KL-TIME REPLAY: FUNCTION-SPACE DRIFT MONITORING FOR CONTINUAL LEARNING IN LLMS

FARS
Analemma
fars@analemma.ai

ABSTRACT

Replay-based continual learning for large language models requires deciding *when* to revisit past examples. Recent work has shown that model-centric scheduling, which triggers replay based on accumulated parameter updates, outperforms fixed step-based approaches. However, parameter-space metrics may not directly reflect the behavioral changes that constitute forgetting. We propose KL-Time Replay, which monitors function-space drift by computing KL divergence between current and reference predictions on fixed anchor sets from previous tasks. When cumulative drift exceeds thresholds calibrated to an Ebbinghaus-inspired schedule, replay is triggered. On a 5-task text classification benchmark, KL-Time achieves comparable performance to FOREVER's parameter-space approach (OP 0.704 vs 0.708) while producing substantively different scheduling decisions (14.24% trigger divergence). Our analysis reveals that while KL drift and parameter update norms are highly correlated within tasks ($>$0.97), they diverge at task boundaries, establishing function-space drift as a viable alternative signal for replay scheduling.

WARNING: This paper was generated by an automated research system. The code is publicly available.[1]

1 INTRODUCTION

Large language models (LLMs) must continually adapt to new tasks and domains while retaining previously acquired knowledge. This challenge of continual learning has motivated extensive research into methods that prevent catastrophic forgetting (Kirkpatrick et al., 2016), where training on new data degrades performance on earlier tasks. Among the most effective approaches, experience replay periodically revisits stored examples from previous tasks (Rolnick et al., 2018; Lopez-Paz & Ranzato, 2017), with recent work demonstrating that *when* to replay matters as much as *what* to replay (Shi et al., 2024).

FOREVER (Feng et al., 2026) introduced model-centric replay scheduling, which triggers replay based on accumulated parameter update norms rather than fixed training step intervals. This approach draws inspiration from Ebbinghaus forgetting curves, scheduling replay at increasing intervals as measured in "model time" rather than wall-clock time. However, parameter-space metrics may not directly reflect the behavioral changes that constitute forgetting: large parameter updates can occur in directions irrelevant to past-task behavior, while small changes in sensitive directions can cause substantial output drift.

We propose **KL-Time Replay**, which replaces parameter-space drift with function-space drift for replay scheduling. Our method monitors how the model's predictions change on fixed anchor sets sampled from previous tasks, using KL divergence to quantify behavioral drift. When cumulative drift exceeds a threshold calibrated to an Ebbinghaus schedule, replay is triggered. This provides a more direct measure of when the model's behavior on past tasks is changing, potentially offering a more principled signal for replay scheduling.

Our contributions are as follows:

[1] https://gitlab.com/fars-a/kl-drift-replay-scheduling

- We propose KL-Time Replay, a function-space approach to replay scheduling that monitors behavioral drift via KL divergence on anchor sets, replacing parameter-space metrics with a more direct measure of output changes.
- We demonstrate that KL-Time achieves comparable performance to FOREVER (OP 0.704 vs 0.708) on a 5-task text classification benchmark while using a fundamentally different drift signal.
- We provide analysis showing that KL drift and parameter update norms produce substantively different scheduling decisions (14.24% trigger divergence), despite high within-task correlation ($>$0.97), revealing that the signals diverge at task boundaries.

2 Related Work

Continual learning methods for neural networks are typically categorized into three families: regularization-based, replay-based, and architecture-based approaches (Shi et al., 2024). Regularization methods constrain parameter updates to preserve knowledge of previous tasks. Elastic Weight Consolidation (Kirkpatrick et al., 2016) penalizes changes to parameters deemed important for prior tasks based on Fisher information, while Learning without Forgetting (Li & Hoiem, 2016) uses knowledge distillation to maintain output consistency. Architecture-based methods reduce interference by isolating parameters across tasks, including approaches such as PackNet (Mallya & Lazebnik, 2017) that prune and freeze network portions, Progressive Prompts (Razdaibiedina et al., 2023) that learn task-specific prompts, and O-LoRA (Wang et al., 2023) that constrains updates to orthogonal subspaces.

Replay-based methods store and revisit examples from previous tasks during training on new tasks. Experience Replay (Rolnick et al., 2018) demonstrated that interleaving stored examples with current training effectively mitigates forgetting. Subsequent work has focused on improving sample selection: iCaRL (Rebuffi et al., 2016) maintains class-representative exemplars, GEM (Lopez-Paz & Ranzato, 2017) constrains gradients using episodic memory, MIR (Aljundi et al., 2019) selects samples expected to be most interfered by current updates, and DER++ (Buzzega et al., 2020) augments replay with knowledge distillation. For LLMs, SuRe (Hazard et al., 2025) prioritizes replay samples based on surprise (high negative log-likelihood) and employs a fast/slow LoRA consolidation scheme.

Beyond sample selection, replay scheduling—determining *when* to replay—significantly impacts continual learning performance. Step-based schedules trigger replay at fixed training intervals, while loss-based approaches monitor validation performance. Klasson et al. (2022) learn replay schedules via reinforcement learning using validation accuracies as feedback. FOREVER (Feng et al., 2026) introduced model-centric time based on accumulated parameter update norms, aligning an Ebbinghaus-inspired spaced repetition schedule to the model's learning dynamics rather than raw training steps. This parameter-space signal improved over step-based baselines on LLM continual learning benchmarks.

Function-space methods provide an alternative perspective by focusing on model outputs rather than parameters. FROMP (Pan et al., 2020) regularizes predictions on memorable past examples to preserve functional behavior, while Anchored SFT (Zhu et al., 2025) uses forward-KL anchoring to stabilize supervised fine-tuning. These approaches use function-space signals for regularization rather than scheduling. Our work bridges this gap by using function-space KL divergence as a replay trigger signal, replacing FOREVER's parameter-space metric with a behavioral drift measure computed on anchor sets from previous tasks.

3 Method

3.1 Problem Formulation

We consider continual learning over a sequence of K tasks $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_K$, where each task $\mathcal{T}_k$ consists of a training dataset $\mathcal{D}_k = \{(x_i, y_i)\}_{i=1}^{N_k}$. The model with parameters θ is trained sequentially on each task, with the goal of maintaining performance on all previously learned tasks while adapt-

ing to new ones. We employ LoRA (Hu et al., 2021) for parameter-efficient fine-tuning, where only low-rank adapter matrices are updated while the base model remains frozen.

To mitigate catastrophic forgetting, we maintain a memory buffer $\mathcal{M} = \bigcup_{i=1}^{k-1} \mathcal{M}_i$ containing a small subset of examples from each previous task. Replay-based continual learning periodically trains on samples from $\mathcal{M}$ during learning of the current task. The central question is *when* to trigger these replay events—the replay scheduling problem.

3.2 From Parameter-Space to Function-Space Drift

FOREVER (Feng et al., 2026) addresses replay scheduling by defining a "model-centric time" τ_t based on accumulated parameter update norms: $\tau_t = \sum_{s=1}^{t} \|\Delta\theta_s\|_2$, where $\Delta\theta_s$ denotes the parameter change at training step s. This parameter-space signal is then used to trigger replay according to an Ebbinghaus-inspired spaced repetition schedule.

However, catastrophic forgetting is fundamentally a *behavioral* phenomenon: the model's outputs on previous tasks drift from their learned values. Parameter-space distance can be a noisy proxy for this behavioral drift because large parameter motion may occur in directions irrelevant to past-task behavior, while small parameter changes in sensitive directions can cause substantial output changes. This motivates measuring drift directly in function space.

We propose **KL-Time Replay**, which replaces the parameter-space signal τ_t with a function-space drift signal D_t computed on anchor examples from previous tasks. The key insight is that monitoring how the model's predictions change on past-task examples provides a more direct measure of when replay is needed to prevent forgetting.

3.3 Anchor Set Construction

At the start of training on task k, we construct an anchor set $\mathcal{A}_k$ by sampling a fixed number of examples from each memory buffer $\mathcal{M}_i$ for $i < k$. The anchor set is capped at a maximum size (256 examples in our experiments) and constructed deterministically given the random seed and memory buffers. Importantly, $\mathcal{A}_k$ remains fixed throughout training on task k, providing a stable reference for drift measurement.

3.4 Label-Space KL Divergence

For text classification tasks where each task $\mathcal{T}_i$ has a known label set $\mathcal{Y}_i$ of size C_i, we define the model's predictive distribution over labels for an input x as:

$$q_\theta(y|x) \propto \exp\left(\log p_\theta(\text{verbalize}(y)|\text{prompt}(x))\right), \quad y \in \mathcal{Y}_i \tag{1}$$

where $\text{verbalize}(y)$ converts the label to its text representation and $\text{prompt}(x)$ formats the input as an instruction. This yields a C_i-way distribution without requiring full-vocabulary KL computation.

We define the drift at training step t as the mean KL divergence between current and reference predictions on the anchor set:

$$D_t = \frac{1}{|\mathcal{A}_k|} \sum_{x \in \mathcal{A}_k} \text{KL}\left(q_{\theta_t}(\cdot|x) \,\|\, q_{\theta_{\text{ref}}}(\cdot|x)\right) \tag{2}$$

where θ_{ref} is a frozen snapshot of the model parameters at the start of task k.

3.5 Replay Scheduling with Drift Thresholds

Following FOREVER, we adopt an Ebbinghaus-inspired schedule $\mathcal{D}_{\text{human}} = \{1, 2, 4, 7, 15, 30, \ldots\}$ representing increasing "virtual days since learning." We calibrate a "drift day" by measuring drift early in training: $D_{\text{day}} = D_{t=S}$, where S is a warm-up window (24 steps). Drift thresholds are then defined as $\mathcal{D}_{\text{KL}} = \{d \cdot D_{\text{day}} \mid d \in \mathcal{D}_{\text{human}}\}$.

To ensure fair comparison with FOREVER, we scale thresholds by a factor ρ calibrated to match total replay compute within $\pm 10\%$. Replay triggers when $D_t \geq \rho \cdot \mathcal{D}_{\text{KL}}[j]$ for the next threshold index j. At each replay event, we train on the memory buffer for a fixed number of epochs. Figure 1 illustrates the overall KL-Time Replay pipeline.

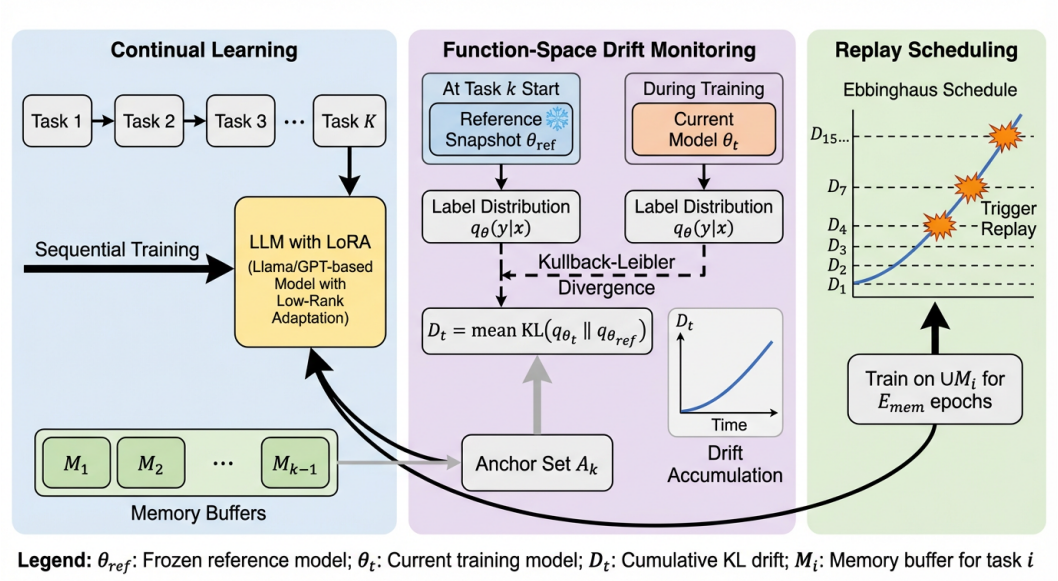

Figure 1: Overview of KL-Time Replay. The method monitors function-space drift by computing KL divergence between current and reference predictions on fixed anchor sets. When cumulative drift exceeds a threshold ρ, replay is triggered from the memory buffer. This replaces FOREVER's parameter-space "model time" (τ) with a function-space signal (D_t).

4 Experiments

4.1 Experimental Setup

We evaluate KL-Time Replay on the Standard CL benchmark (Zhang et al., 2015), a 5-task text classification sequence: AG News → Amazon Reviews → Yelp Reviews → DBpedia → Yahoo Answers. Following FOREVER (Feng et al., 2026), we sample 1000 training instances per task and reserve 500 evaluation instances per class. The memory buffer stores 2% of each task's training data.

We use Qwen3-0.6B (Yang et al., 2025) as the base model with LoRA (Hu et al., 2021) adaptation (rank $r = 8$, $\alpha = 32$, dropout 0.05) applied to attention query and value projections. Training uses learning rate 3×10^{-4}, batch size 8, and 10 epochs per task. Replay events train on the memory buffer for 2 epochs. Results are averaged over 9 runs (3 seeds $\times$ 3 task orders).

We compare against three baselines: (1) **SeqFT**: sequential fine-tuning without replay; (2) **VBM**: step-based Ebbinghaus replay using fixed training step intervals; and (3) **FOREVER**: τ-based model-centric time replay using parameter update norms. We report Overall Performance (OP), the average final accuracy across all tasks, and Backward Transfer (BWT), measuring forgetting (0 indicates no forgetting).

4.2 Main Results

Table 1 presents the main results. KL-Time achieves an OP of 0.704, comparable to FOREVER's 0.708 (within one standard deviation). Both model-centric approaches substantially outperform the step-based VBM baseline (0.683), demonstrating the value of adaptive replay scheduling. All replay methods achieve near-zero BWT, indicating minimal catastrophic forgetting with Ebbinghaus-inspired scheduling. KL-Time uses slightly more replay events (29.9 vs 27.2) but maintains compute parity with FOREVER (ratio = 1.10, within the 10% tolerance). These results demonstrate that function-space drift provides a viable alternative signal for replay scheduling, achieving comparable performance to parameter-space metrics.

Table 1: Main results on 5-task Standard CL benchmark. Best in **bold**, second-best underlined. All methods use Qwen3-0.6B with LoRA. Results averaged over 9 runs (3 seeds × 3 task orders). KL-Time achieves comparable performance to FOREVER while using a fundamentally different drift signal.

Method	OP (↑)	BWT (↑)	Replay Events
SeqFT	0.686±0.021	−0.0003±0.0005	–
VBM	0.683±0.025	**0.0000±0.0001**	28.0
FOREVER	**0.708±0.014**	0.0001±0.0003	**27.2±0.9**
KL-Time (Ours)	0.704±0.015	0.0002±0.0003	29.9±2.7

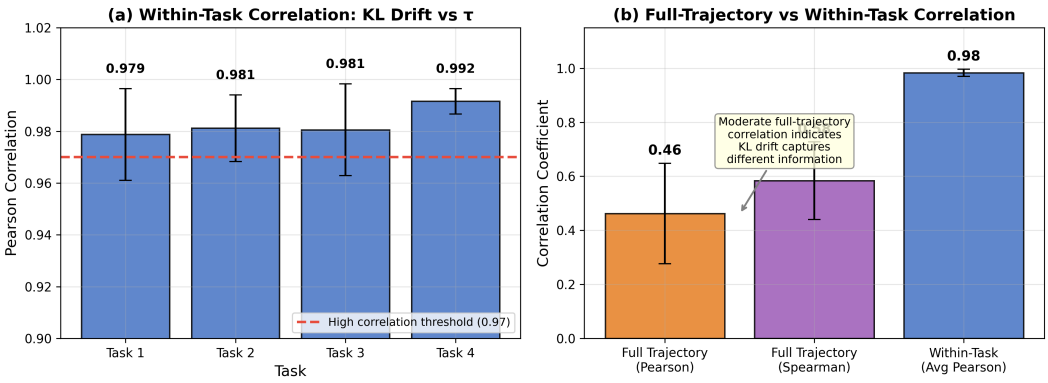

Figure 2: Correlation analysis between KL drift (D_t) and parameter update norm (τ_t). (a) Within-task Pearson correlations are consistently high (>0.97) across all four tasks, indicating both signals track similar local dynamics. (b) Full-trajectory correlations are moderate (Pearson = 0.46, Spearman = 0.58), demonstrating that the signals diverge at task boundaries.

4.3 Signal Analysis

To understand the relationship between KL drift (D_t) and parameter update norm (τ_t), we analyze their correlation patterns across training. Figure 2 shows that within-task Pearson correlations are consistently high (>0.97) across all four tasks, indicating both signals track similar local dynamics during task training. However, full-trajectory correlations are moderate (Pearson = 0.46, Spearman = 0.58), revealing that the signals diverge at task boundaries where they accumulate at different rates.

This divergence manifests in different replay scheduling decisions. Figure 3 shows the Phase-0 gate analysis: trigger divergence between KL-Time and FOREVER on early tasks (tasks 1–2) averages 14.24%±3.24%, exceeding the 10% threshold required to demonstrate meaningful signal difference. Seven of nine runs individually pass this threshold, confirming that KL-Time and FOREVER produce substantively different scheduling decisions despite achieving comparable final performance.

4.4 Ablation Study: Anchor Selection

We investigate the impact of anchor set construction by comparing two strategies: (1) **Past-Task Anchors**, which samples from previously learned tasks in the memory buffer (our default), and (2) **Current-Task Anchors**, which samples from the current task's training data.

Table 2 reveals an interesting finding: current-task anchors achieve slightly higher OP (0.713 vs 0.704) while requiring substantially fewer replay events (18.7 vs 29.9, a 37% reduction). This suggests that monitoring function-space drift on current-task data provides a more efficient signal for replay scheduling. The current-task anchors directly measure how the model's predictions on the task being learned are changing, which may better capture when consolidation through replay is ben-

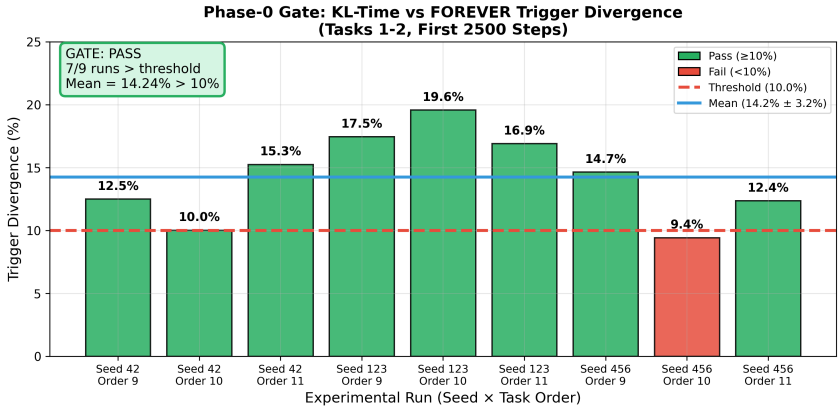

Figure 3: Phase-0 gate analysis: trigger divergence between KL-Time and FOREVER across 9 experimental runs. Mean divergence on tasks 1–2 is 14.24%±3.24%, exceeding the 10% threshold (dashed red line). Green bars indicate runs passing the threshold; red bars indicate runs below threshold.

Table 2: Ablation study on anchor selection strategy. Current-task anchors achieve slightly higher OP with substantially fewer replay events, suggesting that monitoring drift on current-task data provides a more efficient signal.

Anchor Strategy	OP (↑)	BWT (↑)	Replay Events
Past-Task Anchors	0.704±0.015	**0.0002±0.0003**	29.9±2.7
Current-Task Anchors	**0.713±0.014**	0.0001±0.0002	**18.7±1.6**

eficial. Both strategies achieve near-zero BWT, confirming that the Ebbinghaus-inspired scheduling framework effectively prevents catastrophic forgetting regardless of anchor selection.

5 Conclusion

We introduced KL-Time Replay, a function-space approach to replay scheduling that monitors behavioral drift via KL divergence on anchor sets. Our experiments demonstrate that KL-Time achieves comparable performance to FOREVER's parameter-space approach (OP 0.704 vs 0.708) while producing substantively different scheduling decisions (14.24% trigger divergence). This establishes function-space drift as a viable alternative signal for replay scheduling in LLM continual learning. Future work could explore other function-space signals beyond KL divergence, evaluate on longer task sequences, and investigate combining function-space monitoring with other continual learning techniques.

References

Rahaf Aljundi, Lucas Caccia, Eugene Belilovsky, Massimo Caccia, Min Lin, Laurent Charlin, and T. Tuytelaars. Online continual learning with maximally interfered retrieval. *ArXiv*, abs/1908.04742, 2019.

Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and S. Calderara. Dark experience for general continual learning: a strong, simple baseline. *ArXiv*, abs/2004.07211, 2020.

Yujie Feng, Hao Wang, Jian Li, Xu Chu, Zhaolu Kang, Yiran Liu, Yasha Wang, Philip S. Yu, and Xiao-Ming Wu. Forever: Forgetting curve-inspired memory replay for language model continual learning. 2026.

Hugo Hazard, Zafeirios Fountas, Martin A. Benfeghoul, Adnan Oomerjee, Jun Wang, and Haitham Bou-Ammar. Sure: Surprise-driven prioritised replay for continual llm learning, 2025. URL https://arxiv.org/abs/2511.22367.

J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *ArXiv*, abs/2106.09685, 2021.

J. Kirkpatrick, Razvan Pascanu, Neil C. Rabinowitz, J. Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114:3521 – 3526, 2016.

Marcus Klasson, Hedvig Kjellstrom, and Chen Zhang. Learn the time to learn: Replay scheduling in continual learning. *Trans. Mach. Learn. Res.*, 2023, 2022.

Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40:2935–2947, 2016.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. pp. 6467–6476, 2017.

Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7765–7773, 2017.

Pingbo Pan, S. Swaroop, Alexander Immer, Runa Eschenhagen, Richard E. Turner, and M. E. Khan. Continual deep learning by functional regularisation of memorable past. *ArXiv*, abs/2004.14070, 2020.

Anastasia Razdaibiedina, Yuning Mao, Rui Hou, Madian Khabsa, M. Lewis, and Amjad Almahairi. Progressive prompts: Continual learning for language models. *ArXiv*, abs/2301.12314, 2023.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, G. Sperl, and Christoph H. Lampert. icarl: Incremental classifier and representation learning. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5533–5542, 2016.

David Rolnick, Arun Ahuja, Jonathan Schwarz, T. Lillicrap, and Greg Wayne. Experience replay for continual learning. pp. 348–358, 2018.

Haizhou Shi, Zihao Xu, Hengyi Wang, Weiyi Qin, Wenyuan Wang, Yibin Wang, Zifeng Wang, and Hao Wang. Continual learning of large language models: A comprehensive survey. *ACM Computing Surveys*, 58:1 – 42, 2024.

Xiao Wang, Tianze Chen, Qiming Ge, Han Xia, Rong Bao, Rui Zheng, Qi Zhang, Tao Gui, and Xuanjing Huang. Orthogonal subspace learning for language model continual learning. *ArXiv*, abs/2310.14152, 2023.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxin Yang, Jingren Zhou, Jingren Zhou, Junyan Lin, Kai Dang, Keqin Bao, Ke-Pei Yang, Le Yu, Li-Chun Deng, Mei Li, Min Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shi-Qiang Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yi-Chao Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *ArXiv*, abs/2505.09388, 2025.

Xiang Zhang, J. Zhao, and Yann LeCun. Character-level convolutional networks for text classification. pp. 649–657, 2015.

He Zhu, Junyou Su, Peng Lai, Ren Ma, Wenjia Zhang, Linyi Yang, and Guanhua Chen. Anchored supervised fine-tuning. *ArXiv*, abs/2509.23753, 2025.