIR converts the relative preference into a family of resampling policies, and we obtain sampling weights $w_{\beta,\tau}(x)$ as

$$w_{\beta,\tau}(x) = \exp\left(\beta \mathrm{clip}(s(x), -\tau, \tau)\right), \tag{2}$$

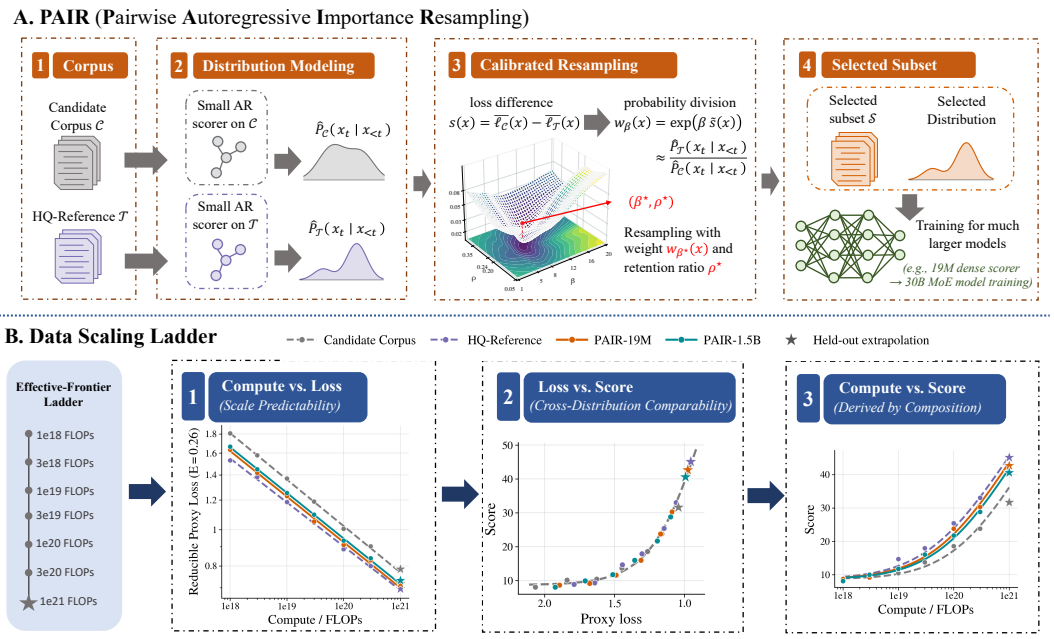


Figure 2: **A. PAIR** adopts lightweight autoregressive language models as density estimators of the candidate and target distributions and resamples the Candidate Corpus according to the estimated target-to-candidate likelihood ratio, constructing a target-aligned subset for training much larger models (e.g., 19M dense scorers to select data for 30B-A3B MoE model continued pre-training). **B. Data Scaling Ladder** evaluates training distributions along an empirical effective frontier using benchmark-aligned proxy sets whose losses scale predictably with compute and map consistently to downstream scores across data distributions, enabling capability-specific assessment of data scaling value. Here we report the reducible proxy estimated from Candidate Corpus runs.

where β controls the strength of redistribution toward the target. For each retention ratio ρ, we sample $\rho|\mathcal{C}|$ documents without replacement according to these weights, yielding a subset $\mathcal{S}_{\beta,\rho}$. Thus, β determines how strongly target-preferred documents are emphasized, while ρ controls the amount of Candidate support retained. We implement weighted sampling without replacement using Gumbel top-k (Kool et al., 2019).

Rather than choosing either parameter heuristically, PAIR calibrates β and ρ jointly by asking which induced subset distribution best matches the target distribution in paired-scorer feature space. For a corpus $\mathcal{A}$, we represent it by its token-weighted distribution $\Phi(\mathcal{A})$ in the paired scorer space $\big(\bar{\ell}_{\mathcal{T}}(x), s(x)\big)$. Given the sampled candidate corpus $\mathcal{C}^{\text{cal}}$ and target $\mathcal{T}^{\text{cal}}$, PAIR then jointly selects β and ρ by minimizing the Jensen-Shannon divergence between the selected and sampled target distributions

$$(\beta^\star, \rho^\star) = \arg\min_{\beta,\rho} D_{\text{JS}}\left(\Phi(\mathcal{S}^{\text{cal}}_{\beta,\rho}), \Phi(\mathcal{T}^{\text{cal}})\right). \tag{3}$$

Token weighting makes the calibration reflect the distribution actually consumed during pre-training rather than the number of documents alone. The two coordinates in the paired scorer space jointly retain the information provided by both scorers. At one extreme, $\beta = 0$ ignores the target and approaches uniform sampling; at the other, an excessively large β collapses mass onto the highest-scoring documents. Similarly, large ρ approaches the original Candidate distribution, whereas overly small ρ sacrifices target coverage. PAIR selects neither documents nor thresholds in isolation: it selects the finite-budget corpus distribution that best matches the target. Appendix A.2 visualizes the full joint calibration landscape for both scorer scales and reports the resulting $(\beta^\star, \rho^\star)$. The complete PAIR pipeline, including scorer subsampling, calibration-set construction, histogram estimation, and full-corpus selection, is summarized in Algorithm 1 and Appendix A.1.

4 DATA SCALING LADDER

A training distribution is valuable at scale only if its advantage survives at the scale where it will ultimately be consumed. We therefore define its *data scaling value* as the capability gain over the Candidate Corpus at a target compute. Estimating this value from smaller runs requires two forms of comparability: training distributions must be compared under controlled recipes across compute, and the evaluation signal must remain predictive across scale while preserving the same capability meaning across distributions. As illustrated in Figure 2B, our Data Scaling Ladder addresses these requirements with an *Effective-Frontier Ladder* and a *Cross-Distribution Proxy*.

4.1 EFFECTIVE-FRONTIER EVALUATION

To isolate the effect of training data, we first make data distributions comparable across compute. We construct an empirical effective frontier at seven approximately log-spaced budgets, $[1e18, 3e18, 1e19, 3e19, 1e20, 3e20, 1e21]$ FLOPs following prior scaling-law studies (Kaplan et al., 2020; Hoffmann et al., 2022). At each budget, we search for the training recipe that minimizes validation loss on the Candidate Corpus, including the model–token allocation and optimization configuration (Liu et al., 2025a; Bi et al., 2024). Once selected, the complete recipe is frozen and reused unchanged for every training distribution. Deriving the frontier only from the Candidate Corpus avoids optimizing the ladder toward any selected distribution and ensures that differences along the resulting trajectories are attributable to training data rather than distribution-specific retuning. Appendix A.4.1 provides the full effective-frontier construction protocol.

4.2 CROSS-DISTRIBUTION CAPABILITY PROXY

The harder problem is deciding *what* to scale. Downstream scores are noisy and sparse scaling signals, while a generic loss may scale smoothly without faithfully tracking the capability of interest. In particular, *predictability alone is insufficient*: even if a loss follows a regular trajectory within every training distribution, it cannot support data comparison if the same loss value corresponds to different capability levels across distributions.

We therefore require a benchmark-aligned proxy to satisfy two properties: ***Scale Predictability***, so that its loss can be reliably extrapolated with compute, and ***Cross-Distribution Comparability***, so that proxy loss preserves a shared relationship with downstream capability across training distributions. Together, they turn proxy loss into a common capability coordinate: as illustrated in Figure 2B, compute determines proxy loss, while proxy loss maps consistently to benchmark performance, yielding a predictable compute-to-capability trajectory. The exact compute–loss and loss–score parameterizations, fitting procedure, and held-out extrapolation protocol are provided in Appendix A.4.2.

Appendices A.4.3–A.4.4 systematically compare five proxy constructions across tasks and analyze their task-level success and failure modes. The results reveal a consistent principle: a good proxy is defined not by its length or predictability, but by whether its prediction target preserves the information required for task success. Adding irrelevant or template-heavy tokens can make loss scale more smoothly without preserving its capability meaning, whereas answer-only targets are sufficient when the native output is itself capability-complete, as in continuation and code-generation tasks. When the final answer discards essential reasoning or evidence, task-grounded, decision-complete targets are required to recover both properties. Based on this criterion, we admit only proxies that are simultaneously scale-predictable and cross-distribution comparable, and compose their compute–loss trajectories with the shared loss–score mapping to extrapolate relative data advantages toward the target compute scale.

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

We evaluate PAIR using a 3T-token **Candidate Corpus** and a 200B-token **HQ-Reference** corpus, with 19M- and 1.5B dense scorers, denoted PAIR-19M and PAIR-1.5B. We consider two complementary evaluation regimes: (1) continued pre-training of a 30B-A3B MoE model for 100B to-

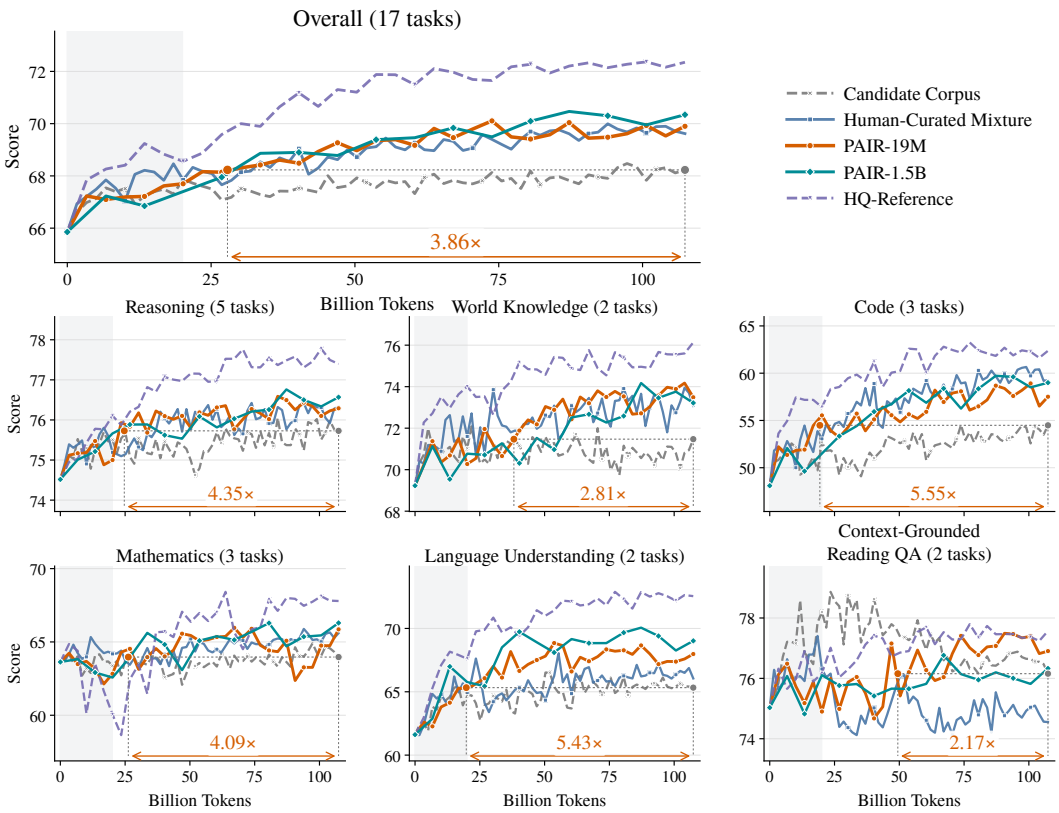


Figure 3: Continued pre-training performance on the 30B-A3B MoE model. We report benchmark trajectories for the aggregate score over 17 tasks and the average scores across six capability categories. We compare the Candidate Corpus, Human-Curated Mixture, HQ-Reference, PAIR-19M, and PAIR-1.5B.

kens, evaluated on 17 benchmarks across six capability categories; and (2) from-scratch pre-training along the 1e18–1e21 FLOP Data Scaling Ladder. Within each regime, all training configurations are shared across data distributions so that only the training data vary. Complete corpus statistics, scorer and calibration configurations, benchmark protocols, training settings, and baseline implementations are provided in Appendix A.3.

5.2 Large-Model Continued Pre-training

We first ask whether data selected by lightweight scorers can improve the continued pre-training of a substantially larger model. As shown in Figure 3, both PAIR-19M and PAIR-1.5B consistently outperform the Candidate Corpus on the overall 17-task score throughout training. PAIR-19M achieves a **3.9× data-efficiency speedup** overall, with consistent gains across all six capability categories, showing that the benefit extends beyond any single task family. We measure data-efficiency speedup by comparing the token budgets required by PAIR and the Candidate Corpus to reach the Candidate Corpus's final 17-task mean score, using a centered three-checkpoint moving average and linear interpolation to identify PAIR's first crossing; the shared 20B transition tokens are included in both budgets. Both PAIR variants also outperform the Human-Curated Mixture, indicating that target-calibrated resampling can recover an effective training distribution without manually specifying source-level mixing ratios.

More importantly, PAIR-19M and PAIR-1.5B follow remarkably similar trajectories despite an approximately 80× difference in scorer size. Their agreement when transferred to a 30B-A3B MoE model provides direct evidence that the data-selection signal does not need to be estimated at the scale of the model that ultimately consumes the data. Complete trajectories for all 17 individual benchmarks are reported in Appendix A.5.

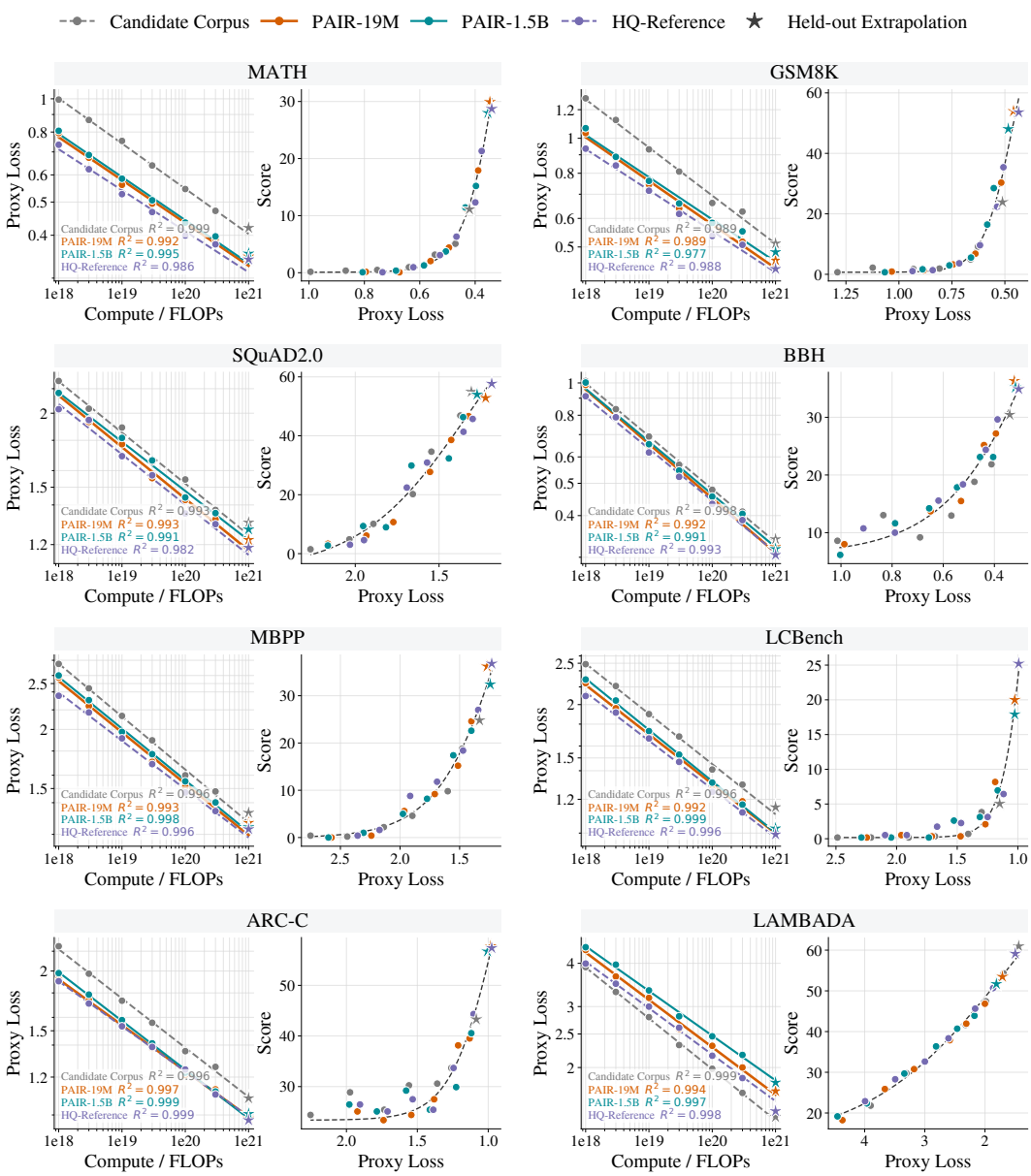


Figure 4: Results of pretraining from scratch on the data scaling ladder. For each benchmark-proxy pair, the left panel shows proxy loss as a function of training compute, while the right panel shows the relationship between proxy loss and benchmark score across training distributions. We adopt the FLOPs from 1e18 to 3e20 for fitting, and 1e21, marked by a star, is held out for extrapolation evaluation.

5.3 Predicting Data Value Across Scale

We next examine whether the advantages of the selected distributions remain predictable as training compute increases. As shown in Figure 4, proxy loss follows smooth scaling trajectories across all evaluated data distributions, with compute–loss fits achieving R^2 values of 0.977–0.999. The proxy loss also preserves a consistent relationship with downstream benchmark performance across distributions. Importantly, fits using only runs up to 3e20 FLOPs closely predict the held-out 1e21-FLOP proxy loss, validating the Data Scaling Ladder as a reliable tool for extrapolating target-scale capability through proxy loss from lower-compute experiments. Detailed fitting specifications and the held-out protocol are given in Appendix A.4.2, with task-level proxy diagnostics in Appendix A.4.4.

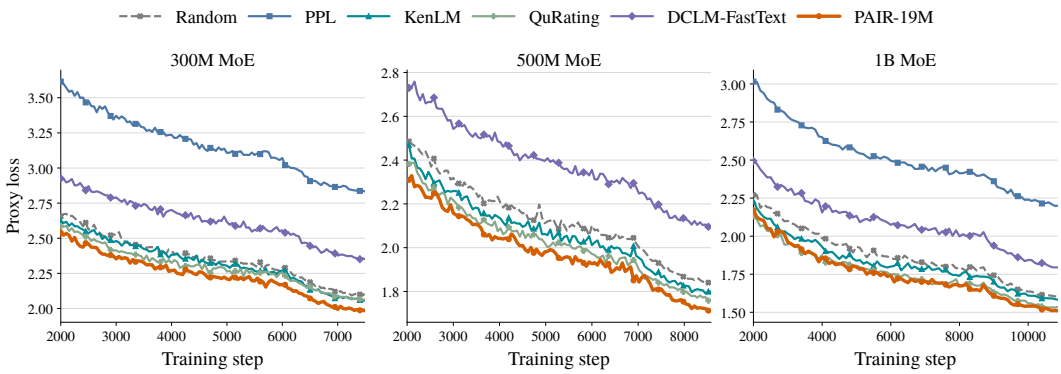


Figure 5: Training dynamics compared with data selection baselines. We fix the retention rate to $\rho = 0.22$ and apply selection from the same Candidate Corpus. We then pre-train 300M, 500M, and 1B MoE models on the resulting distributions and report their proxy loss trajectories across 8 tasks. For the same model size, all training configurations are identical and only the training data distribution varies. PPL is omitted for the 500M MoE because the corresponding run suffered persistent expert-load imbalance and failed to train.

Beyond validating the ladder itself, the scaling trajectories reveal that PAIR-19M and PAIR-1.5B retain nearly identical data scaling value across 1e18–1e21 FLOPs. On capabilities where the HQ-Reference distribution improves over the Candidate Corpus, both PAIR variants preserve their advantage as compute grows, with no visible collapse of the 19M-selected distribution relative to the 1.5B-selected one. Conversely, on LAMBADA (Paperno et al., 2016), where the HQ-Reference itself provides no advantage, neither PAIR variant exhibits a corresponding gain. This negative control highlights that PAIR reconstructs the multidimensional quality profile represented by the HQ-Reference, rather than imposing a single task-agnostic scalar notion of data quality. Together with the 30B continued-pretraining results, these findings show that data decisions made by a 19M scorer can remain effective both across model scale and across training compute.

5.4 Comparison with Data Selection Baselines

We further compare PAIR-19M with representative data-selection methods to test whether its gains extend beyond generic filtering or importance resampling. As shown in Figure 5, PAIR-19M consistently achieves the lowest proxy loss across 300M, 500M, and 1B MoE models, outperforming Random, PPL, KenLM (Heafield, 2011), QuRating (Wettig et al., 2024), and DCLM-FastText (Li et al., 2024). These results show that PAIR's advantage persists across model scales.

6 Analysis

6.1 Validating Joint Calibration

We verify whether the parameters selected by PAIR's calibration procedure indeed yield the most effective training distribution. As shown in Figure 6, the calibrated configuration $(\beta, \rho) = (8, 0.22)$ for PAIR-19M consistently achieves the lowest proxy loss when varying either β or ρ. Neither uniform-like sampling with weak target preference nor increasingly aggressive filtering improves upon the calibrated policy. Similarly, retaining either substantially more or less data leads to worse performance. These results confirm that the jointly calibrated $(\beta^\star, \rho^\star)$ provides the best trade-off between target preference and corpus coverage among the evaluated policies, validating the parameter-selection criterion used by PAIR. The full joint calibration landscapes for both scorer sizes are provided in Appendix A.2.

6.2 Compute Cost of Small-Model Data Selection

PAIR uses only lightweight models to characterize the Candidate and HQ-Reference corpora and score data for selection. As shown in Table 1, for PAIR-19M, we train two 19M dense scorers on

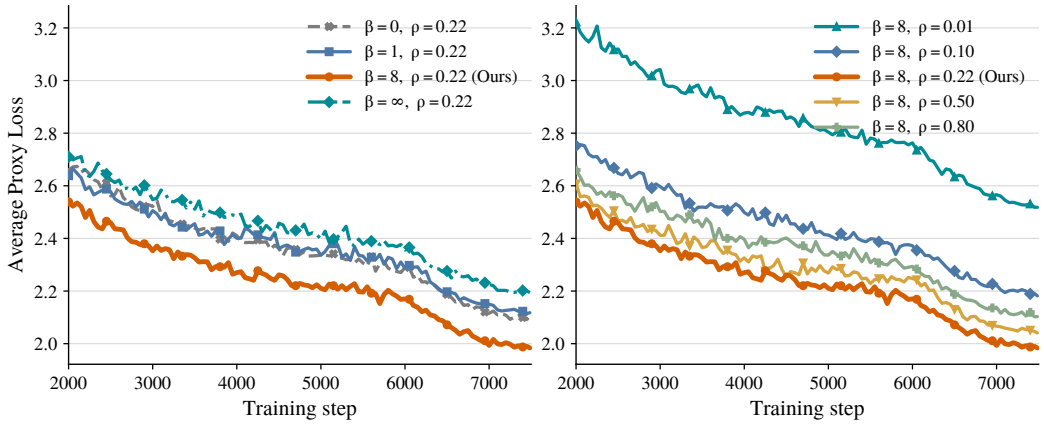


Figure 6: Ablation of the resampling strength β and retention rate ρ. Using the 19M scorer, we vary β while fixing $\rho = 0.22$ (left), and vary ρ while fixing $\beta = 8$ (right). Each selected distribution is evaluated by pre-training the same 300M MoE model and tracking its proxy-loss trajectory across 8 tasks.

Table 1: Compute cost of PAIR relative to continued pre-training (CPT). For PAIR scorers, Training FLOPs include training a single scorer. For the 30B-A3B CPT model, Training FLOPs denote the cumulative training compute already consumed by the checkpoint at the start of continued pre-training ($2e23$ FLOPs), rather than the compute of the subsequent CPT run. Relative costs are normalized to the corresponding training and inference costs of the 30B-A3B model. The full pipeline includes scorer training, scoring the 3T-token Candidate Corpus, and calibration.

Model / Pipeline	**Role**	**Training FLOPs**	**Inference FLOPs/token**	**CPT Result**
30B-A3B MoE	CPT Model	$2e23$ $(1\times)$	$6.6e9$ $(1\times)$	–
PAIR-19M	19M Dense	$7.9e17$ $(1/250{,}000\times)$	$8.8e7$ $(1/75\times)$	$3.9\times$
PAIR-1.5B	1.5B Dense	$2e21$ $(1/100\times)$	$3.3e9$ $(1/2\times)$	$3.9\times$
End-to-end selection cost				
PAIR-19M	Full Pipeline	$5.3e20$ FLOPs $(1/380\times)$		$3.9\times$
PAIR-1.5B	Full Pipeline	$2.4e22$ FLOPs $(1/8\times)$		$3.9\times$

3B tokens each, apply both scorers to the 3T-token Candidate Corpus, and score only 1B tokens from the HQ-Reference corpus for calibration. The complete selection pipeline consumes $5.3e20$ FLOPs, approximately $380\times$ less than the $2e23$ FLOPs already consumed by the 30B-A3B model at the starting checkpoint for continued pre-training. Despite this gap, PAIR-19M achieves a $3.9\times$ data-efficiency speedup and produces data with nearly the same scaling behavior as that selected by the 1.5B scorers. These results show that the compute required for data selection need not scale with the target model: lightweight language models can construct high-value training distributions that transfer across substantial model-scale and architecture gaps.

7 Conclusion

This work shows that data construction for large-scale pre-training need not operate at the scale of the model that ultimately consumes the data. PAIR uses lightweight language models to reconstruct a high-quality reference distribution from a massive Candidate Corpus, while the Data Scaling Ladder evaluates whether the resulting data advantage persists as training compute increases. With only a 19M scorer, PAIR achieves a $3.9\times$ data-efficiency speedup in continued pre-training of a 30B-A3B MoE model and closely matches the scaling behavior of a 1.5B scorer across 1e18–1e21 FLOPs. These results suggest that the compute required to model, select, and evaluate training data can be far smaller than that required to train the models that learn from it.

AI USE STATEMENT

In this work, we used generative AI tools for generating synthetic data sets, helping formulate mathematical claims, designing or providing feedback on research methodology or experiments, assisting with translation, cleaning and reformatting dataset, supporting qualitative and thematic data analysis, interpreting results. We have not used generative AI tools for helping develop theoretical models or conceptual frameworks, and providing critical ingredients for proving mathematical claims, assisting in the writing of proofs. Proposing or refining hypotheses are not applicable to this work. Additionally, we used generative AI tools for summarizing or analysing existing literature, creating or modifying scientific figures or images, identifying relevant literature. We have reviewed all AI-assisted work. We manually checked the correctness of LLM-generated and formatted proxy sets. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We have reported the implementation details in Section 5.1, Appendix A.1, A.3, and A.4. We will release the complete proxy sets and source code upon paper acceptance.

REFERENCES

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

Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiushi Du, Zhe Fu, et al. Deepseek llm: Scaling open-source language models with longtermism. *arXiv preprint arXiv:2401.02954*, 2024.

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

David Brandfonbrener, Hanlin Zhang, Andreas Kirsch, Jonathan Richard Schwarz, and Sham Kakade. Color-filter: Conditional loss reduction filtering for targeted language model pre-training. *Advances in Neural Information Processing Systems*, 37:97618–97649, 2024.

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

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

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

Yalun Dai, Yangyu Huang, Xin Zhang, Wenshan Wu, Chong Li, Wenhui Lu, Shijie Cao, Li Dong, and Scarlett Li. Data efficacy for language model training. *arXiv preprint arXiv:2506.21545*, 2025.

Zhengxiao Du, Aohan Zeng, Yuxiao Dong, and Jie Tang. Understanding emergent abilities of language models from the loss perspective. *Advances in neural information processing systems*, 37:53138–53167, 2024.

Logan Engstrom, Axel Feldmann, and Aleksander Madry. Dsdm: Model-aware dataset selection with datamodels. *arXiv preprint arXiv:2401.12926*, 2024.

Ziqing Fan, Siyuan Du, Shengchao Hu, Pingjie Wang, Li Shen, Ya Zhang, Dacheng Tao, and Yanfeng Wang. Combatting dimensional collapse in llm pre-training data via submodular file selection. In *International Conference on Learning Representations*, volume 2025, pp. 6655–6679, 2025.

Yuxian Gu, Li Dong, Hongning Wang, Yaru Hao, Qingxiu Dong, Furu Wei, and Minlie Huang. Data selection via optimal control for language models. In *International Conference on Learning Representations*, volume 2025, pp. 61932–61959, 2025.

Kenneth Heafield. Kenlm: Faster and smaller language model queries. In *Proceedings of the sixth workshop on statistical machine translation*, pp. 187–197, 2011.

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

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

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1601–1611, 2017.

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

Wouter Kool, Herke Van Hoof, and Max Welling. Stochastic beams and where to find them: The gumbel-top-k trick for sampling sequences without replacement. In *International conference on machine learning*, pp. 3499–3508. PMLR, 2019.

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

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

Jingyuan Liu, Jianlin Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin, Weixin Xu, Enzhe Lu, Junjie Yan, et al. Muon is scalable for llm training. *arXiv preprint arXiv:2502.16982*, 2025a.

Qian Liu, Xiaosen Zheng, Niklas Muennighoff, Guangtao Zeng, Longxu Dou, Tianyu Pang, Jing Jiang, and Min Lin. Regmix: Data mixture as regression for language model pre-training. In *International Conference on Learning Representations*, volume 2025, pp. 38305–38339, 2025b.

Ian Magnusson, Nguyen Tai, Ben Bogin, David Heineman, Jena D Hwang, Luca Soldaini, Akshita Bhagia, Jiacheng Liu, Dirk Groeneveld, Oyvind Tafjord, et al. Datadecide: How to predict best pretraining data with small experiments. *arXiv preprint arXiv:2504.11393*, 2025.

Max Marion, Ahmet Üstün, Luiza Pozzobon, Alex Wang, Marzieh Fadaee, and Sara Hooker. When less is more: Investigating data pruning for pretraining llms at scale. *arXiv preprint arXiv:2309.04564*, 2023.

David Mizrahi, Anders Boesen Lindbo Larsen, Jesse Allardice, Suzie Petryk, Yuri Gorokhov, Jeffrey Li, Alex Fang, Josh Gardner, Tom Gunter, and Afshin Dehghan. Language models improve when pretraining data matches target tasks. *arXiv preprint arXiv:2507.12466*, 2025.

Robert C Moore and William Lewis. Intelligent selection of language model training data. In *Proceedings of the ACL 2010 conference short papers*, pp. 220–224, 2010.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc-Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 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)*, pp. 1525–1534, 2016.

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

Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 784–789, 2018.

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

Mustafa Shukor, Louis Bethune, Dan Busbridge, David Grangier, Enrico Fini, Alaaeldin El-Nouby, and Pierre Ablin. Scaling laws for optimal data mixtures. *Advances in Neural Information Processing Systems*, 38:129554–129579, 2025.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed H Chi, Denny Zhou, et al. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 13003–13051, 2023.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. Commonsenseqa: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4149–4158, 2019.

MiniCPM Team, Chaojun Xiao, Yuxuan Li, Xu Han, Yuzhuo Bai, Jie Cai, Haotian Chen, Wentong Chen, Xin Cong, Ganqu Cui, et al. Minicpm4: Ultra-efficient llms on end devices. *arXiv preprint arXiv:2506.07900*, 2025.

The Microsoft AI Team. MAI-Thinking-1: Building a hill-climbing machine. Technical report, Microsoft AI, 2026. URL `https://microsoft.ai/pdf/mai-thinking-1.pdf`.

Pingjie Wang, Hongcheng Liu, Yusheng Liao, Ziqing Fan, Yaxin Du, Shuo Tang, Yanfeng Wang, and Yu Wang. Mining useful general data for low-resource domain adaptation. *arXiv preprint arXiv:2511.07380*, 2025.

Pingjie Wang, Zechen Hu, Peiru Yang, Fu Guo, and Debing Zhang. Smooth scaling laws hide stepwise token learning. *arXiv preprint arXiv:2606.29858*, 2026.

Yan Wang, Xiaojiang Liu, and Shuming Shi. Deep neural solver for math word problems. In *Proceedings of the 2017 conference on empirical methods in natural language processing*, pp. 845–854, 2017.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37:95266–95290, 2024.

Alexander Wettig, Aatmik Gupta, Saumya Malik, and Danqi Chen. Qurating: Selecting high-quality data for training language models. *arXiv preprint arXiv:2402.09739*, 2024.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 4791–4800, 2019.

Algorithm 1 PAIR: Pairwise Autoregressive Importance Resampling

Input: Candidate corpus $\mathcal{C}$, reference corpus $\mathcal{T}$; scorer-training budgets $D_C^{\mathrm{tr}}, D_T^{\mathrm{tr}}$; calibration budgets $D_C^{\mathrm{cal}}, D_T^{\mathrm{cal}}$; resampling-strength grid $\mathcal{B}$; retention-ratio grid $\mathcal{R}$; clipping threshold τ; histogram bin number K_1, K_2 and quantile range q
Output: Selected corpus $\mathcal{S}^\star$

▷ **Pairwise Distribution Modeling**

Sample scorer-training data: $\mathcal{C}_{\mathrm{tr}} \leftarrow \textsc{Sample}(\mathcal{C}, D_C^{\mathrm{tr}})$, $\mathcal{T}_{\mathrm{tr}} \leftarrow \textsc{Sample}(\mathcal{T}, D_T^{\mathrm{tr}})$
Train paired scorers: $p_{\theta_C} \leftarrow \textsc{Train}(\mathcal{C}_{\mathrm{tr}})$, $p_{\theta_T} \leftarrow \textsc{Train}(\mathcal{T}_{\mathrm{tr}})$
Sample reference calibration data: $\mathcal{T}_{\mathrm{cal}} \leftarrow \textsc{Sample}(\mathcal{T}, D_T^{\mathrm{cal}})$
Score candidate corpus: $\{\bar{\ell}_C(x), \bar{\ell}_T(x)\}_{x \in \mathcal{C} \cup \mathcal{T}_{\mathrm{cal}}} \leftarrow \textsc{Score}(\mathcal{C} \cup \mathcal{T}_{\mathrm{cal}}, p_{\theta_C}, p_{\theta_T})$
Compute pairwise preference: $s(x) \leftarrow \bar{\ell}_C(x) - \bar{\ell}_T(x), \forall x \in \mathcal{C} \cup \mathcal{T}_{\mathrm{cal}}$

▷ **Target-Calibrated Resampling**

Sample Candidate calibration data: $\mathcal{C}_{\mathrm{cal}} \leftarrow \textsc{Sample}(\mathcal{C}, D_C^{\mathrm{cal}})$
Construct histogram bins: $\{\mathcal{H}_j\}_{j=1}^{K_1 \times K_2} \leftarrow \textsc{BuildBins}(\{(\bar{\ell}_T(x), s(x))\}_{x \in \mathcal{C}_{\mathrm{cal}} \cup \mathcal{T}_{\mathrm{cal}}}, q, K_1, K_2)$
Define token-weighted distribution: $\Phi_j(\mathcal{A}) \leftarrow \frac{\sum_{x \in \mathcal{A}} |x| \mathbf{1}[(\bar{\ell}_T(x), s(x)) \in \mathcal{H}_j]}{\sum_{x \in \mathcal{A}} |x|}$, and construct reference distribution $\Phi(\mathcal{T}_{\mathrm{cal}})$
Initialize: $d^\star \leftarrow +\infty$
for $\beta \in \mathcal{B}$ **do**
    Compute calibration weights: $w_\beta(x) \leftarrow \exp\big(\beta \,\mathrm{clip}(s(x), -\tau, \tau)\big), \forall x \in \mathcal{C}_{\mathrm{cal}}$
    for $\rho \in \mathcal{R}$ **do**
        Resample calibration subset: $\mathcal{S}_{\beta,\rho}^{\mathrm{cal}} \leftarrow \textsc{GumbelTopK}(\mathcal{C}_{\mathrm{cal}}, w_\beta, \lceil \rho |\mathcal{C}_{\mathrm{cal}}| \rceil)$
        Compute distribution mismatch: $d_{\beta,\rho} \leftarrow D_{\mathrm{JS}}\big(\Phi(\mathcal{S}_{\beta,\rho}^{\mathrm{cal}}), \Phi(\mathcal{T}_{\mathrm{cal}})\big)$
        if $d_{\beta,\rho} < d^\star$ **then**
            Update optimum: $d^\star \leftarrow d_{\beta,\rho}$, $(\beta^\star, \rho^\star) \leftarrow (\beta, \rho)$
        end if
    end for
end for

▷ **Full-Corpus Selection**

Apply calibrated weights: $w^\star(x) \leftarrow \exp\big(\beta^\star \,\mathrm{clip}(s(x), -\tau, \tau)\big), \forall x \in \mathcal{C}$
Select final corpus: $\mathcal{S}^\star \leftarrow \textsc{GumbelTopK}(\mathcal{C}, w^\star, \lceil \rho^\star |\mathcal{C}| \rceil)$
return $\mathcal{S}^\star$

A Appendix

A.1 Complete PAIR Procedure

Algorithm 1 summarizes the complete implementation of PAIR. The procedure consists of three stages. First, we train the Candidate- and Reference-side scorers on sampled training subsets and use the resulting paired losses to compute the preference score $s(x)$ for the full Candidate Corpus and a sampled Reference subset. Second, we use sampled Candidate and Reference data to search over (β, ρ) and select the resampling policy whose induced token-weighted distribution $\Phi(\cdot)$ best matches the Reference distribution. Finally, the calibrated $(\beta^\star, \rho^\star)$ is applied to the precomputed scores of the full Candidate Corpus to construct the final training subset. This separation keeps scorer training and policy calibration inexpensive while requiring full-corpus processing only for scoring and the final selection.

A.2 Joint Calibration Landscape

PAIR is jointly controlled by the resampling strength β and retention ratio ρ. A small β approaches uniform sampling, whereas a large β over-concentrates the subset on high-scoring documents. Similarly, a large ρ moves the selected distribution toward the Candidate Corpus, whereas an excessively small ρ reduces coverage of the target distribution. Considering sampling without replacement makes document inclusion probabilities depend jointly on both parameters, β and ρ must be calibrated together. For each (β, ρ) pair, we measure the token-weighted Jensen-Shannon distance between the selected subset and the HQ-Reference corpus. As shown in Figure 7, the closest dis-

tributions are obtained at $(\beta^\star, \rho^\star) = (8, 0.22)$ for PAIR-19M and $(13, 0.23)$ for PAIR-1.5B. These parameters are fixed for all subsequent experiments, and Section 6.1 verifies that the target-side optima also yield the strongest downstream performance.

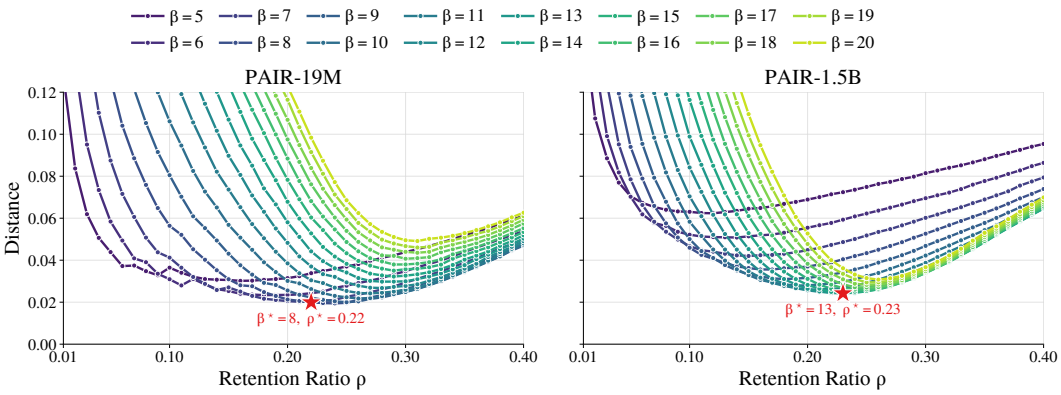


Figure 7: Joint calibration of PAIR's resampling parameters. For PAIR-19M (left) and PAIR-1.5B (right), each curve shows the token-weighted Jensen-Shannon distance between the selected subset and the HQ-Reference corpus as the retention ratio ρ varies at a fixed resampling strength β, and lower is better. Red stars $\star$ mark the joint minima, yielding $(\beta^\star, \rho^\star) = (8, 0.22)$ and $(13, 0.23)$, respectively. These parameters are fixed for all downstream experiments.

A.3 Experimental Details

Data Sources. We consider three types of training distributions. The **Candidate Corpus** $\mathcal{C}$ contains approximately 3T tokens and consists of a mixture of web, knowledge, code, Wikipedia, mathematics, and examination data. The **HQ-Reference** corpus $\mathcal{T}$ contains approximately 200B tokens curated from high-quality sources and serves as the target reference distribution for calibrated selection.

Selection Protocol. We instantiate PAIR at two scorer scales. For PAIR-19M, the Candidate- and target-side scorers are each trained on 3B tokens; for PAIR-1.5B, both scorers are trained on 200B tokens each. The paired scorers evaluate the full 3T-token Candidate Corpus, and 1B-token samples from the Candidate and HQ-Reference corpora are used for calibration. We jointly select β and ρ by minimizing the token-weighted Jensen–Shannon divergence between the induced subset and the HQ-Reference distribution in the paired-scorer feature space. This yields $(\beta, \rho) = (8, 0.22)$ for PAIR-19M and $(13, 0.23)$ for PAIR-1.5B; both configurations are fixed for all subsequent experiments. Complete settings are provided in Table 2.

Benchmarks. We evaluate models on 17 benchmarks spanning six capability categories: reasoning, world knowledge, code, mathematics, language understanding, and context-grounded reading QA. Each category contains 2∼5 tasks, providing broader coverage than evaluations based on only a few selected benchmarks. For each task, we follow its standard evaluation protocol and align the prompting format, number of demonstrations, decoding procedure, and evaluation metric with the official implementation whenever possible. Detailed task assignments and few-shot settings are provided in Table 3. Unless otherwise specified, category-level performance is the unweighted average across tasks within that category, and overall performance is the unweighted average across all 17 tasks. This task-level macro-averaging prevents benchmarks with larger test sets from dominating the reported results. All 17 tasks are used for continued-pretraining evaluation.

Evaluation. We evaluate each training distribution using both continued pre-training and pre-training from scratch. For continued pre-training, we initialize a 30B-A3B mixture-of-experts model from a checkpoint pretrained on the Candidate Corpus for 10T tokens. We then continue pre-training

[1] https://github.com/open-compass/CodeBench/
[2] https://github.com/flageval-baai/FlagEval/tree/master/csem

Table 2: Detailed configuration of PAIR.

Stage	Module	Hyperparameter
Pre-processing	Candidate Corpus	3T tokens
	HQ-Reference	200B tokens
	Tokenizer	Qwen2 tokenizer
Scorer Training	Scorer Size for PAIR-19M	19M-parameter dense
	Scorer Size for PAIR-1.5B	1.5B-parameter dense
	Training Budget $D_C^{\mathrm{tr}}, D_T^{\mathrm{tr}}$ for PAIR-19M	3B tokens
	Training Budget $D_C^{\mathrm{tr}}, D_T^{\mathrm{tr}}$ for PAIR-1.5B	200B tokens
	Training Framework	Megatron
	Max Sequence Length	8192
	Warmup Steps	500
	Adam	$(\beta_1, \beta_2) = (0.9, 0.95)$
Calibration	Calibration Budget $D_C^{\mathrm{cal}}, D_T^{\mathrm{cal}}$	1B tokens
	Histogram Bin Number K_1	80
	Histogram Bin Number K_2	240
	Quantile q	[0.001, 0.999]
	Clip Parameter τ	50
Scoring	Inference Framework	SGLang
Selection	Calibration Parameters (β, ρ) for PAIR-19M	$(8, 0.22)$
	Calibration Parameters (β, ρ) for PAIR-1.5B	$(13, 0.23)$

Table 3: Evaluation benchmarks and prompting settings. We evaluate 17 tasks spanning six capability categories. The Evaluation column reports the number of in-context demonstrations used for each task, with 0-shot denoting evaluation without demonstrations.

Category	**Tasks**	**Evaluation**
Reasoning	BBH (Suzgun et al., 2023)	3-shot
	CommonSenseQA (Talmor et al., 2019)	7-shot
	HellaSwag (Zellers et al., 2019)	10-shot
	PIQA (Bisk et al., 2020)	5-shot
	WinoGrande (Sakaguchi et al., 2021)	5-shot
World Knowledge	ARC-C (Clark et al., 2018)	25-shot
	MMLU-Pro (Wang et al., 2024)	5-shot
Code	MBPP (Austin et al., 2021)	3-shot
	LCBench[1]	3-shot
	HumanEval (Chen et al., 2021)	0-shot
Mathematics	MATH (Hendrycks et al., 2021)	4-shot
	GSM8K (Cobbe et al., 2021)	4-shot
	Math23K (Wang et al., 2017)	3-shot
Language Understanding	LAMBADA (Paperno et al., 2016)	0-shot
	C-SEM[2]	5-shot
Context-Grounded Reading QA	SQuAD2.0 (Rajpurkar et al., 2018)	3-shot
	TriviaQA (Joshi et al., 2017)	0-shot

the model for 100B tokens on different distributions. All experiments use the same model initialization, tokenizer, optimizer, learning-rate schedule, batch size, sequence length, and evaluation protocol. During the first 20B tokens, we apply the same linear data-mixture transition to all experiments to ensure training stability. We evaluate the model periodically throughout continued pre-training and report the resulting benchmark trajectories. For pre-training from scratch, we follow the effective-frontier protocol described in Section 4.1 and reuse the resulting frozen configurations for all training distributions.

Baselines. We compare our selection strategy against the following baselines. **Random** uniformly samples documents from the Candidate Corpus. **PPL** ranks documents according to their perplexity computed by the neural scorer and selects those with the highest perplexity scores. **KenLM** Heafield (2011); Moore & Lewis (2010) applies Moore–Lewis selection using two 5-gram LMs and retains documents with the highest normalized likelihood ratios. **QuRating** Wettig et al. (2024) applies a quality-based data selection framework that assigns scalar quality scores to documents based on criteria such as writing style, required expertise, facts and trivia, and educational value. **DCLM-FastText** Li et al. (2024) uses a supervised fastText classifier to distinguish high-quality documents from lower-quality web documents and retains documents with the highest classifier scores. Finally, **Human-Curated Mixture** represents the prevailing data-construction practice among large-scale pre-training teams, where experts manually specify source-level filtering rules and mixing ratios. Its recipe reflects experience accumulated through extensive data-ablation studies and repeated training experiments over a long period, rather than being designed specifically for this comparison.

A.4 Data Scaling Ladder: Fits and Proxy Validation

A.4.1 Effective-Frontier Construction

Following prior scaling-law studies (Kaplan et al., 2020; Hoffmann et al., 2022), we construct an effective-frontier ladder with seven approximately log-spaced compute budgets: $[1e18, 3e18, 1e19, 3e19, 1e20, 3e20, 1e21]$ FLOPs. At each budget, we fix the total training compute and search over model topology, model–token allocation, learning rate, batch size, learning-rate scheduler, and other training hyperparameters (Liu et al., 2025a; Bi et al., 2024). The recipe achieving the lowest validation loss within the search space defines the corresponding frontier point.

All configurations are trained on the Candidate Corpus and evaluated on a held-out Candidate validation set, avoiding optimization toward any selected subset. We then freeze the complete training recipe at each compute budget and apply it unchanged to all data distributions, ensuring that the training distribution is the only experimental variable.

A.4.2 Scaling Fits and Held-Out Extrapolation

Let C denote training compute, $L_{b,d}(C)$ the proxy loss for benchmark–proxy pair b under training distribution d, and $S_{b,d}(C) \in [0, 1]$ the corresponding benchmark score. For each task-level analysis, we fit $L_{b,d}(C) = A_{b,d}(C/1e20)^{-\alpha_{b,d}}$ separately for each distribution using unweighted ordinary least squares in log–log space. The fit uses the first six ladder points, from 1e18 to 3e20 FLOPs. We do not estimate an irreducible loss at the task level because fluctuations in individual proxy losses make the additional asymptotic parameter unstable. For the eight-task average, where task-specific fluctuations are substantially reduced, we instead fit $\bar{L}_d(C) = E + A_d(C/1e20)^{-\alpha_d}$. We estimate the irreducible loss E from the Candidate-Corpus trajectory by nonlinear least squares, hold it fixed across distributions, and fit each reducible trajectory $\bar{L}_d - E$ in log–log space.

For each benchmark–proxy pair, we pool observations from all training distributions and fit a shared loss–score mapping $g_b(L) = s_{0,b} + (1 - s_{0,b})/[1 + \exp((L - m_b)/t_b)]$ using unweighted nonlinear least squares. The mapping is shared across distributions rather than refitted for each one, directly testing whether proxy loss provides a distribution-independent capability coordinate. The compute–score relationship is obtained by composition as $\widehat{S}_{b,d}(C) = g_b(\widehat{L}_{b,d}(C))$. We report $R^2_{\text{loss-}C}$ for each distribution's log–log fit, $R^2_{\text{score-loss}}$ for the pooled shared sigmoid, and $R^2_{\text{score-}C}$ for the composed predictions.

The 1e21-FLOP point is excluded only from the compute–loss fit and serves as the held-out test of loss extrapolation. The shared loss–score mapping instead uses all seven ladder points pooled across

Table 4: Admission results for benchmark-aligned proxy designs. A benchmark–proxy pair is admitted only if its loss is predictable across compute and maps consistently to benchmark scores across training distributions. Green check marks denote the proxies selected for the Data Scaling Ladder, black check marks denote additional admitted designs, crosses indicate failure of at least one criterion, and dashes indicate inapplicable designs. Task-grounded Design V is selected for all decision-oriented tasks, whereas answer-only Design I is used for LAMBADA.

Category	Task	Proxy Design I Answer-only	Proxy Design II Shared Irrelevant CoT	Proxy Design III Template-heavy CoT	Proxy Design IV Correct-option-only CoT	Proxy Design V Task-grounded CoT
Reasoning	BBH	✗	✗	✗	✗	✓
World Knowledge	ARC-C	✗	✗	✗	✗	✓
Code	MBPP	✓	✗	✗	-	✓
	LCBench	✓	✗	✗	-	✓
Mathematics	MATH	✗	✗	✗	-	✓
	GSM8K	✗	✗	✗	-	✓
Language Modeling	LAMBADA	✓	✗	✗	-	✗
Context-Grounded Reading QA	SQuAD2.0	✗	✗	✗	-	✓

distributions, because the asymptotes, midpoint, and scale of a sigmoid cannot be reliably estimated from observations covering only a narrow loss range. In practical applications, this mapping should first be calibrated using representative models and training distributions spanning the relevant loss and score ranges. It can then be held fixed while lower-compute runs are used to fit and extrapolate the compute–loss trajectory. Thus, our held-out evaluation tests compute–loss extrapolation through a pre-calibrated loss–score mapping rather than extrapolation of the sigmoid itself.

A.4.3 Proxy Construction and Design Variants

A valid proxy set must satisfy the two requirements introduced in Section 4.2. *Scale Predictability* requires proxy loss to follow a stable, extrapolatable trajectory with compute, while *Cross-Distribution Comparability* requires a consistent mapping between the proxy loss and the benchmark score across training distributions. These requirements guide our comparison of alternative proxy designs.

We consider five designs: **I** uses only the final answer; **II** adds task-irrelevant reasoning shared across examples; **III** uses a repeated reasoning template with limited task-specific content; **IV** explains the correct option without rejecting the alternatives; and **V** uses a task-grounded, decision-complete reasoning trace that supports the correct answer and rejects the relevant alternatives. Figure 8 illustrates these designs using the same ARC-C example, isolating the effect of the proxy target while keeping the prompt fixed. We generated the proxy examples using the GPT-5, and the generated examples were subsequently randomly sampled and checked manually.

Constructing and validating each benchmark–proxy pair requires evaluation across all compute budgets and training distributions. We therefore select one or two representative tasks from each of the six capability categories, covering diverse output formats while keeping the ladder computationally tractable. This yields eight tasks: BBH, ARC-C, MBPP, LCBench, MATH, GSM8K, LAMBADA, and SQuAD2.0. All 17 tasks remain part of the downstream evaluation suite. Table 4 summarizes which proxy designs satisfy both admission requirements, with detailed task-level results presented next.

A.4.4 Task-Level Proxy Validation

Table 4 and Figures 9-16 show that proxy validity depends on the interaction between target content and task structure, rather than target length alone. Designs II and III often make proxy loss smoother by adding predictable tokens, but these tokens do not necessarily preserve a shared loss-score mapping across training distributions.

For BBH and ARC-C, answer-only Design I provides too few task-conditioned prediction events, yielding weak compute-loss fits and shared loss-score mappings ($R^2 = -0.18$ and 0.02, respectively). Designs II and III progressively improve scale predictability, but their generic or repeated content still dilutes the decision signal. Design IV shows that relevance matters more than verbosity: its correct-option rationale nearly matches Design V on BBH. However, it does not model the dis-

ARC-C

Prompt (shared)

```
Question: Which is responsible for creating wind?
A. wave action
B. solar energy
C. trees blowing
D. gravitational force

Instruction: Write a decision certificate that verifies the correct answer and rejects the relevant
alternatives. Then give the final answer as the option label plus the option text.

Use this exact output structure:
Decision certificate:
- ...
Final answer: ...
```

Answer — Proxy Design I (Answer-only)

```
Final answer: (B) solar energy
```

Answer — Proxy Design II (Shared Irrelevant CoT)

```
Reasoning:
This response is written as a short, well-formed sequence of sentences. It uses ordinary punctuation and ends
with a concise result.

Final Answer: (B) solar energy
```

Answer — Proxy Design III (Template-heavy CoT)

```
Reasoning:
Step 1: Read the task carefully.
Step 2: Use the relevant information.
Step 3: Check the result.
Task anchor: Decision certificate:
- Wind is created by the uneven heating of the Earth's surface by solar energy, causing air to move from
high-pressure to low-pressure areas.
Therefore, the answer follows.

Final Answer: (B) solar energy
```

Answer — Proxy Design IV (Correct-option-only CoT)

```
Reasoning:
Solar energy is responsible for creating wind because it heats the Earth's surface unevenly, causing air to move
from high-pressure areas to low-pressure areas, which generates wind.

Final answer: (B) solar energy
```

Answer — Proxy Design V (Task-grounded CoT)

```
Decision certificate:
- Wind is primarily caused by the uneven heating of the Earth's surface by the sun, which is solar energy.
- Solar energy heats the air, causing it to expand and rise, creating areas of low pressure.
- Cooler air then moves in to replace the rising warm air, resulting in wind.
- Wave action is a result of wind, not a cause of it, so option A is incorrect.
- Trees blowing is a consequence of wind, not a cause, making option C incorrect.
- Gravitational force affects tides and planetary motion but does not directly create wind, so option D is
incorrect.

Final answer: (B) solar energy
```

Figure 8: Five proxy designs for the same ARC-C example. All designs share the same prompt but use different prediction targets: answer only, irrelevant CoT, template-heavy CoT, correct-option-only CoT, and task-grounded CoT. The orange-highlighted Design V provides a decision-complete target that supports the correct answer and rejects every distractor, and is therefore selected as the ARC-C proxy. Proxy loss is computed only over the target tokens conditioned on the shared prompt.

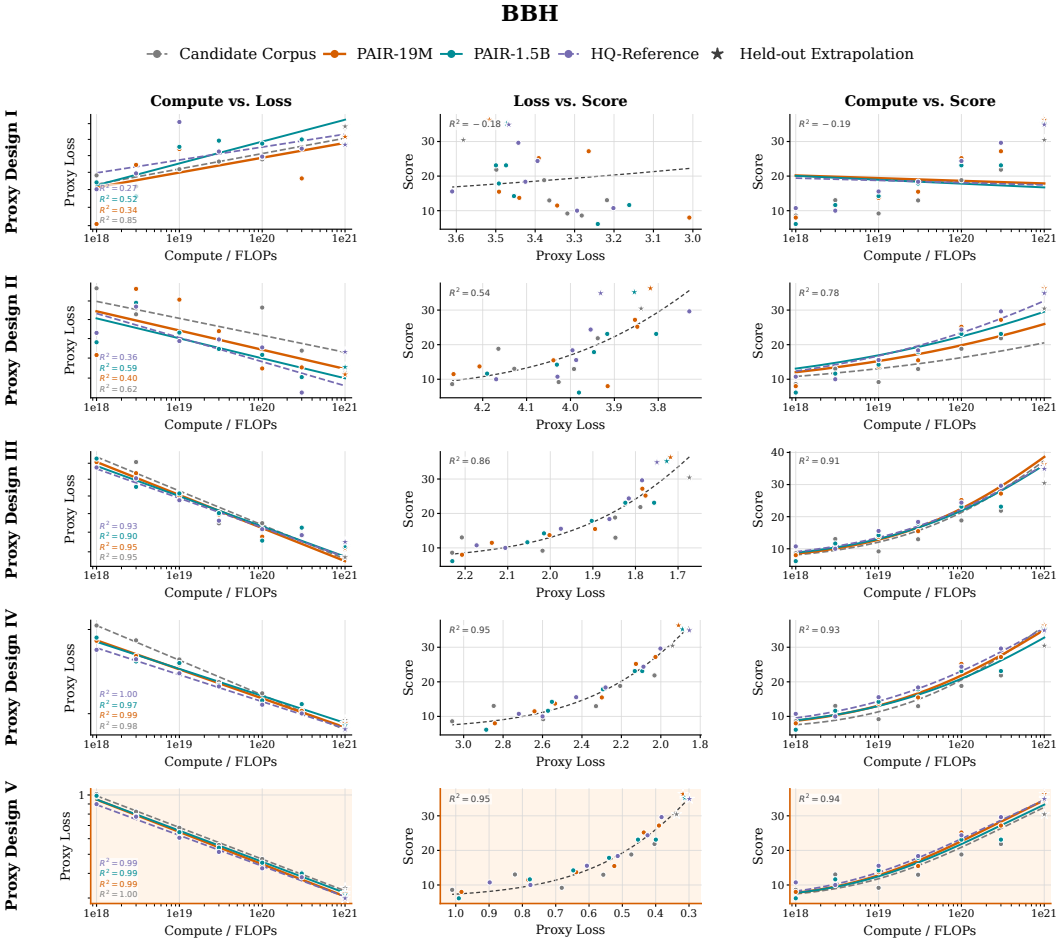


Figure 9: Proxy-design ablation for BBH. Rows compare Proxy Designs I–V; columns show proxy loss versus compute, the shared proxy-loss–score mapping across training distributions, and the resulting compute–score trajectories. Fits use the first 6 compute budgets, while stars mark the held-out 1e21-FLOP observations used to evaluate extrapolation. The orange-highlighted Design V is selected.

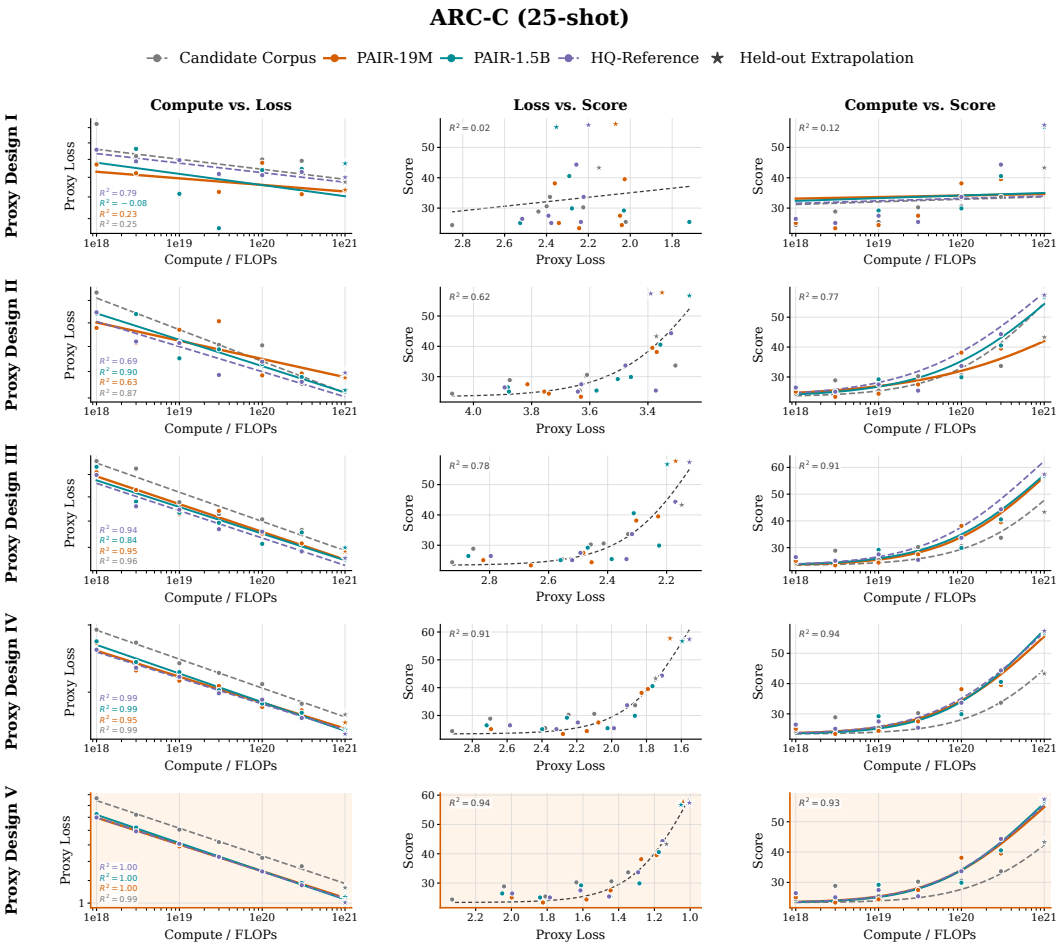


Figure 10: Proxy-design ablation for ARC-C. Rows compare Proxy Designs I–V; columns show proxy loss versus compute, the shared proxy-loss–score mapping across training distributions, and the resulting compute–score trajectories. Fits use the first 6 compute budgets, while stars mark the held-out 1e21-FLOP observations used to evaluate extrapolation. The orange-highlighted Design V is selected.

tinction between the correct option and competing distractors. On ARC-C, the option-complete Design V improves the worst-case compute-loss R^2 from 0.94 to 1.00, the shared loss-score R^2 from 0.91 to 0.94, and reduces the held-out 1e21-FLOP loss error from 5.4% to 1.4%. We therefore adopt Design V for its stronger worst-case robustness and broader applicability.

For MATH and GSM8K, short symbolic or numerical answers omit most of the computation that determines task success. Design I consequently produces irregular compute-loss trajectories and weak shared mappings, with loss-score R^2 values of 0.53 and -0.07. Irrelevant and template-heavy rationales improve some individual trajectories but remain only partially aligned with benchmark capability. By scoring complete worked solutions, Design V covers intermediate reasoning steps and diverse problem difficulties, yielding compute-loss R^2 values of 0.97-1.00 and shared loss-score R^2 values of 0.99 and 0.97 on MATH and GSM8K, respectively.

MBPP and LCBench provide an important contrast. Their answer-only targets are complete gold programs rather than short labels, and therefore already encode algorithms, control flow, and edge-case handling. Designs I and V both satisfy the two requirements, achieving compute-loss R^2 values of 0.98-1.00 and shared loss-score R^2 values of approximately 0.96-0.98. Designs II and III can still fail cross-distribution comparability despite smooth scaling. On LCBench, for example, Design II achieves per-distribution compute-loss $R^2 \geq 0.97$ but only $R^2 = 0.73$ for the shared loss-score mapping. This indicates that predictable boilerplate can dominate proxy loss without carrying a

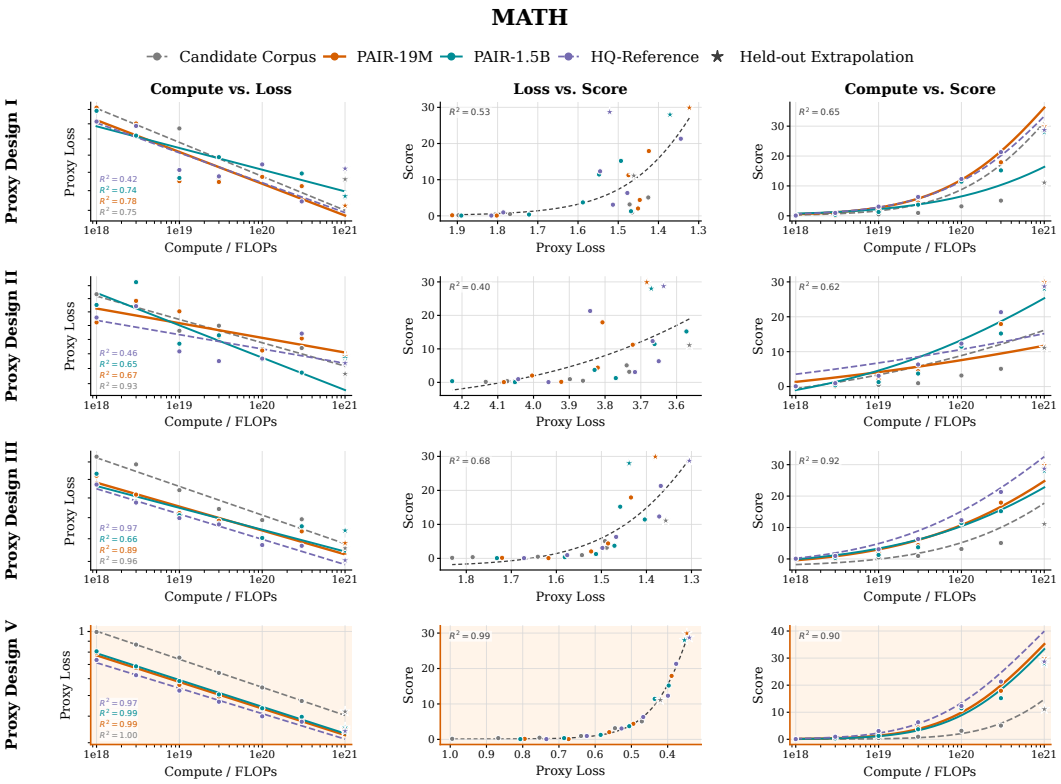


Figure 11: Proxy-design ablation for MATH. Rows compare Proxy Designs I–III and V; Design IV is not applicable to open-ended mathematical reasoning. Columns show proxy loss versus compute, the shared proxy-loss–score mapping across training distributions, and the resulting compute–score trajectories. Fits use the first six compute budgets, and stars mark the held-out 1e21-FLOP observations. The orange-highlighted task-grounded Design V is selected.

consistent code-capability meaning. We adopt Design V as the common task-grounded construction, while Design I remains an admissible alternative for code.

Finally, the appropriate construction must match the benchmark output structure. For LAMBADA, the gold continuation is exactly the benchmark prediction target; Design I therefore achieves compute-loss $R^2 = 0.99$-1.00 and shared loss-score $R^2 = 0.99$. Adding a rationale changes the measured quantity from continuation probability to explanation likelihood, causing Design V to retain smooth scaling but weaken cross-distribution comparability to $R^2 = 0.92$. SQuAD2.0 exhibits the complementary failure mode: answer-only loss has a strong shared mapping ($R^2 = 0.96$), but its held-out 1e21-FLOP losses are mispredicted by 16%-19%. Passage-grounded Design V supplies additional evidence-bearing prediction events, improving compute-loss fits to $R^2 = 0.99$ across all distributions while retaining a shared loss score $R^2 = 0.97$.

Overall, answer-only proxies are effective when the native output is sufficiently rich or exactly matches the benchmark objective, as in code generation and LAMBADA. For tasks whose final answers discard the underlying reasoning or evidence, task-grounded targets are required. These results confirm that scale predictability and cross-distribution comparability are distinct requirements: a valid proxy must simultaneously match the task interface, provide task-relevant predictive richness, and avoid loss being dominated by irrelevant or repeated templates.

A.5 Per-Task Continued Pre-training Results

Figure 17 decomposes the aggregate continued-pretraining results into all 17 tasks. After the common 20B-token transition, PAIR-19M and PAIR-1.5B follow broadly similar trajectories, with neither scorer exhibiting a systematic advantage. Improvements over the Candidate Corpus and Human-

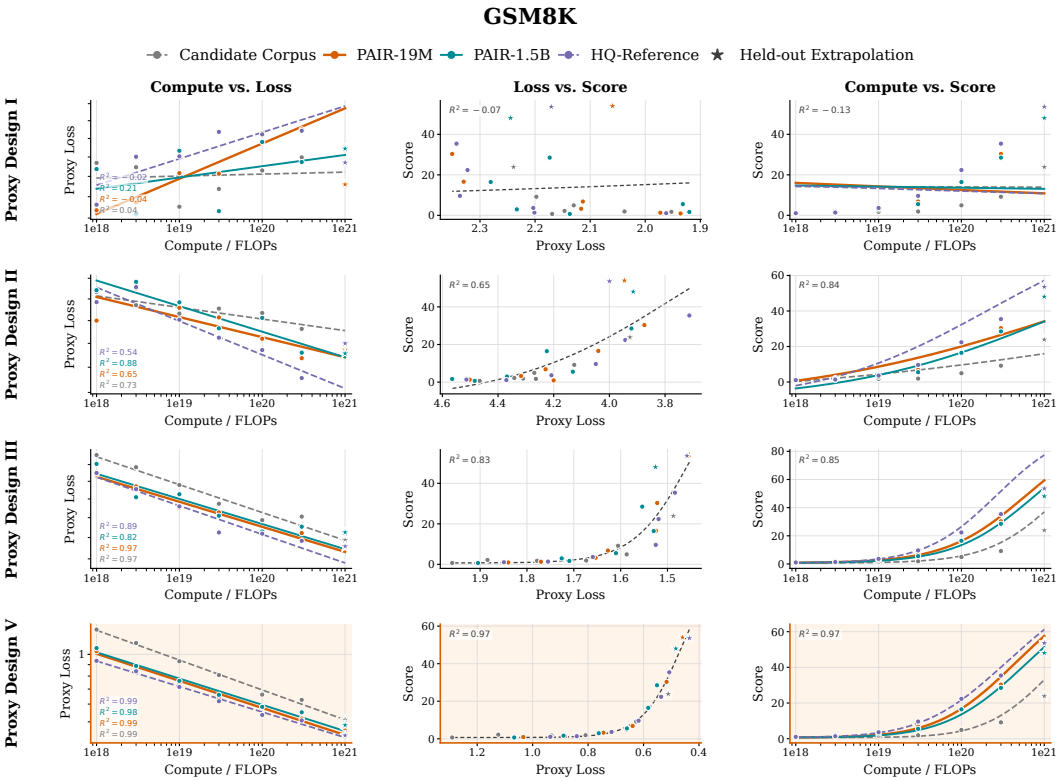


Figure 12: Proxy-design ablation for GSM8K. Rows compare Proxy Designs I–III and V; columns show proxy loss versus compute, the shared proxy-loss–score mapping, and the resulting compute–score trajectories. Fits use the first six compute budgets, while stars mark the held-out 1e21-FLOP observations. Answer-only loss exhibits poor scaling regularity, whereas the orange-highlighted task-grounded Design V satisfies both admission criteria.

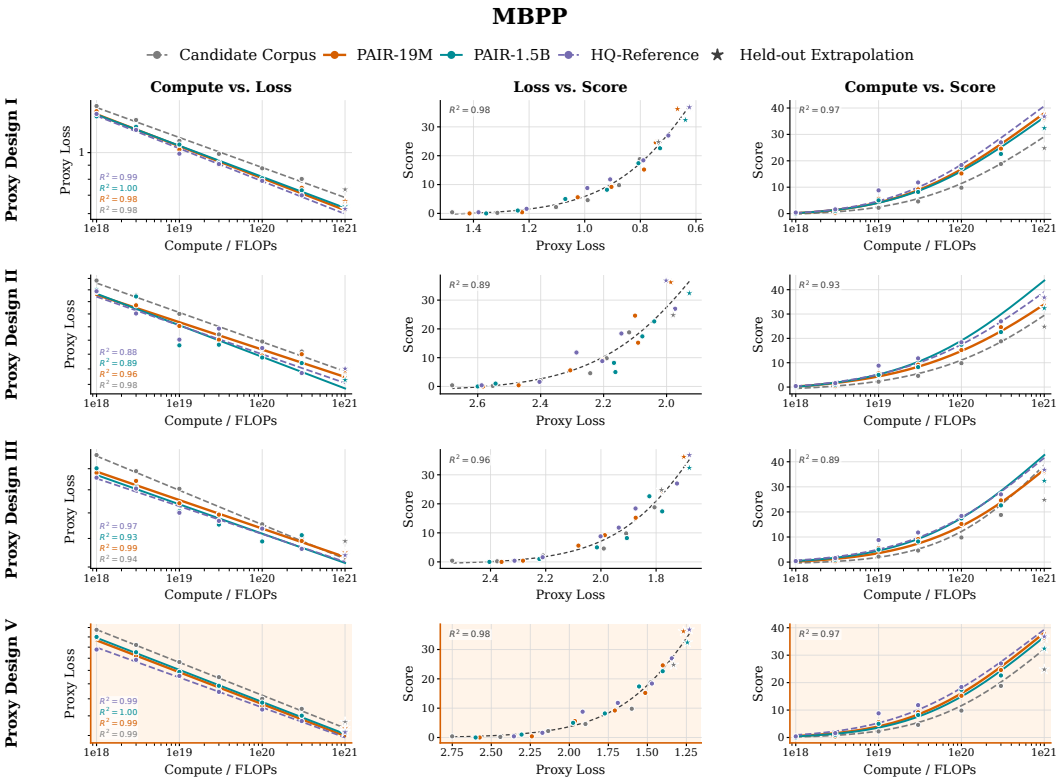


Figure 13: Proxy-design ablation for MBPP. Rows compare the applicable Proxy Designs I–III and V; Design IV is not applicable to open-ended code generation. Columns show proxy loss versus compute, the shared proxy-loss–score mapping, and the resulting compute–score trajectories. Stars denote held-out 1e21-FLOP observations. Designs I and V both satisfy the admission criteria, with the orange-highlighted Design V adopted for the ladder.

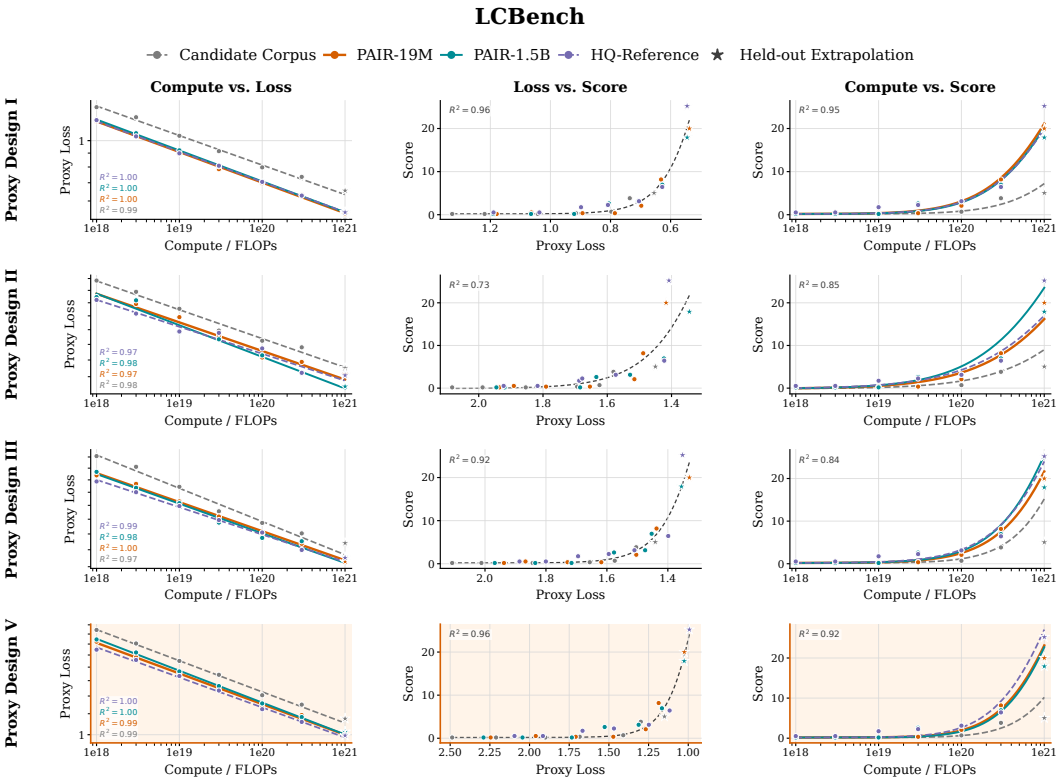


Figure 14: Proxy-design ablation for LCBench. Rows compare the applicable Proxy Designs I–III and V; Design IV is not applicable to open-ended code generation. Columns show proxy loss versus compute, the shared proxy-loss–score mapping, and the resulting compute–score trajectories. Stars denote held-out 1e21-FLOP observations. Designs I and V both pass the admission tests, with the orange-highlighted Design V used in the ladder.

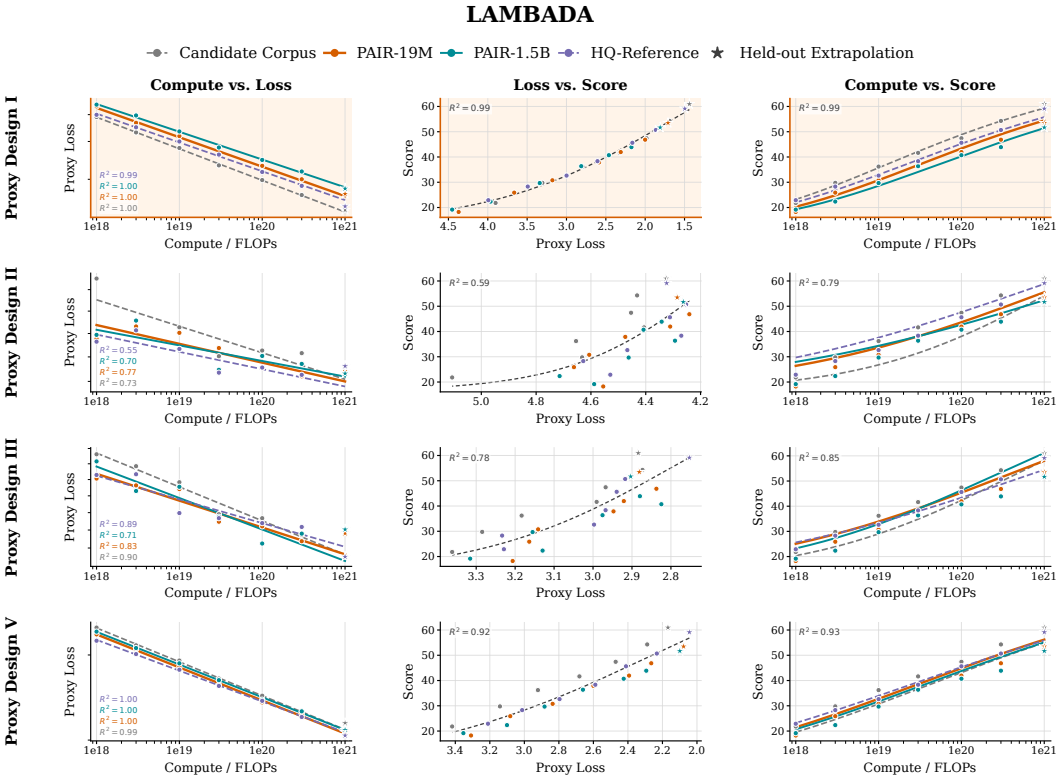


Figure 15: Proxy-design ablation for LAMBADA. Rows compare Proxy Designs I–III and V; columns show proxy loss versus compute, the shared proxy-loss–score mapping, and the resulting compute–score trajectories. Fits use the first six compute budgets, and stars mark the held-out 1e21-FLOP observations. Because LAMBADA is a continuation task, its gold continuation under the orange-highlighted answer-only Design I provides the admitted proxy.

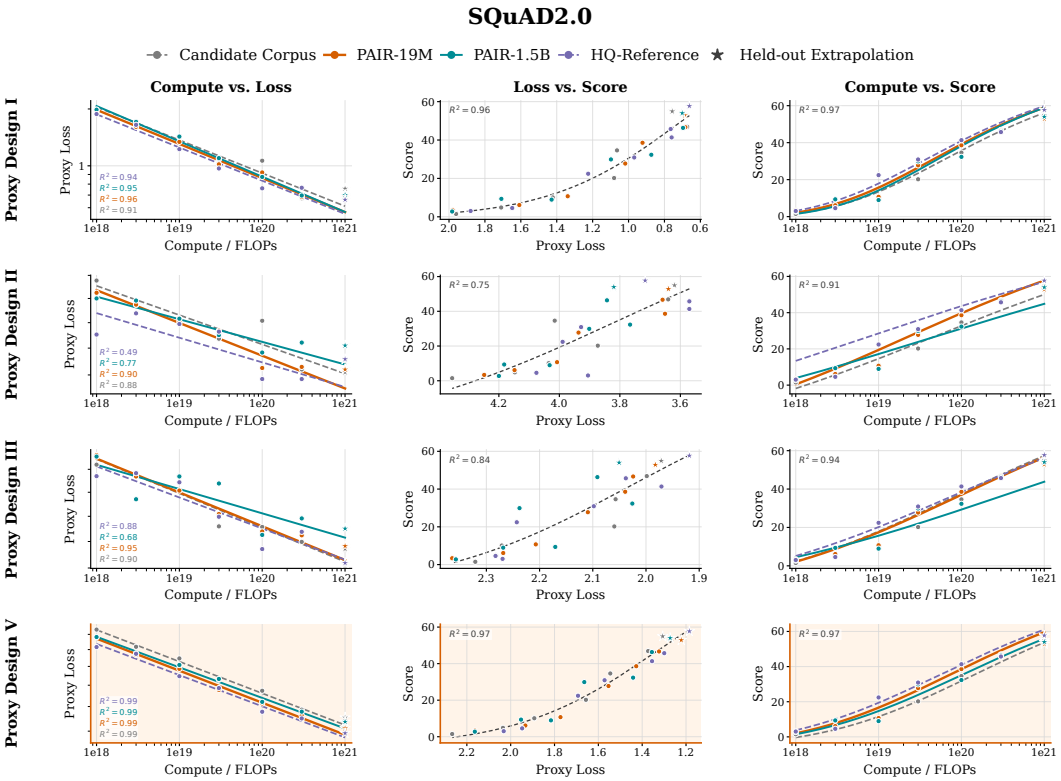


Figure 16: Proxy-design ablation for SQuAD2.0. Rows compare Proxy Designs I–III and V; Design IV is not applicable to open-ended reading comprehension. Columns show proxy loss versus compute, the shared proxy-loss–score mapping, and the resulting compute–score trajectories. Fits use the first six compute budgets, while stars mark the held-out 1e21-FLOP observations. The orange-highlighted task-grounded Design V is selected.

Curated Mixture appear across multiple capability categories, including BBH, ARC-C, MMLU-Pro, LCBench, Math23K, and C-SEM, whereas saturated or noisy tasks show smaller and less stable differences. This task-level heterogeneity reflects both the alignment of the HQ-Reference corpus with each capability and the coverage of corresponding data in the Candidate Corpus. Nevertheless, the consistent agreement between the two PAIR variants confirms that the distribution constructed by the 19M scorer transfers effectively to continued pre-training of the substantially larger 30B-A3B MoE model.

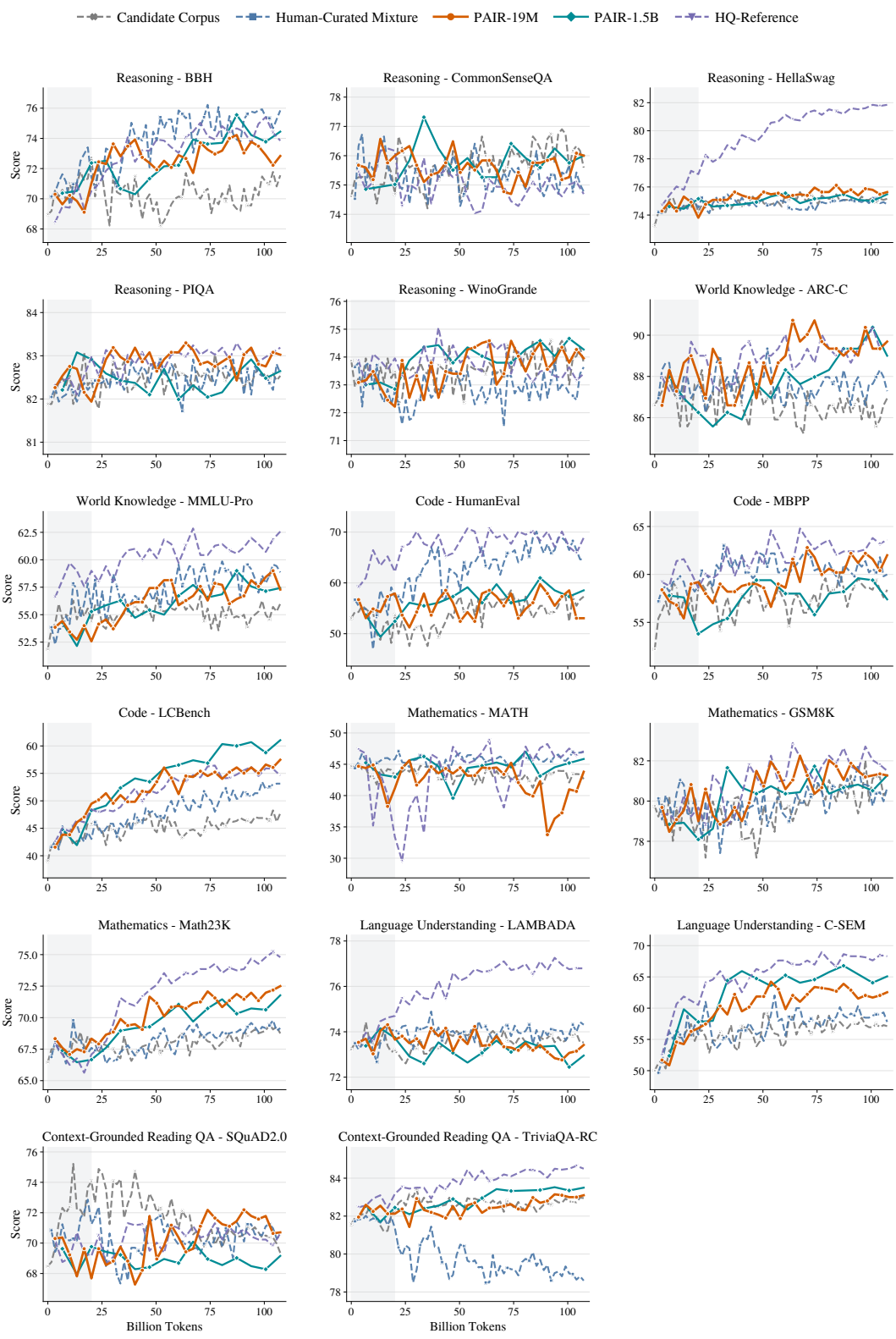


Figure 17: Complete per-task continued-pretraining results on the 30B-A3B MoE model. Each panel reports checkpoint-level benchmark scores as a function of additional training tokens for the Candidate Corpus, Human-Curated Mixture, PAIR-19M, PAIR-1.5B, and HQ-Reference distributions. All runs share the same initialization and training configuration and differ only in their training distributions. The shaded first 20B tokens denote the common linear data-mixture transition used for training stability. The two PAIR variants exhibit broadly comparable trajectories across the 17 tasks.