Calibrated Target-Aware Data Selection for Continual Mid-Training in Streams

Ke Wan [1] **Tianyi Zhao** [1] **Tianhao Huang** [1] **Guanghui Min** [1] **Chen Chen** [1]

Abstract

Mid-training is a practical mechanism for adapting foundation models to specialized domains after pretraining, yet current pipelines are typically static, assuming a fixed candidate corpus that can be curated or selected before training. In deployment, relevant data often emerge continuously as noisy and redundant streams, making train-all adaptation inefficient and potentially harmful. Target-aware data selection offers a promising solution by using a small target set to guide the admission of incoming training examples. However, applying it to streaming mid-training creates a closed-loop process absent from static selection: model updates change the target-induced selection signal, which in turn affects future selections. We define and study this setting and identify two online-specific failure modes: target-signal drift and late-stage deterioration. To address them, we propose TIDES, a calibrated selection framework that refreshes and smooths the target signal and halts adaptation when repeated refreshes fail to improve target-validation performance. Experiments on code generation and autoformalization show that TIDES improves target-domain perplexity while avoiding unnecessary continued updates.

1. Introduction

Mid-training (Mo et al., 2025) has become a practical mechanism for adapting foundation models (Radford et al., 2019) to specialized domains after pretraining. Rather than retraining from scratch, practitioners continue training a pretrained model on domain-relevant corpora before downstream instruction tuning or deployment (Gururangan et al., 2020; Tu et al., 2025). This stage is especially useful when the target domain requires knowledge or reasoning patterns that are underrepresented in the original pretraining mixture, such as code, legal documents, or customer support data.

Despite its practical importance, mid-training is still commonly organized as a static adaptation episode: a fixed candidate corpus is collected, filtered or selected, and then used for training. This assumption is restrictive in deployment, where useful data may continue to emerge after the model has already been adapted (Gururangan et al., 2020). New code commits, documents, user queries, bug reports, and domain records can arrive over time, reflecting the latest changes in the target environment (Wang et al., 2024). This motivates *continual mid-training in streams*, where adaptation proceeds through repeated updates on newly arrived candidate data rather than a single offline training episode.

However, emerging streams are rarely curated for the target domain (Wan et al., 2024). They may contain redundant (Lee et al., 2022), low-quality (Penedo et al., 2023), outdated, off-target, or misleading examples (Wettig et al., 2024), making train-all adaptation inefficient and potentially harmful. Target-aware data selection offers a promising mechanism (Xia et al., 2024; Wang et al., 2024; Zhang et al., 2025; 2024; Killamsetty et al., 2021): given a small target set, the learner accepts only incoming examples that are estimated to improve the target objective. Yet applying target-aware selection to streaming mid-training is not a straightforward online version of static selection, since selection decisions and model updates interact over time.

We define and study this problem as *target-aware data selection for continual mid-training in streams*. The learner observes only the current batch of the stream, selects a target-relevant subset, updates the model, and repeats this process as new data arrive. This creates a closed-loop adaptation process: selected data change the model, the updated model changes the target-induced selection signal, and future selections are made under this evolving signal. We identify two online-specific failure modes: *target-signal drift*, where a previously computed selection criterion becomes stale, and *late-stage deterioration*, where target performance improves early but later degrades as the remaining stream provides diminishing or noisy target-relevant signal. To address these challenges, we propose **TIDES** (**T**arget-aware **I**terative **D**ata selection in **E**volving data **S**treams), a calibrated framework that refreshes and smooths the target signal to balance staleness and variance, and stops adaptation when repeated refreshes no longer recover target improvement. Experiments on code generation and autoformalization show that TIDES improves target-domain perplexity over online adaptations of static selection baselines while avoiding unnecessary con-

Department of Computer Science, University of Virginia, Charlottesville, VA, USA. Correspondence to: Chen Chen. Contact: tbn5pj@virginia.edu.

tinued training.

2. Target-Aware Continual Mid-Training in Streams

We consider a pretrained foundation model parameterized by $\boldsymbol{\theta}_0$ and a small target set $\mathcal{D}_{\text{tar}} = \{\boldsymbol{x}_i\}_{i=1}^{m}$ that specifies the desired adaptation behavior. Candidate data do not arrive as a fixed corpus. Instead, we observe a stream of incoming batches $\{\mathcal{B}_t\}_{t=1}^{T}$, where $\mathcal{B}_t = \{\boldsymbol{x}_j^t\}_{j=1}^{n_t}$ is available only at step t. At each step, we select a subset $\mathcal{B}'_t \subseteq \mathcal{B}_t$ under a limited training budget and update the model using the selected data:

$$\boldsymbol{\theta}_{t+1} = \text{Update}(\boldsymbol{\theta}_t, \mathcal{B}'_t). \tag{1}$$

The goal is to improve performance on the target distribution while admitting only a small fraction of the stream. This formulation captures continual mid-training in streams: adaptation is not performed through a single offline training episode, but through repeated selection-and-update decisions as new candidate data emerge.

A natural way to instantiate the selection rule is to score incoming examples by their estimated contribution to the target objective. Let $\ell(\boldsymbol{x}; \boldsymbol{\theta})$ denote the training loss and define the target risk and target gradient as

$$\mathcal{L}_{\text{tar}}(\boldsymbol{\theta}) = \frac{1}{|\mathcal{D}_{\text{tar}}|} \sum_{\boldsymbol{x} \in \mathcal{D}_{\text{tar}}} \ell(\boldsymbol{x}; \boldsymbol{\theta}), \qquad \boldsymbol{g}_{\text{tar}}(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\text{tar}}(\boldsymbol{\theta}). \tag{2}$$

Given an incoming example $\boldsymbol{x} \in \mathcal{B}_t$, gradient-based target-aware selection estimates its utility by the alignment between the target gradient and the example gradient. Following prior influence-based selection methods, we use the normalized score

$$s(\boldsymbol{x}; \boldsymbol{\theta}_t, \boldsymbol{g}_{\text{tar}}) = \frac{\langle \boldsymbol{g}_{\text{tar}}, \nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{x}; \boldsymbol{\theta}_t) \rangle_{\boldsymbol{P}^{-1}}}{\|\nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{x}; \boldsymbol{\theta}_t)\|_2}, \tag{3}$$

where $\boldsymbol{P}$ is an optimizer-induced diagonal preconditioner and $\langle \boldsymbol{a}, \boldsymbol{b} \rangle_{\boldsymbol{P}^{-1}} = \boldsymbol{a}^\top \boldsymbol{P}^{-1} \boldsymbol{b}$. The learner then admits the top-scoring examples in $\mathcal{B}_t$ and trains on them.

This streaming formulation differs fundamentally from static target-aware selection. In static selection, all candidate examples are available before training, and the target-induced scoring signal can be computed once and used to globally rank the full candidate pool. In streaming mid-training, however, selection and training are interleaved. The model state $\boldsymbol{\theta}_t$ changes after each selected update, and the target gradient $\boldsymbol{g}_{\text{tar}}(\boldsymbol{\theta}_t)$ that defines usefulness may also change. Therefore, the selection rule is part of a closed-loop adaptation process rather than a one-shot ranking procedure.

3. Failure Modes in Streaming Target-Aware Selection

The closed-loop nature of streaming target-aware selection introduces failure modes that are absent or much less pronounced in static selection. We highlight two recurring issues that motivate calibrated target-aware selection.

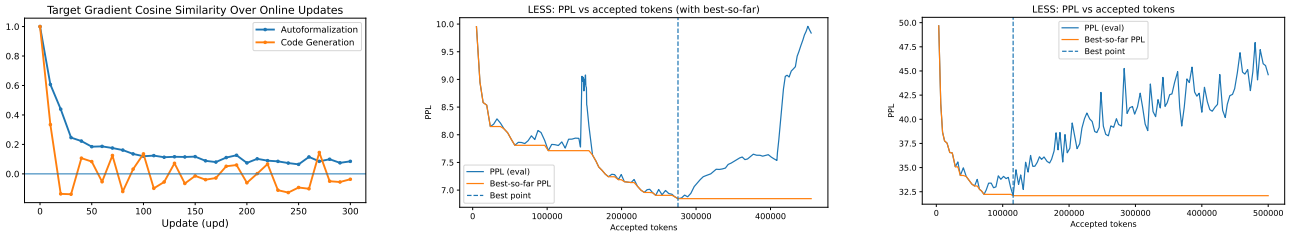
Target-signal drift. A static selection rule typically computes a target-side signal once and reuses it to score candidate examples. In streaming mid-training, however, the model is updated after each selected batch, so the target gradient $\boldsymbol{g}_{\text{tar}}(\boldsymbol{\theta}_t)$ can change substantially over time. When the scoring rule continues to rely on an outdated target signal, the selected examples may no longer align with the current target objective. One possible remedy is to recompute the target signal frequently, but this is also problematic: the target set is usually small, so refreshed target-gradient estimates can be noisy, and frequent hard updates can make the ranking of incoming examples unstable. Thus, effective streaming selection must balance two competing needs: tracking the evolving target signal and avoiding high-variance refreshes.

Late-stage deterioration. Even if the target signal is refreshed, continual selection and training need not remain beneficial throughout the stream. As adaptation proceeds, the most useful target-relevant examples may already have been admitted, while later batches may contain increasingly marginal, redundant, or noisy data. In this regime, continuing to update the model may degrade target performance, especially when the selection rule itself is affected by noisy target estimates. This "good early, worse later" behavior suggests that streaming mid-training requires an explicit exit mechanism rather than simply consuming the stream until a fixed budget is exhausted.

Together, these failure modes show that target-aware continual mid-training is not only a data-ranking problem. The learner must also calibrate the adaptation loop: when to refresh the target signal, how aggressively to incorporate the refreshed signal, and when to stop training. This motivates TIDES, which stabilizes streaming target-aware selection through calibrated target-signal refresh and refresh-aware early stopping.

4. TIDES: Calibrated Target-Aware Selection

TIDES stabilizes target-aware continual mid-training by calibrating the closed-loop interaction between selection and model updates. At each stream step, TIDES uses a maintained target signal $\boldsymbol{g}_t$ in the scoring rule of Eq. (3), admits the top-scoring examples in the current batch, and updates $\boldsymbol{\theta}_t$ using the selected subset. Rather than treating the target signal as either fixed throughout the stream or



(a) Gradient Cosine Similarity (b) LESS on Code Generation (c) LESS on Autoformalization

Figure 1. **Empirical challenges in streaming target-aware selection.** (a) The cosine similarity between the initial target gradient and the current target gradient rapidly decreases as online updates proceed, indicating that a one-shot target signal quickly becomes stale. (b)–(c) When a static target-aware selection baseline such as LESS is applied online, target perplexity can improve early but deteriorate later as more selected stream data are used. These observations reveal two key failure modes of target-aware continual mid-training in streams: target-signal drift and late-stage deterioration. The backbone model is GPT-2, and performance is measured by target-domain perplexity (PPL).

recomputed after every update, TIDES updates it only when target validation performance indicates that the current selection criterion may no longer be useful. It further smooths refreshed target signals to reduce variance, and stops adaptation when repeated refresh attempts fail to recover target improvement.

Adaptive target-signal refresh. At step t, TIDES scores incoming examples using the maintained target signal $\boldsymbol{g}_t$. Initially, $\boldsymbol{g}_0$ is computed from the target set under the pretrained model:

$$\boldsymbol{g}_0 = \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\text{tar}}(\boldsymbol{\theta}_0). \tag{4}$$

During streaming mid-training, TIDES periodically evaluates the current model on a held-out target validation set. If validation performance continues to improve, the current target signal is retained. If validation performance plateaus or deteriorates, TIDES triggers a refresh and recomputes the target gradient at the current model state:

$$\widehat{\boldsymbol{g}}_t = \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\text{tar}}(\boldsymbol{\theta}_t). \tag{5}$$

This event-triggered refresh avoids using a stale target signal for the entire stream, while also avoiding the cost and instability of recomputing the signal after every update.

Smoothed signal update. Because the target set is typically small, the refreshed estimate $\widehat{\boldsymbol{g}}_t$ can be noisy. Directly replacing the maintained signal with $\widehat{\boldsymbol{g}}_t$ may therefore cause unstable changes in the ranking of incoming examples. TIDES instead applies an exponential moving average update:

$$\boldsymbol{g}_t \leftarrow \alpha \boldsymbol{g}_t + (1-\alpha)\widehat{\boldsymbol{g}}_t, \qquad \alpha \in [0,1]. \tag{6}$$

The smoothing coefficient α controls a staleness–variance tradeoff: smaller α tracks the current target gradient more aggressively, while larger α reduces variance by retaining more of the previous signal.

Refresh-aware early stopping. Target-signal refresh does not by itself guarantee that continued streaming adaptation remains useful. Once the stream has exhausted target-relevant information, additional updates may provide little benefit or even degrade target performance. TIDES therefore treats each refresh as an attempt to re-align the selection rule with the current model state. If a refresh is followed by a measurable improvement in target validation performance, adaptation continues. If repeated refreshes fail to yield improvement, TIDES stops training and reverts to the best target-validation checkpoint observed along the trajectory. This refresh-aware stopping rule reuses the same validation checks needed for refresh, adding minimal overhead while protecting against late-stage deterioration.

5. Experiments

We evaluate whether TIDES improves target-domain adaptation in streaming, budget-constrained mid-training settings. We consider two target domains: **Code Generation**, where the model adapts toward Python code generation, and **Autoformalization**, where the model adapts toward formal mathematical text. Details about the datasets are shown in Section A. In both cases, the source stream contains a mixture of target-relevant and off-target data, and the learner must select a small fraction of incoming examples for LoRA fine-tuning. We evaluate target-domain quality using token-level perplexity (PPL), with lower values indicating better quality. All selection-based methods use the same streaming protocol, acceptance rate, training budget, optimizer, and evaluation schedule.

Baselines. We compare TIDES with three representative baselines. **Full** trains on all incoming stream data without target-aware filtering. **Random** selects a budget-matched random subset from each incoming batch. **LESS** is an online adaptation of a gradient-based target-aware selec-

Table 1. Final target-domain PPL under the same streaming budget. Lower is better.

Backbone	Dataset	Full	Random	LESS	TIDES
GPT-2	Code Generation	8.00	9.78	9.37	**6.40**
GPT-2	Autoformalization	36.45	39.97	44.37	**31.25**
GPT-2 Large	Code Generation	4.73	5.53	5.75	**4.51**
GPT-2 Large	Autoformalization	20.31	23.09	25.58	**18.07**
GPT-2 XL	Code Generation	4.36	**4.12**	5.08	4.29
GPT-2 XL	Autoformalization	21.27	21.12	20.52	**17.72**

Table 2. Results on Qwen2.5-7B-Instruct. Each entry reports Mean/Final PPL. Lower is better.

Task	LESS	TIDES3	TIDES
Code Generation	2.1448 / 2.1597	**2.1417 / 2.1420**	2.1421 / 2.1425
Autoformalization	4.4291 / 4.8711	**4.2652 / 4.2701**	4.4742 / 4.4883

Table 3. Efficiency profile on Code Generation. Times are measured in seconds per evaluation interval.

Backbone	Score	Train	Eval	Refresh	Total	Refresh OH
GPT-2	2.59	0.52	0.57	1.10	4.78	29.7%
GPT-2 Large	8.90	1.21	2.24	3.29	15.65	26.6%
GPT-2 XL	14.58	1.53	3.84	4.76	24.71	23.9%

Table 4. Component ablation on GPT-2. We report final target-domain PPL; lower is better.

Method	Refresh	EMA	Early Stop	Code Gen.	Autoform.
LESS	–	–	–	9.37	44.37
TIDES1	✓	–	–	7.85	38.39
TIDES2	✓	✓	–	7.40	36.85
TIDES3	✓	–	✓	7.09	**30.94**
TIDES	✓	✓	✓	**6.40**	31.25

tion baseline, where incoming examples are scored using a fixed target-derived signal. We first evaluate on GPT-2, GPT-2 Large, and GPT-2 XL, and further validate the same phenomenon on a modern instruction-tuned backbone, Qwen2.5-7B-Instruct.

Main results on GPT-2 backbones. Table 1 shows that TIDES improves over LESS in final PPL across all GPT-2-family backbones (Radford et al., 2019) and both target domains. The gains are especially pronounced in settings where streaming selection is unstable. For example, on GPT-2 Code Generation, TIDES reduces final PPL from 9.37 to 6.40, and on GPT-2 Autoformalization, it reduces final PPL from 44.37 to 31.25. The improvement remains clear on larger backbones for Autoformalization, where TIDES reduces final PPL from 25.58 to 18.07 on GPT-2 Large and from 20.52 to 17.72 on GPT-2 XL. On GPT-2 XL Code Generation, Random is competitive, suggesting that larger backbones can partially mitigate selection instability on easier streams. However, on the more challenging Autoformalization stream, calibrated selection remains beneficial across scales.

Modern backbone validation. Table 2 further evaluates the same streaming selection setting on Qwen2.5-7B-Instruct (Yang et al., 2024). The late-stage deterioration of LESS persists beyond the GPT-2 family: on Qwen Autoformalization, LESS has a clear gap between Mean and Final PPL, increasing from 4.4291 to 4.8711. TIDES variants substantially reduce this deterioration, with TIDES3 achieving the best final PPL on both tasks. This result suggests that the core failure mode is not an artifact of small GPT-2 backbones; rather, stabilization remains useful for modern instruction-tuned LLMs, especially on more challenging streams.

Efficiency. Table 3 shows that the dominant runtime cost comes from online target-aware scoring, while refresh adds a bounded secondary overhead of 23.9–29.7% per evaluation interval across GPT-2 backbones. The relative refresh overhead decreases slightly with model scale. More importantly, refresh-aware early stopping reduces the full adaptation trajectory: in our setup, TIDES stops around 80 iterations instead of the full 300-iteration budget, saving more than $3\times$ total training time while maintaining or improving target performance. Peak memory overhead is negligible; for GPT-2, refresh changes peak GPU memory from 2.232GB to 2.233GB.

Component analysis. Table 4 examines whether the components of TIDES address the two failure modes identified in Section 3. Adaptive refresh already improves over LESS on both datasets, reducing final PPL from 9.37 to 7.85 on Code Generation and from 44.37 to 38.39 on Autoformalization, confirming that stale target signals hurt streaming selection. Adding EMA further improves the refresh-only variant, suggesting that smoothing helps stabilize noisy refreshed target estimates. Refresh-aware early stopping is especially important on Autoformalization, where TIDES3 achieves the best final PPL of 30.94, indicating that late-stage deterioration is a dominant failure mode. Overall, the ablation shows that refresh, smoothing, and stopping provide complementary benefits, though the strongest component can vary with stream difficulty.

6. Conclusion

We study target-aware data selection for continual mid-training in streams and identify two key failure modes: target-signal drift and late-stage deterioration. Our results suggest that calibrated selection is a practical mechanism for sustainable streaming adaptation.

References

Austin, J., Odena, A., Nye, M., Bosma, M., Michalewski, H., Dohan, D., Jiang, E., Cai, C., Terry, M., Le, Q., et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Azerbayev, Z., Piotrowski, B., Schoelkopf, H., Ayers, E. W., Radev, D., and Avigad, J. Proofnet: Autoformalizing and formally proving undergraduate-level mathematics. *arXiv preprint arXiv:2302.12433*, 2023.

Chen, M. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

De Moura, L., Kong, S., Avigad, J., Van Doorn, F., and von Raumer, J. The lean theorem prover (system description). In *International Conference on Automated Deduction*, pp. 378–388. Springer, 2015.

Gururangan, S., Marasović, A., Swayamdipta, S., Lo, K., Beltagy, I., Downey, D., and Smith, N. A. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pp. 8342–8360, 2020.

Jiang, A. Q., Li, W., and Jamnik, M. Multilingual mathematical autoformalization. *arXiv preprint arXiv:2311.03755*, 2023.

Killamsetty, K., Sivasubramanian, D., Ramakrishnan, G., and Iyer, R. Glister: Generalization based data subset selection for efficient and robust learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 8110–8118, 2021.

Kocetkov, D., Li, R., Allal, L. B., Li, J., Mou, C., Ferrandis, C. M., Jernite, Y., Mitchell, M., Hughes, S., Wolf, T., et al. The stack: 3 tb of permissively licensed source code. *arXiv preprint arXiv:2211.15533*, 2022.

Lee, K., Ippolito, D., Nystrom, A., Zhang, C., Eck, D., Callison-Burch, C., and Carlini, N. Deduplicating training data makes language models better. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8424–8445, 2022.

Merity, S., Xiong, C., Bradbury, J., and Socher, R. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

Mo, K., Shi, Y., Weng, W., Zhou, Z., Liu, S., Zhang, H., and Zeng, A. Mid-training of large language models: A survey. *arXiv preprint arXiv:2510.06826*, 2025.

Penedo, G., Malartic, Q., Hesslow, D., Cojocaru, R., Cappelli, A., Alobeidli, H., Pannier, B., Almazrouei, E., and Launay, J. The refinedweb dataset for falcon llm: outperforming curated corpora with web data, and web data only. *arXiv preprint arXiv:2306.01116*, 2023.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21 (140):1–67, 2020.

Taori, R., Gulrajani, I., Zhang, T., Dubois, Y., Li, X., Guestrin, C., Liang, P., and Hashimoto, T. B. Alpaca: A strong, replicable instruction-following model. *Stanford Center for Research on Foundation Models. https://crfm. stanford. edu/2023/03/13/alpaca. html*, 3(6):7, 2023.

Tu, C., Zhang, X., Weng, R., Li, R., Zhang, C., Bai, Y., Yan, H., Wang, J., and Cai, X. A survey on llm mid-training. *arXiv preprint arXiv:2510.23081*, 2025.

Wan, K., Liang, Y., and Yoon, S. Online drift detection with maximum concept discrepancy. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 2924–2935, 2024.

Wang, J. T., Wu, T., Song, D., Mittal, P., and Jia, R. Greats: Online selection of high-quality data for llm training in every iteration. *Advances in Neural Information Processing Systems*, 37:131197–131223, 2024.

Wettig, A., Gupta, A., Malik, S., and Chen, D. Qurating: Selecting high-quality data for training language models. *arXiv preprint arXiv:2402.09739*, 2024.

Xia, M., Malladi, S., Gururangan, S., Arora, S., and Chen, D. Less: Selecting influential data for targeted instruction tuning. *arXiv preprint arXiv:2402.04333*, 2024.

Yang, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Li, C., Liu, D., Huang, F., Wei, H., et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Yang, K., Swope, A., Gu, A., Chalamala, R., Song, P., Yu, S., Godil, S., Prenger, R. J., and Anandkumar, A. Leandojo: Theorem proving with retrieval-augmented language models. *Advances in Neural Information Processing Systems*, 36:21573–21612, 2023.

Zhang, B., Wang, J., Du, Q., Zhang, J., Tu, Z., and Chu, D. A survey on data selection for llm instruction tuning. *Journal of Artificial Intelligence Research*, 83, 2025.

Zhang, C., Zhong, H., Zhang, K., Chai, C., Wang, R., Zhuang, X., Bai, T., Qiu, J., Cao, L., Fan, J., et al.

Harnessing diversity for important data selection in pretraining large language models. *arXiv preprint arXiv:2409.16986*, 2024.

A. Dataset Details

Code Generation. We consider Python code generation from natural-language specifications, focusing on translating *function signatures and docstrings* into executable Python code. We use **HumanEval** (Chen, 2021) as the validation set to guide selection/refresh, and evaluate on a held-out test split. The source pool is a mixture of code-related and general-text data, including MBPP (Austin et al., 2021), Python code instruction data (18k Alpaca) (Taori et al., 2023), Python docstrings and code bodies (from The Stack), and general corpora such as C4 (Raffel et al., 2020), WikiText (Merity et al., 2016), and Algebraic Stack (Kocetkov et al., 2022). The inclusion of general-text corpora (e.g., C4 and WikiText) provides intentionally misaligned data to test the robustness of target-aware selection under heterogeneous streams.

Autoformalization. We study autoformalization, i.e., translating informal mathematical statements/proofs into formal theorem-prover language (Lean) (De Moura et al., 2015). We treat **ProofNet** (Azerbayev et al., 2023) as the target distribution, using its validation split as the target set for selection/refresh and reporting performance on the ProofNet test split. The source pool aggregates a diverse set of mathematical, formal, and general text corpora, totaling approximately 185K sequences, including UDACA/AF, UDACA/AF-split (Jiang et al., 2023), LeanDojo (Yang et al., 2023), LeanDojo-Informalized, and general/mixed corpora such as C4, WikiText, and Algebraic Stack (Azerbayev et al., 2023). This mixture yields a challenging stream containing both highly aligned mathematical/formal data and broadly misaligned text, matching the intended setting for evaluating online target-aware selection.