SAGE: ONLINE CONTINUAL KNOWLEDGE DISTILLATION FOR SMALL LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Small language models are increasingly deployed in resource-constrained settings because of their computational efficiency, yet their limited capacity often leads to performance degradation when they encounter previously unseen data. Although knowledge distillation can transfer capabilities from larger teacher models, conventional approaches are typically confined to offline training and therefore cannot provide continuous supervision after deployment. Here, we introduce SAGE, an online continual distillation framework that enables a deployed small language model to selectively acquire knowledge from a teacher model in streaming environments. SAGE comprises three coordinated modules. The Trigger identifies inputs beyond the student's competence boundary by combining teacher–student discrepancies without requiring labels. The Buffer uses streaming clustering and stability filtering to organize triggered samples into semantically coherent knowledge regions. Finally, the Debugger performs a two-stage search over low-rank parameter updates to derive effective adaptation configurations for targeted knowledge injection. Experiments show that SAGE substantially improves both sample efficiency and adaptation accuracy, providing a practical framework for continuously enhancing small language models after deployment.

1 INTRODUCTION

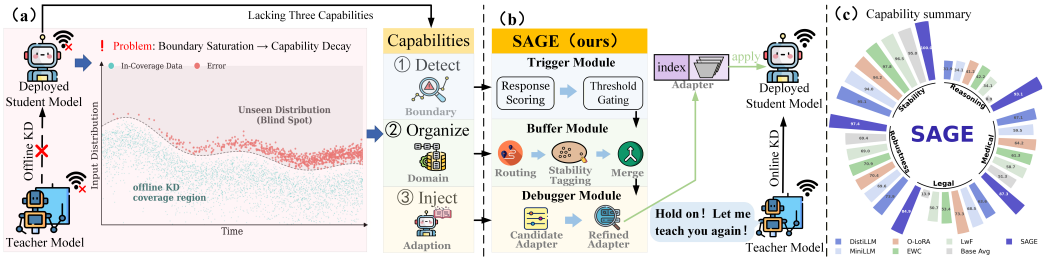

Figure 1: Motivation of Online Continual Knowledge Distillation. (a) Gap. Offline KD leads to boundary saturation, degrading performance. (b) SAGE (Ours). An online loop with three modules: Trigger (detect), Buffer (organize), and Debugger (inject).(c) Gains. SAGE resolves boundary saturation via online KD, improving performance.

Large language models (LLMs) have achieved remarkable performance on reasoning and knowledge-intensive tasks Brown et al. (2020); Touvron et al. (2023); Bai et al. (2023). However, their substantial computational and memory requirements hinder deployment in resource-constrained environments Kaplan et al. (2020); Dettmers et al. (2022; 2023). Knowledge distillation (KD) offers a practical solution by transferring capabilities from large teacher models to compact student models Hinton et al. (2015), enabling small language models (SLMs) to deliver competitive performance with substantially lower inference costs Sanh (2019). Developing effective KD techniques is therefore critical to making language-model broadly accessible under real-world resource constraints.

Considerable progress has been made in distilling capable SLMs, including conventional response-level distillation and more recent methods such as MiniLLM Gu et al. (2024) and DistillM Ko et al. (2024; 2025). Nevertheless, these methods typically treat distillation as a one-time, offline procedure:

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.

the student learns from a fixed training distribution and receives no further teacher supervision after deployment. Consequently, its performance may deteriorate as incoming data move beyond the distribution covered during training Koh et al. (2021). We refer to this phenomenon as *boundary saturation*: the student remains fixed while evolving inputs progressively exceed its competence boundary, leading to capability degradation (Fig.1(a)).

Continual learning methods provide a potential mechanism for post-deployment adaptation, but existing approaches do not directly address this problem. Regularization-based methods such as Elastic Weight ConsolidationKirkpatrick et al. (2017) preserve prior knowledge by constraining parameter updates, but introduce no external supervision Parisi et al. (2019). Replay-based methods Lopez-Paz & Ranzato (2017); Rolnick et al. (2019) depend on stored examples and incur substantial memory costs, while remaining agnostic to the student's competence boundary. Distillation-based continual learning methods, such as Learning without Forgetting Li & Hoiem (2017), also operate predominantly within predefined, offline training stages De Lange et al. (2021). Thus, existing methods neither continuously identify emerging deficiencies nor selectively distil the knowledge required to correct them. This raises a central question: Can a deployed SLM continually distil knowledge from a larger teacher in an online data stream?

We address this question with SAGE, an online continual distillation framework that enables SLMs to selectively acquire teacher knowledge after deployment (Fig. 1(b)). The central idea of SAGE is to transform teacher supervision from a static, one-time training resource into an adaptive service that is invoked only when and where the student requires assistance. Rather than indiscriminately distilling every incoming sample, SAGE detects inputs that exceed the student's current competence, consolidates related deficiencies into stable knowledge regions, and injects the corresponding teacher knowledge through targeted parameter updates. This design decomposes online continual distillation into three essential capabilities: *Detect*, identifying inputs beyond the student's competence boundary; *Organize*, consolidating related deficiencies into coherent and stable knowledge regions; and *Inject*, efficiently incorporating targeted teacher supervision without retraining the entire model.

SAGE implements these capabilities through three coordinated modules. First, the *Trigger* detects competence-boundary violations by aggregating lexical, semantic, and confidence-based discrepancies between the teacher and student into a label-free anomaly score. Second, the *Buffer* organizes triggered samples into semantically coherent knowledge regions through streaming clustering. A stability-filtering mechanism distinguishes persistent deficiencies from transient noise, ensuring that adaptation is initiated only when a knowledge region has sufficiently converged. Third, the *Debugger* determines how the student should be updated by conducting a two-stage search over low-rank adaptation configurations. It first explores the hyperparameter space at a coarse granularity and then performs fine-grained local refinement around promising configurations. Together, these modules form a closed-loop distillation process that continuously diagnoses, organizes, and corrects the student's emerging deficiencies under evolving streaming conditions.

We evaluate SAGE across language models of different sizes and families and examine five capability dimensions: stability, robustness, legal knowledge, medical knowledge, and reasoning (Fig. 1(c)). Across these settings, SAGE consistently outperforms the evaluated baselines in both adaptation accuracy and sample efficiency. Moreover, performance improves steadily over successive distillation cycles, indicating that SAGE supports cumulative capability acquisition rather than one-shot adaptation. These results establish online continual distillation as an effective approach for maintaining and extending the capabilities of deployed SLMs under distribution shift.

Our main contributions are threefold:

- We formulate *online continual distillation* as a post-deployment learning problem and identify boundary saturation as a key limitation of conventionally distilled SLMs operating on evolving data streams.
- We introduce SAGE, a closed-loop framework that detects competence-boundary violations, organizes emerging deficiencies into stable knowledge regions, and injects targeted teacher knowledge through efficient low-rank adaptation.
- We conduct extensive experiments across multiple model families, scales, and capability dimensions, demonstrating that SAGE improves adaptation accuracy and sample efficiency while enabling cumulative gains over repeated distillation cycles.

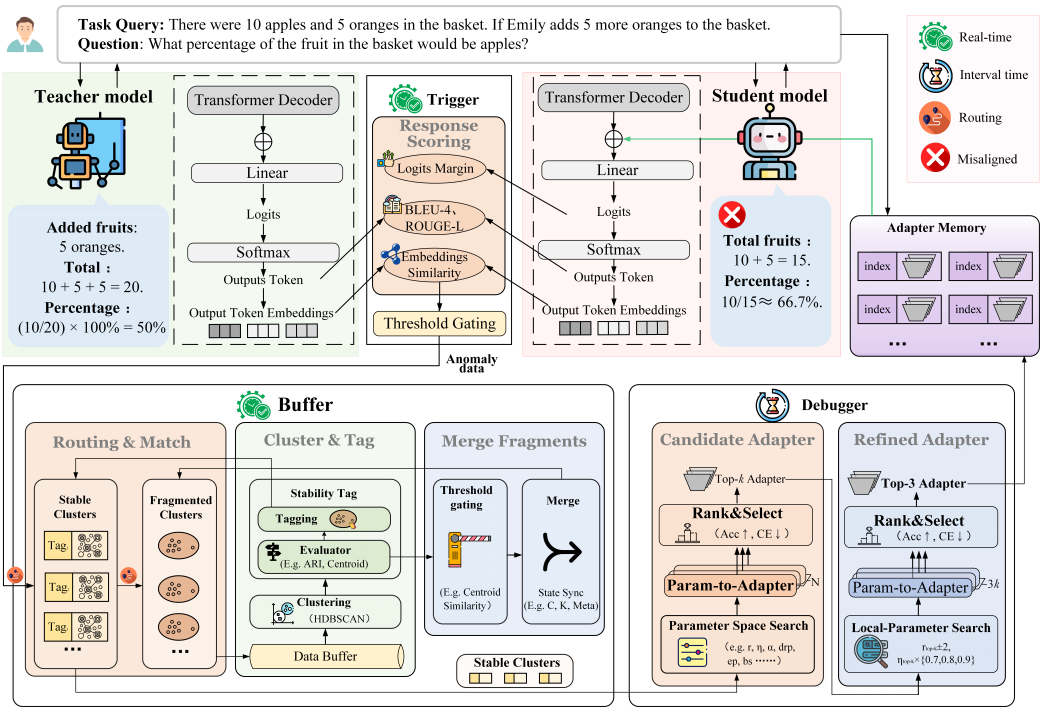

Figure 2: Overview of our SAGE framework. The Trigger module detects anomaly data by comparing predictions between student model and teacher model, injects data into the Buffer module for clustering, and commits the clusters to the Debugger for fine-tuning. The resulting adapter is finally applied to improve student model's performance.

2 Related Work

Knowledge Distillation. Knowledge distillation (KD) transfers capabilities from a high-capacity teacher to a compact student. Existing approaches can be broadly categorized into response-based methods, which align the output distributions of the teacher and student Hinton et al. (2015), and feature-based methods, which transfer knowledge through intermediate representations Srivastava et al. (2015). Recent studies have extended KD to large language models, with an emphasis on preserving their generative and reasoning capabilities in smaller models. Representative approaches, including DistillM Ko et al. (2024; 2025) and MiniLLM Gu et al. (2024), develop specialized objectives and optimization strategies for generative language-model distillation. Related lines of research have explored retrieval-augmented generation Lewis et al. (2020); Liu et al. (2024); Borgeaud et al. (2022) and long-context modeling Wang et al. (2023a), which enhance the knowledge accessible to compact models and are largely complementary to distillation.

Continual Learning. Continual learning aims to acquire knowledge from sequential data while mitigating catastrophic forgetting. Regularization-based methods, such as EWC Kirkpatrick et al. (2017), constrain important parameter updates, whereas replay-based methods, such as GEM Lopez-Paz & Ranzato (2017), preserve past examples at storage cost. Distillation-based approaches, including Learning without Forgetting Li & Hoiem (2017), retain prior behavior using predictions from previous models. Recent language-model studies have explored implicit or explicit memory mechanisms Geva et al. (2021); Berges et al. (2024); Li et al. (2025); Yang et al. (2024); Wang et al. (2024) and self-generated supervision for model improvement Huang et al.; Prasad et al.; Shafayat et al. (2025).

As discussed, existing KD methods predominantly transfer knowledge through predefined offline training, whereas continual learning methods generally adapt to incoming data without sustained supervision from a stronger teacher. Neither paradigm directly addresses how a deployed student can identify emerging capability deficiencies and selectively acquire the required knowledge from a teacher in real time. Our work bridges this gap through *online continual knowledge distillation*, which continuously detects inputs beyond the student's competence boundary, organizes recurring failures into stable knowledge regions, and injects targeted teacher supervision during deployment.

3 METHOD

Overview. SAGE realizes online continual knowledge distillation through a closed loop with three capabilities: *Detect*, *Organize*, and *Inject*, implemented by the Trigger, Buffer, and Debugger modules, respectively, as illustrated in Fig. 2. At each time step t, the Trigger $\mathcal{T}$ evaluates an incoming input x^t against the current student M^t to determine whether it exceeds the student's competence boundary. In-boundary inputs are processed normally, while anomalous inputs are paired with teacher responses and sent to the Buffer $\mathcal{B}$, which incrementally organizes them into semantically coherent regions and monitors temporal stability. Once a region stabilizes, the Debugger $\mathcal{U}$ searches for an effective low-rank adaptation configuration and distils the corresponding teacher knowledge into the student. This loop enables selective updates targeting persistent capability deficiencies rather than transient or isolated failures. We summarize this process as

$$A^t = \mathcal{U}\big(\mathcal{B}\big(\mathcal{T}(x^t; M^t, T)\big)\big), \qquad M^{t+1} = M^t \oplus A^t, \tag{1}$$

where T denotes the teacher model, A^t is the adapter produced at time t, and $\oplus$ denotes adapter injection. If no stable knowledge region is identified, A^t is null and the student remains unchanged.

3.1 TRIGGER: DETECTING COMPETENCE-BOUNDARY VIOLATIONS

The Trigger determines when teacher supervision should be converted into a student update. Relying on one discrepancy measure can be unreliable: token confidence does not necessarily reflect response correctness, lexical overlap may overlook semantically equivalent answers, and semantic similarity alone may obscure local errors. We therefore characterize teacher-student discrepancies at three complementary levels: token-level confidence, sequence-level agreement, and semantic consistency.

Given an input x^t, the teacher and student generate responses y_T^t and y_S^t, respectively. We first estimate the student's token-level confidence using the logit margin. Let $\mathbf{z} \in \mathbb{R}^V$ denote the student logits for the predicted token, where V is the vocabulary size, and let $z_{(1)}$ and $z_{(2)}$ be the largest and second-largest logits. The logit margin is

$$LM(x^t) = z_{(1)}(x^t) - z_{(2)}(x^t). \tag{2}$$

A larger margin indicates that the student assigns substantially greater probability to its preferred token and is therefore more confident in its prediction.

We next compare the complete teacher and student responses using BLEU and ROUGE-L. BLEU captures local lexical agreement through n-gram precision. We use BLEU-4, defined as

$$B_4 = BP \cdot \exp\left(\sum_{n=1}^{4} w_n \log p_n\right), \tag{3}$$

where p_n is the modified n-gram precision, w_n is its corresponding weight, and BP is the brevity penalty (Papineni et al., 2002). In parallel, ROUGE-L measures global sequence agreement through the longest common subsequence (Lin, 2004). Let P_{LCS} and R_{LCS} denote the resulting precision and recall. We use their harmonic mean:

$$F_{\mathrm{L}} = \frac{2P_{\mathrm{LCS}}R_{\mathrm{LCS}}}{P_{\mathrm{LCS}} + R_{\mathrm{LCS}}}. \tag{4}$$

Lexically different responses may nevertheless express the same meaning. To capture this case, we encode y_S^t and y_T^t as $\mathbf{e}_S^t$ and $\mathbf{e}_T^t$, respectively, and compute their cosine similarity (Lin et al., 2017):

$$ES(x^t) = \frac{\mathbf{e}_S^t \cdot \mathbf{e}_T^t}{\|\mathbf{e}_S^t\|_2 \|\mathbf{e}_T^t\|_2}. \tag{5}$$

These four signals are normalized and combined into a single anomaly score:

$$\mathcal{AS}^t = \sum_{i=1}^{4} w_i \left(1 - s_i^t\right), \qquad \sum_{i=1}^{4} w_i = 1, \tag{6}$$

where

$$s_1^t = \min\left\{\frac{LM(x^t)}{LM_m}, 1\right\}, \qquad s_2^t = B_4, \qquad s_3^t = F_{\mathrm{L}}, \qquad s_4^t = ES(x^t). \tag{7}$$

Here, LM_m is the normalization upper bound for the logit margin, set to 5.0 in our experiments, and w_i controls the contribution of each signal. Higher values of $\mathcal{AS}^t$ indicate lower student confidence or greater disagreement with the teacher.

Finally, a threshold $\tau_{\mathcal{T}}$ converts the anomaly score into a gating decision:

$$g^t = \begin{cases} 1, & \mathcal{AS}^t \geq \tau_{\mathcal{T}}, \\ 0, & \mathcal{AS}^t < \tau_{\mathcal{T}}. \end{cases} \tag{8}$$

When $g^t = 0$, the input is considered to lie within the student's current competence boundary. When $g^t = 1$, the tuple (x^t, y_S^t, y_T^t) is treated as an anomalous teacher–student pair and forwarded to the Buffer for subsequent organization and adaptation.

3.2 Buffer: Organizing Stable Knowledge Regions

A single anomalous sample does not necessarily justify updating the student, as it may reflect noise, ambiguity, or an isolated failure. The Buffer therefore accumulates related anomalies and converts them into stable knowledge regions that provide reliable adaptation targets. This process combines online routing, streaming clustering, and temporal stability estimation.

Each incoming anomaly is assigned to a coarse knowledge scope and compared with stable clusters previously identified within that scope. Let $\mathbf{e}_x$ denote the embedding of x^t, $\mathbf{e}c$ the centroid embedding of a candidate cluster $\mathcal{C}c^t$, and kx^t and $k\mathcal{C}_c^t$ their keyword sets. We measure their compatibility as

$$\mathrm{Match}(x^t, \mathcal{C}_c^t) = \alpha \cos(\mathbf{e}_x, \mathbf{e}_c) + (1-\alpha) \frac{\left| k_{x^t} \cap k_{\mathcal{C}_c^t} \right|}{\max\left(|k_{x^t}|, |k_{\mathcal{C}_c^t}| \right)}, \tag{9}$$

where $\alpha \in [0, 1]$ balances semantic similarity and keyword overlap. If the highest matching score exceeds a routing threshold τ_{match}, the sample is assigned to the corresponding cluster. Otherwise, it is retained as a new fragment. When no stable cluster is available, the sample is stored until sufficient evidence has accumulated for clustering.

Once the number of buffered samples in a knowledge scope reaches a predefined threshold, we apply HDBSCAN (Campello et al., 2013) to discover dense semantic regions without predefining the number of clusters. As the buffer evolves continuously, clusters from a single pass may be transient. Updating the student on such unstable regions can produce noisy or overly specialized adapters. We therefore compare successive clustering results using both structural and semantic criteria.

The structural criterion is the Adjusted Rand Index (ARI), which measures the consistency of cluster assignments across two clustering passes:

$$ARI = \frac{RI - \mathbb{E}[RI]}{\max(RI) - \mathbb{E}[RI]}, \tag{10}$$

where RI is the Rand Index computed over samples shared by both passes. The semantic criterion measures displacement of corresponding centroids. For a cluster observed at $t - m$ and t, we define

$$Emb_C = \cos\left(\mathbf{e}_c^{t-m}, \mathbf{e}_c^t\right). \tag{11}$$

A cluster is marked as stable when either its assignment structure or semantic center remains sufficiently consistent:

$$s_i^t = \mathbb{I}\left[ARI_i^t > \tau_{ARI} \ \vee \ Emb_{C,i}^t > \tau_{Emb} \right], \tag{12}$$

where $s_i^t \in \{0, 1\}$ is the stability label of cluster $\mathcal{C}_i^t$. These labels are retained and used to route subsequent inputs, allowing persistent knowledge regions to grow incrementally.

Small fragments may remain after clustering because their samples are insufficiently dense to form independent regions. Accumulating such fragments indefinitely would increase routing ambiguity and dilute otherwise coherent clusters. When the number of fragments exceeds a threshold, we compare their centroid embeddings and merge fragments $\mathcal{F}_i^t$ and $\mathcal{F}_j^t$ whenever

$$\cos\left(\mathbf{e}_{\mathcal{F}_i^t}, \mathbf{e}_{\mathcal{F}_j^t}\right) > \tau_{\mathrm{merge}}. \tag{13}$$

After each merge, the centroid, keywords, and cluster metadata are updated synchronously. The procedure terminates when no pair satisfies the merging criterion. Through this process, the Buffer transforms a stream of individual anomalies into compact, coherent, and temporally stable knowledge regions suitable for targeted distillation.

3.3 DEBUGGER: INJECTING TARGETED TEACHER KNOWLEDGE

Table 1: Overall Performance Comparison across GSM8K, PubMedQA and LexGlue. ↑ / ↓ indicate higher/lower is better. GT/GPT denote teacher signals using Ground-truth/ChatGPT.

Model	Method	GSM8K (%)			PubMedQA (%)		LexGlue (%)	
		EM↑	MAE↓	NER↑	Acc↑	F1↑	Acc↑	F1↑
Qwen2.5-7B	Base Model	17.84	4144	100	50.00	33.33	18.23	5.62
	LoRA	66.04	192	100	49.79	66.48	60.84	33.63
	CA-LoRA	69.93	2096	100	80.59	80.48	72.87	72.58
	EWC	74.15	811	99.80	72.75	51.59	74.95	74.93
	LwF	53.91	1079	100	64.89	59.77	78.67	81.26
	O-LoRA	81.36	9584	100	65.60	58.18	72.20	72.16
	MiniLLM	61.92	366	100	69.21	68.77	63.91	63.32
	DistiLLM	44.69	15936	100	67.42	66.93	67.96	67.95
	SAGE (Ours) GT	**91.32**↑	**45**↓	92.38	**89.12**↑	**87.56**↑	**92.00**↑	**92.13**↑
	SAGE (Ours) GPT	86.83	14854	87.37	73.25	74.66	79.81	78.45
Llama2-7B	Base Model	2.22	2258	5.09	50.60	34.65	12.53	13.19
	LoRA	7.44	2079	99.79	66.60	64.13	35.29	35.29
	CA-LoRA	10.64	1096	99.80	84.80	84.70	72.21	75.50
	EWC	23.38	1451	99.75	62.40	60.12	42.15	41.80
	LwF	20.00	2310	99.70	58.20	56.45	39.80	38.50
	O-LoRA	11.54	2313	99.00	61.26	61.61	71.39	71.43
	MiniLLM	11.02	1797	100	69.23	70.55	68.91	68.23
	DistiLLM	25.55	1189	99.60	67.32	66.91	68.33	68.33
	SAGE (Ours) GT	**99.80**↑	**0.01**↓	**100**↑	**95.70**↑	**95.37**↑	**87.14**↑	**87.63**↑
	SAGE (Ours) GPT	88.65	425	100	84.73	83.67	77.22	77.87
Mistral-7B	Base Model	6.67	3174	10.24	53.20	48.29	10.84	13.19
	LoRA	41.30	321	100	66.60	64.13	43.58	44.70
	CA-LoRA	48.04	583	100	73.80	71.80	73.68	73.68
	EWC	29.12	1119	99.80	64.89	63.45	43.18	42.44
	LwF	28.46	2919	100	53.00	46.58	33.58	34.00
	O-LoRA	30.66	1945	100	65.60	57.29	76.20	76.13
	MiniLLM	29.26	572	100	70.00	70.54	72.84	72.86
	DistiLLM	25.56	1188	99.60	66.56	65.24	54.42	54.12
	SAGE (Ours) GT	**97.59**↑	**7.29**↓	**99.60**↑	**95.64**↑	**95.67**↑	**86.67**↑	**86.83**↑
	SAGE (Ours) GPT	82.95	6564	100	82.23	82.51	79.09	79.47

Once a knowledge region becomes stable, the Debugger determines how its teacher supervision should be injected into the student. A fixed adaptation configuration is unlikely to perform uniformly well across regions, as different capability deficiencies may require different adapter capacities and optimization dynamics. The Debugger therefore performs a two-stage search over low-rank adaptation configurations, combining broad exploration with local refinement.

At each interval Δ, the Debugger retrieves a stable cluster $\mathcal{C}_s$ and constructs distillation data from associated input–teacher-response pairs. Let Θ denote the search space over LoRA rank, learning rate, and other training hyperparameters. In the first stage, we sample N configurations from this space:

$$\mathcal{P}_n \sim \Theta, \qquad n = 1, \ldots, N. \tag{14}$$

Fine-tuning the student under configuration $\mathcal{P}_n$ produces an adapter A_n. Each adapter is evaluated using accuracy $Acc(A_n)$ and cross-entropy loss $CE(A_n)$ on the knowledge region. We rank candidates lexicographically, prioritizing higher accuracy and using lower cross-entropy as a tie-breaker:

$$\mathrm{TopK}^{\mathrm{lex}}_{n\in\{1,\ldots,N\}} \left(Acc(A_n), -CE(A_n)\right). \tag{15}$$

The resulting top-k configurations provide promising starting points for the refinement stage.

For each selected configuration θ_{k_i}, we refine the two hyperparameters that most directly control adaptation capacity and optimization: LoRA rank and learning rate. Let θ_{r,k_i} and θ_{l,k_i} denote the rank and learning rate of the i-th selected configuration. Its local neighborhood is

$$\begin{aligned}\mathcal{N}_{k_i} = &\{\theta_{r,k_i} - 2,\ \theta_{r,k_i},\ \theta_{r,k_i} + 2\} \\ &\times \{0.7\theta_{l,k_i},\ 0.8\theta_{l,k_i},\ 0.9\theta_{l,k_i}\},\end{aligned} \tag{16}$$

with invalid rank values excluded. This refinement produces up to $9k$ candidate configurations. We train the corresponding adapters and rank them using the same lexicographic criterion:

$$\mathrm{Top3}^{\mathrm{lex}}_{A \in \mathcal{A}_{\mathrm{refine}}} \left(Acc(A), -CE(A) \right), \tag{17}$$

where $\mathcal{A}_{\mathrm{refine}}$ denotes the adapters obtained from the local neighborhoods.

The selected adapters are associated with their corresponding knowledge regions and stored for subsequent reuse. When a future input is routed to a previously adapted region, the associated adapter is activated to improve the student's response without modifying its shared backbone. In this way, the Debugger converts persistent failure patterns into modular updates that can be selectively retrieved, reused, and extended as the input stream evolves.

4 Experiments

4.1 Experimental Settings

We conduct all experiments on a single NVIDIA A100 (80 GB) GPU using Qwen2.5-7B, LLaMA-2-7B, and Mistral-7B as base models. We use BGE-large-en-v1.5 for trigger detection and clustering, as decoder-only representations are less effective for sentence-level similarity. Because teacher signals from the OpenAI GPT-5 API are relatively unstable (85.71% OOD accuracy), we use ground-truth labels to simulate an ideal teacher and estimate the upper-bound performance of SAGE. We evaluate ID performance on TriviaQA Joshi et al. (2017) and OOD performance on GSM8K Cobbe et al. (2021), PubMedQA Jin et al. (2019), and LexGLUE Chalkidis et al. (2022), covering mathematical, biomedical, and legal domains, respectively. GSM8K is evaluated using Exact Match (EM), Mean Absolute Error (MAE), and Numeric Extraction Rate (NER), whereas PubMedQA and LexGLUE use accuracy (Acc) and F1; for the latter two, Chain-of-Thought decomposition is applied to produce shorter reasoning steps for SAGE (Appendix E). We compare SAGE with parameter-efficient adaptation methods (Base Model, LoRA Hu et al. (2022), O-LoRA Wang et al. (2023b), and CA-LoRA), continual-learning methods (EWC Kirkpatrick et al. (2017) and LwF Li & Hoiem (2017)), and distillation methods (MiniLLM Gu et al. (2024) and DistilLM Ko et al. (2024)). CA-LoRA uses statically clustered data, whereas all other baselines use the original datasets; all datasets are split into training, validation, and test sets at an 80%/10%/10% ratio.

Table 2: Efficiency analysis of SAGE for processing one sample.

Component	Latency (ms)	Throughput (samples/s)	Peak Memory (MiB)	Notes
Trigger	24.01	41	29330	real-time
Buffer	12.63	79	29475	real-time
Debugger	7726.05	0.13	40960	asynchronous
SAGE	493.11	2	–	–
Inference	9.62	103	–	–

4.2 Main Results

Comparison With Baslines. The results in Table 1 demonstrate that SAGE consistently achieves strong adaptation performance across different model families and OOD tasks. Under the ground truth setting (*GT*), SAGE outperforms all baselines across the three backbones. For example, on Qwen2.5-7B, SAGE improves GSM8K EM from 81.36% (O-LoRA) to 91.32% (+9.96), PubMedQA accuracy from 80.59% (CA-LoRA) to 89.12% (+8.53), and LexGLUE accuracy from 78.67% (LwF) to 92.00% (+13.33). The gains are even more pronounced on LLaMA2-7B and Mistral-7B, where SAGE reaches 99.80% and 97.59% EM on GSM8K, respectively, substantially exceeding the strongest competing baselines. Results on smaller backbone models show a similar trend, and evaluations with different teacher signals under the same setting are reported in Appendix D.1. With actual ChatGPT outputs, SAGE shows moderate degradation, achieving 86.83%, 88.65%, and 82.95% GSM8K EM on Qwen2.5-7B, LLaMA2-7B, and Mistral-7B, respectively, consistent with the gap between realistic and ideal teacher signals. Overall, SAGE can effectively identify, organize, and inject recurring knowledge gaps through its *Detect–Organize–Inject* pipeline.

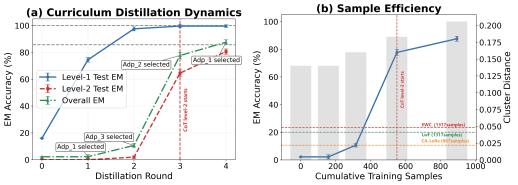

Figure 3: Curriculum learning and sample-efficient adaptation under SAGE. (a) EM accuracy across distillation rounds, highlighting the transition to Level-2 reasoning. (b) Overall accuracy versus cumulative training samples, with improving cluster structure quality over time.

Continual Learning Capability Analysis. To evaluate whether SAGE supports long-term continual learning, we conduct a four-round curriculum distillation experiment on GSM8K, introducing Level-1

data in early rounds and more complex Level-2 data (via CoT decomposition) later. As shown in Fig. 3(a), SAGE improves from 15.6% to ∼99% EM on Level-1 by Round 2 and maintains ≥99% thereafter, indicating strong retention without catastrophic forgetting. After introducing Level-2 data at Round 3, EM rises from near 0 to ∼65%, reaching ∼81% at Round 4, demonstrating capability expansion beyond the initial boundary. While Fig. 3(a) demonstrates what SAGE learns over time, Fig. 3(b) reveals how efficiently it does so. As SAGE selectively updates on failure cases, cluster separability increases, leading to more precise adaptation. Overall, results confirm SAGE enables both stable retention and continuous capability growth under long-term deployment.

Efficiency Analysis. Table 2 shows that SAGE separates lightweight online processing from expensive adaptation through a synchronous–asynchronous execution design. The Trigger and Buffer remain on the streaming path, requiring only 24.01,ms and 12.63,ms per sample and sustaining 41 and 79 samples/s, respectively. In contrast, the Debugger is substantially more expensive (7726.05,ms, 0.13 samples/s), but it is activated asynchronously at scheduled intervals to retrieve clusters already marked as stable and perform fine-tuning, rather than being executed for every incoming sample. Consequently, its cost does not directly block the online inference path, and the amortized cost of the complete SAGE pipeline remains 493.11,ms per sample. Once the adapters are produced, inference only requires matching the incoming sample to an existing knowledge region and loading the corresponding adapter, resulting in 9.62 ms latency and 103 samples/s throughput. These results indicate that SAGE can isolate costly knowledge injection from real-time serving while maintaining low inference overhead.

5 Ablation Study

5.1 Component Ablation

Table 3 reports component and data-matched ablations on GSM8K. Removing the Trigger causes a sharp EM drop from 99.80% to 8.89% under GT and from 84.95% to 6.72% with ChatGPT, while replacing the Buffer with HDBSCAN or K-means still underperforms full SAGE, confirming the value of boundary detection and structured knowledge organization. Using the same 315 selected samples, Debugger-based adaptation achieves 94.65%/82.68% EM under GT/ChatGPT, compared with 80.64%/68.80% for fixed LoRA, indicating that SAGE's gains do not arise solely from data curation.

Table 3: Ablation studies of SAGE on GSM8K. ↑ / ↓ indicate higher/lower is better. GT/GPT denote teacher signals using Ground-truth/ChatGPT.

	Method	Samples		GSM8K EM (%)↑	GSM8K MAE↓	GSM8K NER (%)↑
Component	**w/o Trigger** (random)	GT	499	8.89	11594	100
		GPT		6.72	14995	99.17
	w/o Trigger (All-to-Buffer)	GT	499	92.32	95	93.99
		GPT		81.27	473	63.13
	w/o Buffer	GT	499	92.32	95	93.99
		GPT		72.13	13917	57.52
	w/o Buffer (HDBSCAN)	GT	484	96.69	429	100
		GPT		78.70	21812	89.26
	w/o Buffer (K-means)	GT	329	97.55	14	99.09
		GPT		74.47	377	100
Data-Matched	**w/op Debugger**	GT	315	94.65	377	88.61
		GPT		82.68	578	99.40
	w/o Debugger (LoRA)	GT	315	80.64	14862	72.95
		GPT		68.80	13574	83.42
	SAGE (full)	GT	315	**99.80**↑	**0.01**↓	**100**↑
		GPT		**84.95**	**25846**	**99.40**

5.2 Sensitivity Analysis

Sensitive analysis on the Trigger. Separability Across ID/OOD Sources. To evaluate whether Trigger can identify boundary violations without labels, we analyze whether the anomaly score $\mathcal{AS}(x_t)$ separates ID and OOD samples. Fig. 4(a) shows a clear separation: semantic indicators drop from ID to OOD, while the logits margin shifts consistently, enabling boundary detection. Robustness to Threshold and Indicator Weights. To assess whether Trigger requires tuning, we analyze its sensitivity to the threshold τ and indicator weights. Fig. 4(b) shows stable performance across a wide threshold range, and Fig. 4(c) shows consistent results under different weight settings. These results suggest Trigger is robust to hyperparameter choices and generalizes well across settings.

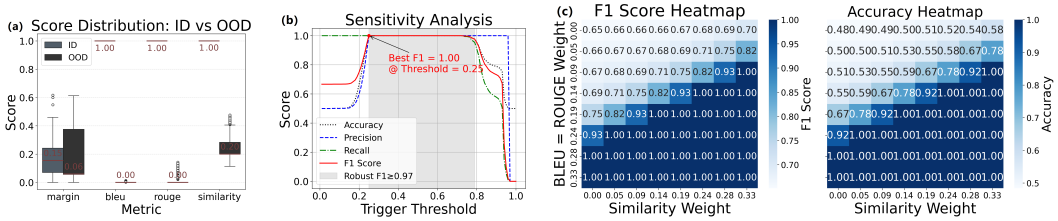

Figure 4: Parameter analysis of the Trigger Module. (a) False positive rates on ID vs. OOD samples. Red lines indicate the mean; logits margin metric is normalized. (b) Threshold sensitivity analysis. The gray area marks the optimal threshold range. (c) Heatmap of score weight sensitivity. The logits margin weight is computed dynamically from BLEU, ROUGE-L, and embedding similarity.

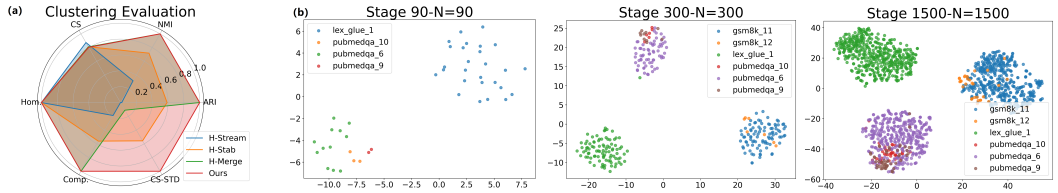

Figure 5: Results of the Buffer Module evaluation. (a) Evaluation of Streaming HDBSCAN Variants on Clustering Metrics: Homogeneity, ARI, NMI, Completeness, Central Similarity (CS), and Cluster Sample Std. (CS-STD). (b) Dynamic t-SNE Visualization as Data Volume (N) Increases.

Sensitive analysis on the Buffer Clustering Quality Across Streaming Variants. To evaluate whether the buffer produces reliable clusters under streaming conditions, we compare clustering strategies. Fig. 5(a) shows HDBSCAN alone is unstable, while adding stability tagging and fragment merging improves consistency. The full SBC achieves the best overall clustering quality, indicating all components are necessary. Convergence Under Streaming Data. To assess whether clustering improves over time, we analyze cluster evolution as more data are observed. Fig. 5(b) shows that clusters become more compact and better separated, indicating convergence to a stable structure.

Sensitive analysis on the Debugger. Rank and Learning Rate as Dominant Axes. To understand which hyperparameters govern effective adaptation and guide the debugger design, we analyze the impact of LoRA rank and learning rate. As shown in Fig. 6, performance varies substantially across configurations, with optimal regions concentrated at moderate values, while large learning rates degrade performance. The absence of a single optimal rank suggests cluster-dependent adaptation, motivating the two-stage search strategy. Robustness to Dropout and Target Modules. To assess whether other hyperparameters require tuning, we analyze dropout and target module selection. Fig. 7 shows minimal variation across configurations, indicating limited impact compared to rank and learning rate. This confirms that adaptation is primarily governed by these dominant axes, allowing the debugger to focus its search efficiently while fixing other parameters.

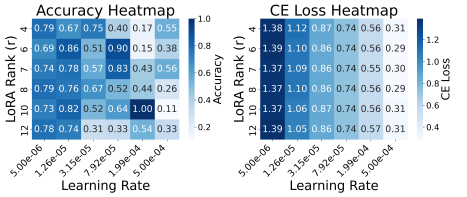

Figure 6: Heatmap of accuracy and cross-entropy loss for LoRA parameters: examining the necessity of tuning rank and learning rate.

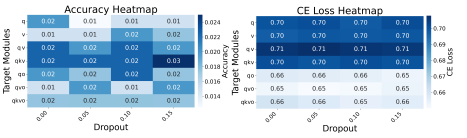

Figure 7: Heatmap of accuracy and cross-entropy loss for LoRA parameters: evaluating sensitivity to dropout and target modules.

6 Conclusion

We present SAGE, an online continual distillation framework that enables models to internalize teacher knowledge via a detect–organize–inject cycle. By integrating a label-free Trigger module, a streaming clustering Buffer module, and a low-rank adaptation Debugger module, SAGE operates without ground-truth labels and overcomes the limitations of offline distillation. Experiments show strong sample efficiency and consistent gains under distribution shift, validating continual learning. Future work will explore evolving knowledge dynamics in continual learning.

AI USE STATEMENT

In this work, we used assisting with translation, generative AI tools for editing a research paper to improve readability, proposing or refine hypotheses, supporting qualitative and thematic data analysis and proposing a title or keywords for a research paper. We have not used generative AI tools for generate synthetic data sets, helping develop theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, designing or providing feedback on research methodology or experiments, implementing methods, cleaning and reformating dataset, interpret results and assisting in the writing of proofs are not applicable to this work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

Vincent-Pierre Berges, Barlas Oğuz, Daniel Haziza, Wen-tau Yih, Luke Zettlemoyer, and Gargi Gosh. Memory layers at scale. *arXiv e-prints*, pp. arXiv–2412, 2024. URL https://openreview.net/forum?id=ATqGm1WyDj.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. Improving language models by retrieving from trillions of tokens. In *International conference on machine learning*, pp. 2206–2240. PMLR, 2022. URL https://proceedings.mlr.press/v162/borgeaud22a.html.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Ricardo JGB Campello, Davoud Moulavi, and Jörg Sander. Density-based clustering based on hierarchical density estimates. In *Pacific-Asia conference on knowledge discovery and data mining*, pp. 160–172. Springer, 2013. URL https://link.springer.com/chapter/10.1007/978-3-642-37456-2_14.

Ilias Chalkidis, Abhik Jana, Dirk Hartung, Michael Bommarito, Ion Androutsopoulos, Daniel Katz, and Nikolaos Aletras. Lexglue: A benchmark dataset for legal language understanding in english. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4310–4330, 2022.

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.

Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3366–3385, 2021.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale. *Advances in neural information processing systems*, 35: 30318–30332, 2022.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *Advances in neural information processing systems*, 36:10088–10115, 2023.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2021. URL https://arxiv.org/abs/2012.14913.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Minillm: Knowledge distillation of large language models. In *The twelfth international conference on learning representations*, 2024.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Audrey Huang, Adam Block, Dylan J Foster, Dhruv Rohatgi, Cyril Zhang, Max Simchowitz, Jordan T Ash, and Akshay Krishnamurthy. Self-improvement in language models: The sharpening mechanism. In *The Thirteenth International Conference on Learning Representations*. URL https://openreview.net/forum?id=WJaUkwci9o.

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua Lu. Pubmedqa: A dataset for biomedical research question answering. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pp. 2567–2577, 2019.

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.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114 (13):3521–3526, 2017.

Jongwoo Ko, Sungnyun Kim, Tianyi Chen, and Se-Young Yun. Distillm: Towards streamlined distillation for large language models. In *International Conference on Machine Learning*, pp. 24872–24895. PMLR, 2024.

Jongwoo Ko, Tianyi Chen, Sungnyun Kim, Tianyu Ding, Luming Liang, Ilya Zharkov, and Se-Young Yun. Distillm-2: A contrastive approach boosts the distillation of llms. In *International Conference on Machine Learning*, pp. 31044–31062. PMLR, 2025.

Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, et al. Wilds: A benchmark of in-the-wild distribution shifts. In *International conference on machine learning*, pp. 5637–5664. PMLR, 2021.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html.

Dongfang Li, Zetian Sun, Xinshuo Hu, Baotian Hu, and Min Zhang. Cmt: A memory compression method for continual knowledge learning of large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 24413–24421, 2025. URL https://ojs.aaai.org/index.php/AAAI/article/view/34619.

Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pp. 74–81, 2004. URL https://aclanthology.org/W04-1013.pdf.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=BJC_jUqxe.

Jingyu Liu, Jiaen Lin, and Yong Liu. How much can rag help the reasoning of llm? *arXiv preprint arXiv:2410.02338*, 2024. URL https://openreview.net/forum?id=Q6M7bZIo9t.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pp. 311–318, 2002. URL https://dl.acm.org/doi/abs/10.3115/1073083.1073135.

German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural networks*, 113:54–71, 2019.

Archiki Prasad, Weizhe Yuan, Richard Yuanzhe Pang, Jing Xu, Maryam Fazel-Zarandi, Mohit Bansal, Sainbayar Sukhbaatar, Jason E Weston, and Jane Yu. Self-consistency preference optimization. In *Forty-second International Conference on Machine Learning*. URL https://openreview.net/forum?id=94G4eL3RWi.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. *Advances in neural information processing systems*, 32, 2019.

V Sanh. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. In *Proceedings of Thirty-third Conference on Neural Information Processing Systems (NIPS2019)*, 2019.

Sheikh Shafayat, Fahim Tajwar, Ruslan Salakhutdinov, Jeff Schneider, and Andrea Zanette. Can large reasoning models self-train? *arXiv preprint arXiv:2505.21444*, 2025. URL https://arxiv.org/abs/2505.21444.

Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. Training very deep networks. *Advances in neural information processing systems*, 28, 2015.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Weizhi Wang, Li Dong, Hao Cheng, Xiaodong Liu, Xifeng Yan, Jianfeng Gao, and Furu Wei. Augmenting language models with long-term memory. *Advances in Neural Information Processing Systems*, 36:74530–74543, 2023a. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/ebd82705f44793b6f9ade5a669d0f0bf-Abstract-Conference.html.

Xiao Wang, Tianze Chen, Qiming Ge, Han Xia, Rong Bao, Rui Zheng, Qi Zhang, Tao Gui, and Xuan-Jing Huang. Orthogonal subspace learning for language model continual learning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 10658–10671, 2023b.

Yu Wang, Yifan Gao, Xiusi Chen, Haoming Jiang, Shiyang Li, Jingfeng Yang, Qingyu Yin, Zheng Li, Xian Li, and Bing Yin. Memoryllm: Towards self-updatable large language models. *ICML*, 2024. URL https://dl.acm.org/doi/abs/10.5555/3692070.3694135.

Hongkang Yang, Zehao Lin, Wenjin Wang, Hao Wu, Zhiyu Li, Bo Tang, Wenqiang Wei, Jinbo Wang, Zeyun Tang, Shichao Song, et al. Memory3: Language modeling with explicit memory. *arXiv preprint arXiv:2407.01178*, 2024. URL https://openreview.net/forum?id=FZZc7qgE7p.

A Implementation Details

We will release the source code, training scripts, and evaluation pipelines under the MIT License for research purposes. All datasets and pretrained models are used in compliance with their original licenses, solely for academic research. Generated artifacts are likewise intended for non-commercial research use. The details are as follows.

Trigger The anomaly trigger score is computed as a weighted combination of four signals: logit margin, BLEU-1, ROUGE-L, and embedding cosine similarity, each assigned equal weight of 0.25. The trigger threshold is initialized to 0.5 and dynamically recalibrated using 50 warm-up ID samples, with the valid similarity range set to the 5th–95th percentile of in-distribution scores.

Buffer Anomaly samples are routed into domain buckets by source tag. Clustering is triggered once a bucket accumulates 15 samples, using HDBSCAN with `min_cluster_size`=3. Cluster stability is validated by two criteria before any cluster is committed: the Adjusted Rand Index (ARI) between two consecutive clustering passes must exceed 0.8, and the mean cosine similarity between corresponding cluster centers must exceed 0.9. Clusters whose centers share cosine similarity above 0.8 are subsequently merged. When similarity threshold is set to `auto`, it is estimated as the 30th percentile of up to 1,000 randomly sampled pairwise similarities from existing embeddings.

Debugger For each stable cluster, LoRA adapters are selected via a two-stage search. First, 6 configurations are sampled from the search space: rank $r \in \{4, 8, 16, 32\}$, learning rate $\in \{10^{-4}, 10^{-5}, 10^{-6}, \}$, dropout 0.05, target modules`q_proj`/`k_proj`/`v_proj`/`o_proj`, batch size 1, 5, 10 training epochs, with α fixed at $2r$. The final adapter is selected by validation accuracy (primary) and validation loss (secondary). All training inputs are formatted as `Question: {prompt} / Context: {completion} / Answer: {label}`, truncated to 512 tokens, and optimized with AdamW. A 20% validation split is held out per cluster for adapter selection.

B Algorithmic Framework

B.1 Details for Buffer Module of SBC Algorithm

The Buffer module of Streaming Buffer Clustering (SBC) algorithm details are as follows:

For each newly arrived abnormal sample x, the SBC algorithm infers its structure tag and extracts its semantic embedding vector and keyword set for subsequent similarity matching. If the cluster for the structure tag s is deemed "*stable*" (i.e., a structure cluster $\mathcal{C}_s$ already exists), SBC algorithm computes the similarity between the current sample and all candidate clusters under the same tag, selecting the one with the highest score (Line 5). The clustering score γ is a weighted combination of semantic embedding similarity and keyword overlap. If γ exceeds the threshold τ, the sample x is added to the selected cluster $\mathcal{C}^*$ (Line 8), and the assignment result is returned, terminating the process. If the structure cluster is not stable or the score does not meet the threshold, the sample is temporarily stored in the buffer $\mathcal{B}_s$ for tag s (Line 12), awaiting further processing by the delayed clustering mechanism.

When a newly abnormal sample x causes the number of samples in the buffer associated with structure tag s to reach the predefined threshold T (Line 14), the SBC algorithm triggers batch clustering. The system extracts the semantic embedding vectors $\mathbf{e}_i$ of all buffered samples (Line 15), then applies HDBSCANfor unsupervised density-based clustering. However, HDBSCAN alone cannot guarantee clustering *stability*. To address this, we introduce two stability check metrics: ① the Adjusted Rand Index (ARI), measuring label consistency across successive clusterings; and ② the Average Cosine Similarity, quantifying semantic coherence between new and previous cluster centroids. When both exceed their thresholds η_{ARI} and $\eta_{\cos}$, clustering is considered stable. The buffered samples are migrated to a newly formed formal cluster $\mathcal{C}_{\text{new}}$, the buffer $\mathcal{B}_s$ cleared (Line 21), and the structure tag s marked as "*stable*" (Line 22).

Given that streaming input can fragment clusters, when the number of stable clusters for tag s exceeds the threshold, an inter-cluster merge (Line 24) is triggered. Clusters are merged if their centroids' cosine similarity exceeds threshold δ, reducing redundancy and increasing sample density. If batch clustering fails to meet stability criteria, the result is discarded and samples remain buffered for future clustering(Line 28). Finally, samples unassigned to any cluster and without triggered clustering are marked as "*unassigned*" until clustering conditions are met (Line 31).

B.2 Details for Debugger Module of CLO Algorithm

The Debugger module of Cluster-Aware LoRA Optimization (CLO) algorithm are as follows:

Algorithm 1: Streaming Buffer Clustering (SBC)

Input: Incoming anomaly sample x, data buffer $\mathcal{D}$, current clusters $\mathcal{C}$, structure tag s
Output: Cluster assignment or data buffer update

$s \leftarrow \text{InferStructure}(x)$
$e \leftarrow \text{GetEmbedding}(x)$
$k \leftarrow \text{ExtractKeywords}(x)$
if IsClusterStable(s) == True **then**
    $(\mathcal{C}^*, \gamma) \leftarrow \text{FindBestCluster}(s, e, k)$
γ: weighted score combining embedding similarity e and keyword overlap k
    if $\gamma \geq \tau$ **then** ▷ Exceeding threshold τ
        Assign x to cluster $\mathcal{C}^*$
        return cluster assignment
    end if
else
    $\text{AddToBuffer}(x, e, k, \mathcal{D}_s)$
end if
if $|\mathcal{D}_s| \geq T$ **then** ▷ Exceeding clustering threshold T
    $e \leftarrow \text{ExtractEmbeddings}(\mathcal{D}_s)$
    $h \leftarrow \text{HDBSCAN}(e)$
▷ Density clustering without preset count
    $(\text{ARI}, \text{Sim}) \leftarrow \text{EvalStability}(h, h^{(t-1)}, e, e^{(t-1)})$
    if $\text{ARI} \geq \theta_{\text{ari}} \vee \text{Sim} \geq \theta_{\text{sim}}$ **then**
        $\text{CreateNewClusters}(h, \mathcal{D}_s)$
        $\mathcal{D}_s \leftarrow \emptyset$ ▷ Clear buffer
        $\text{StableFlag}_s \leftarrow \text{True}$
        if $|\mathcal{C}_s| \geq 3$ **then**
            $\text{MergeSimilarClusters}(\mathcal{C}_s)$
▷ Merge clusters with centroid similarity
        end if
    else
        return ▷ Clustering unstable, wait more
    end if
end if
return x as "Unassigned" ▷ No matching cluster

Given a cluster $\mathcal{C}$ with stable structure, the CLO algorithm randomly samples candidate configurations n from the hyperparameter search space Θ to form the initial configuration set Θ_0. Simultaneously, the system creates an empty list to store the results of each configuration after training, including validation accuracy, cross-entropy loss, save path, and corresponding LoRA adapter configuration information. The CLO algorithm then performs complete LoRA fine-tuning on each configuration in $\mathcal{P}_n$ for the samples in the cluster $\mathcal{C}_s$ (Line 4), recording the training results to form a preliminary result set $\mathcal{R}_0$ (Line 5). Next, CLO algorithm ranks the results in $\mathcal{R}_0$ based on a sorting strategy that prioritizes validation accuracy firstly and cross-entropy loss secondly, selecting the Top-k configurations (Line 9). This strategy accounts for the limited data size, prioritizing accuracy to ensure SAGE's correct response to similar inputs.

Based on these k optimal configurations, CLO algorithm conducts exploitation optimization in their neighboring parameter spaces (Line 12), mainly adjusting the LoRA rank and learning rate. For each configuration in the fine-tuned set $\mathcal{P}_{\text{opt}}$, CLO algorithm continues LoRA fine-tuning, resulting in a refined result set $\mathcal{R}_1$ (Line 14). Finally, CLO algorithm merges the preliminary result set $\mathcal{R}_0$ with the refined set $\mathcal{R}_1$ into the total result set $\mathcal{R}$all (Line 19), sorts it again by accuracy and cross-entropy loss, and selects the final Top-3 configurations (Line 21). The corresponding LoRA adapters are saved as the final optimized results $\mathcal{A}_{\text{LoRA}}$ (Line 22), and the three optimal adapters, along with the best configuration θ^*, are returned for subsequent reasoning calls (Line 23). For new data with different characteristics, adaptation is achieved by triggering additional LoRA fine-tuning.

The CLO algorithm combining initial exploration with local refinement, making it suitable for atomic tasks characterized by sparse data and diverse atomic tasks. It ensures that each cluster has the

Algorithm 2: Cluster-Aware LoRA Optimization (CLO)

Input: Incoming stable cluster $\mathcal{C}_s$, base model $\mathcal{M}$, tokenizer $\mathcal{T}$, parameter space Θ , top-k configs k
Output: Optimized LoRA config θ^* and LoRA adapter $\mathcal{A}$

$\mathcal{P}_n \leftarrow \text{InitialConfigs}(\Theta, n)$ ▷ Initialize n seed configs
$\mathcal{R}_0 \leftarrow []$ ▷ Initial training results
for all $\theta \in \mathcal{P}_n$ **do**
    $r \leftarrow \text{TrainLoRA}(\theta, \mathcal{C}, \mathcal{M}, \mathcal{T})$
    $\mathcal{R}_0 \leftarrow \mathcal{R}_0 \cup \{r\}$
end for
$\mathcal{R}_0 \leftarrow \text{SortByScore}(\mathcal{R}_0)$
maximize accuracy, break ties by minimizing loss
$\mathcal{P}_{\text{top}} \leftarrow \text{Top-KConfigs}(\mathcal{R}_0, k)$
$\mathcal{R}_1 \leftarrow []$ ▷ Refined training results
for all $\theta_{\text{top}} \in \mathcal{P}_{\text{top}}$ **do**
    $\mathcal{P}_{\text{opt}} \leftarrow \text{LocalParamSearch}(\theta_{\text{top}})$
#local search over rank and learning rate near θ_{top}
    for all $\theta' \in \mathcal{P}_{\text{opt}}$ **do**
        $r' \leftarrow \text{TrainLoRA}(\theta', \mathcal{C}, \mathcal{M}, \mathcal{T})$
        $\mathcal{R}_1 \leftarrow \mathcal{R}_1 \cup \{r'\}$
    end for
end for
$\mathcal{R}_{\text{all}} \leftarrow \mathcal{R}_0 \cup \mathcal{R}_1$
$\mathcal{R}_{\text{all}} \leftarrow \text{SortByScore}(\mathcal{R}_{\text{all}})$
$\mathcal{A}_{\text{LoRA}} \leftarrow \text{Top-KResults}(\mathcal{R}_{\text{all}}, 3)$
$\text{SaveBuffer}(\mathcal{A}_{\text{LoRA}})$
return $\mathcal{A}_{\text{LoRA}}$ and θ^*

optimal LoRA configuration for the current data, providing a solid foundation for subsequent adapter upgrades and dynamic fine-tuning.

C Discussion

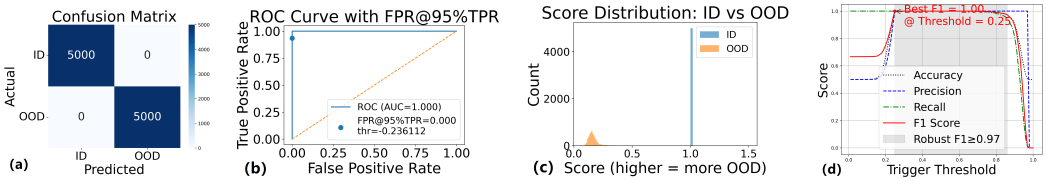

Figure 8: Trigger module evaluation on long-form OOD datasets with TriviaQA as ID. (a) Confusion matrix. (b) ROC curve. (c) Anomaly score distributions. (d) Threshold sensitivity.

Applicability and Selective Adaptation. SAGE excels when tasks are decomposable into atomic units with consistent low-rank gradient directions. For complex tasks, Chain-of-Thought (CoT) decomposition can restore this property. When atomicity is absent, SAGE's selective mechanism—comprising a competence-based trigger and a stability-oriented buffer—deliberately withholds updates. This prevents noisy gradients from contaminating the student model, ensuring robust streaming distillation.

Generalizability and Vision Extension. The trigger module demonstrates strong zero-shot transferability. Evaluations on OOD datasets (EntailmentBank, WorldTree, ARC) using TriviaQA-calibrated parameters show near-perfect ID/OOD separation (AUC $\approx$ 1.0, Figure 8). This confirms that SAGE captures semantic teacher-student discrepancies rather than surface lexical patterns. Such robustness suggests SAGE is natively extensible to computer vision. The detect-organize-inject cycle can be adapted by replacing text-based indicators with visual feature similarity or confidence divergence. Given that streaming clustering and LoRA are already established in vision tasks, extending SAGE primarily requires redefining trigger indicators for visual inputs.

Limitations and Future Directions. SAGE's primary constraint is its dependence on task atomicity; highly entangled representations may hinder cluster convergence. Furthermore, while sparse triggering amortizes the LoRA update latency to 493ms per sample, the 7726ms per-cluster overhead remains a bottleneck. Future work will explore asynchronous execution to hide latency, higher-dimensional memory fields with learned routing, and task-aware controllers to further enhance trigger precision.

D Experimental Details

D.1 Details for the SAGE

Table 4: Overall Performance Comparison across GSM8K, PubMedQA and LexGlue. ↑ indicates higher is better, ↓ indicates lower is better.

Model	Method		GSM8K (%)			PubMedQA (%)		LexGlue (%)	
			EM↑	MAE↓	NER↑	Acc↑	F1↑	Acc↑	F1↑
Qwen2.5-1.5B	Base Model		4.07	3194	81.77	23.80	17.46	34.68	13.68
	LoRA		23.36	1491	100	50.50	34.61	50.64	47.78
	CA-LoRA		32.71	951	100	69.48	69.48	66.12	67.78
	EWC		2.33	4894	20.54	56.16	56.81	69.80	71.37
	LwF		19.44	1615	89.40	49.50	50.04	73.00	72.92
	O-LoRA		36.46	1182	100	51.14	50.33	69.60	69.42
	MiniLLM		22.79	3351	100	50.40	49.40	70.26	70.13
	DistiLLM		31.72	1161	100	50.30	34.61	58.12	57.64
	SAGE (ours)	GT	**47.70** ↑	**1226**↓	**100**↑	**85.64**↑	**85.67**↑	**82.46**↑	**81.73**↑
		ChatGPT	36.47	2019	100	73.91	74.62	74.71	73.45
Qwen2.5-3B	Base Model		2.73	3965	88.18	57.72	57.59	25.14	25.37
	LoRA		35.48	1842	100	63.01	60.84	51.50	51.28
	CA-LoRA		44.78	4112	100	64.62	62.62	62.17	62.17
	EWC		2.21	11119	9.80	59.79	58.83	73.80	73.69
	LwF		21.04	1404	100	61.37	59.64	75.80	75.75
	O-LoRA		55.11	1431	100	64.60	65.29	62.60	62.42
	MiniLLM		46.08	1167	100	62.78	61.54	69.57	69.58
	DistiLLM		51.30	4126	100	67.32	66.91	58.30	51.88
	SAGE (ours)	GT	**58.23** ↑	**1515**↓	**100**↑	**95.64**↑	**95.67**↑	**83.95**↑	**84.17**↑
		ChatGPT	51.52	15498	90.10	76.92	75.71	75.64	76.74

Analysis of Smaller Backbone Models for SAGE Table 4 shows that SAGE remains effective on smaller backbone models, indicating that its adaptation capability is not limited to larger models. Under the ground-truth setting, SAGE improves GSM8K EM from 36.46% to 47.70% on Qwen2.5-1.5B and from 55.11% to 58.23% on Qwen2.5-3B, while also achieving clear gains on PubMedQA and LexGLUE. In particular, SAGE reaches 85.64% and 95.64% accuracy on PubMedQA for the 1.5B and 3B backbones, respectively, substantially outperforming the strongest baselines. With actual ChatGPT outputs, performance moderately decreases to 36.47% and 51.52% GSM8K EM, respectively, consistent with the expected gap between realistic and ideal teacher signals. Overall, these results suggest that SAGE provides stable adaptation benefits across different model scales.

Table 5: Effect of Different Teacher Signals on SAGE Performance with LLaMA2-7B. ↑ indicates higher is better, ↓ indicates lower is better.

Teacher Signal	Teacher Accuracy (%)	GSM8K			PubMedQA (%)		LexGLUE (%)	
		EM (%)↑	MAE↓	NER (%)↑	Acc↑	F1↑	Acc↑	F1↑
Ground Truth	100.00	99.80	0.01	100	95.70	95.37	87.14	87.63
GPT-5	85.71	88.65	425	100	84.73	83.67	77.22	77.87
Claude	84.04	86.43	579	99.20	76.85	75.17	78.70	77.25
Qwen	81.42	82.05	874	94.60	68.06	70.10	70.77	70.29
DeepSeek	80.95	73.06	7028	89.76	68.94	68.79	67.43	68.34

Analysis of Different Teacher Models for SAGE Table 5 shows that SAGE's adaptation performance broadly tracks the quality of the teacher signal. On GSM8K, EM decreases from 99.80% with

ground-truth supervision to 88.65%, 86.43%, 82.05%, and 73.06% when using GPT-5, Claude, Qwen, and DeepSeek, respectively. A similar overall pattern is observed on PubMedQA, where accuracy decreases from 95.70% under ground-truth supervision to 84.73%, 76.85%, 68.06%, and 68.94% with the four model-based teachers. On LexGLUE, ground-truth supervision achieves 87.14% accuracy, compared with 77.22%, 78.70%, 70.77%, and 67.43% for GPT-5, Claude, Qwen, and DeepSeek, respectively. Although the relationship is not strictly monotonic across every dataset, higher-quality teacher signals generally lead to stronger downstream adaptation, indicating that teacher reliability is an important factor in SAGE's continual knowledge transfer. Since SAGE requires only output-level reference answers rather than access to teacher parameters or intermediate representations, ground-truth answers provide an idealized teacher signal for estimating the upper-bound performance of the framework while isolating the effect of noise introduced by imperfect teacher predictions.

D.1.1 Cross-Round Retention Analysis

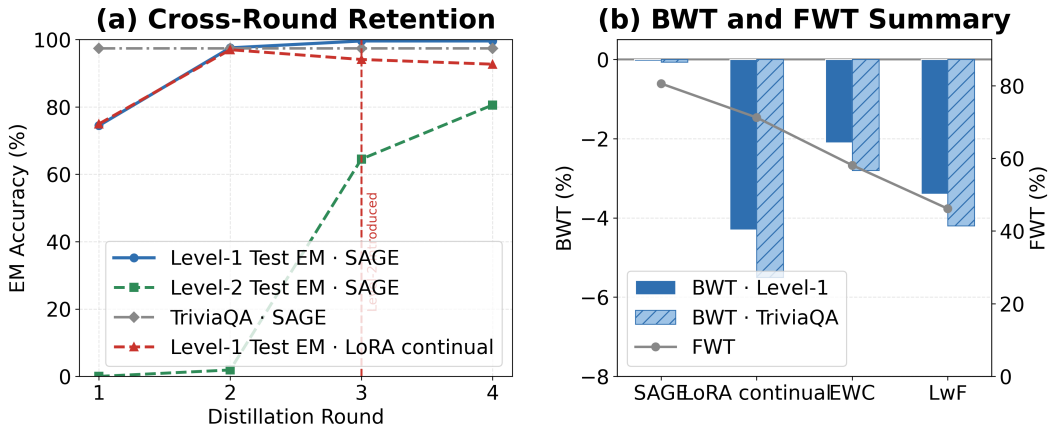

Figure 9: Cross-round knowledge retention and transfer under SAGE. (a) EM accuracy across rounds, showing stable retention and capability expansion.(b) BWT and FWT across methods, where SAGE achieves near-zero forgetting with the highest forward transfer.

To evaluate whether SAGE's external adapter design preserves previously acquired knowledge under continuous distillation, we measure cross-round retention on three held-out test sets: GSM8K Level-1 (learned in Rounds 1–2), GSM8K Level-2 (introduced in Rounds 3–4), and TriviaQA (ID benchmark, never used for adaptation). Retention is quantified via two standardized continual learning metrics: Backward Transfer ($BWT = \frac{1}{T-1}\sum_{i=1}^{T-1}(R_{T,i} - R_{i,i})$), measuring average performance drop on previously learned tasks after subsequent adaptation rounds, and Forward Transfer ($FWT = \frac{1}{T-1}\sum_{i=2}^{T}(R_{i-1,i} - R_{0,i})$), measuring average performance gain on new tasks relative to the zero-shot baseline. We compare SAGE against Standard LoRA continual (sequential fine-tuning directly on base model parameters without adapter isolation), EWC, and LwF under identical data conditions across all four rounds.

Figure 9(a) reports EM accuracy of Level-1, Level-2, and TriviaQA across all four rounds. Level-1 EM remains stable at over 99% throughout Rounds 1–4, demonstrating that the introduction of Level-2 knowledge at Round 3 causes no degradation of previously acquired reasoning capabilities. TriviaQA accuracy similarly remains stable across all rounds, confirming that sequential low-rank adaptation does not impair the student model's general language understanding. This stability arises directly from SAGE's parametric isolation design: since base model parameters are never modified and each adapter is stored independently in the Debugger, new adaptation cycles cannot overwrite previously consolidated knowledge regions. In contrast, Standard LoRA continual exhibits a measurable Level-1 EM drop after Level-2 data is introduced at Round 3, confirming that parameter sharing is the root cause of forgetting rather than the distillation process itself.

Figure 9(b) quantifies this comparison via BWT and FWT. SAGE achieves a BWT of 0.0% on both Level-1 and TriviaQA, confirming near-perfect knowledge retention without any explicit anti-forgetting mechanism. Meanwhile, SAGE achieves the highest FWT among all compared methods, indicating that parametric isolation does not compromise the system's ability to absorb new knowledge.

In contrast, EWC and LwF exhibit non-trivial BWT despite being explicitly designed for continual learning, while their FWT values decline monotonically, revealing the stability–plasticity tradeoff inherent in regularization-based and distillation-based retention strategies. These results collectively confirm that SAGE's each distillation cycle expands the student's competence boundary without eroding previously internalized knowledge regions.

D.2 Details for Trigger Module

In further experiments, we fixed the threshold at 0.5 and set equal weights for all four indicators. We then evaluated the Trigger module independently under these test-time parameters. As shown in Figure 10, the module achieved 100% ID/OOD separation under this configuration, confirming that the chosen parameters fall within a favorable operational range. The OOD and ID score distributions in Figure 10(c) show that the OOD and ID samples fall within two distinct ranges, providing a sufficiently wide range of thresholds to distinguish the two classes. Together with the sensitivity analysis in Figure 4, these findings suggest that the Trigger module possesses not only high accuracy but also strong robustness and generalizability, making it well-suited to serve as a reliable activation mechanism for subsequent adaptive fine-tuning in LLM-based systems.

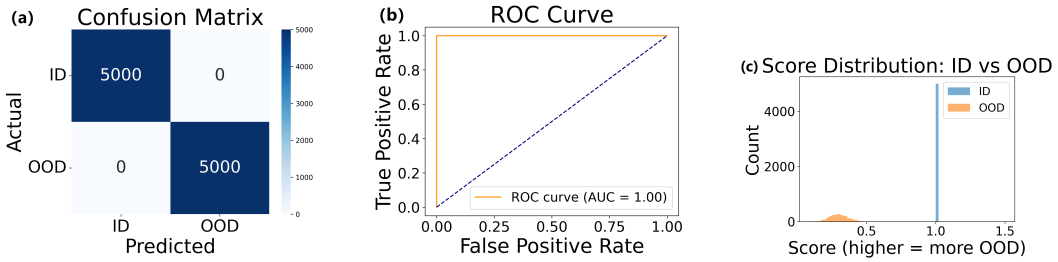

Figure 10: Results of the Trigger Module evaluation under formal test parameters. (a) Confusion matrix for ID and OOD samples. (b) ROC curve illustrating detection performance. (c)Anomaly score of ID datasets and OOD datasets.

D.3 Details for Buffer Module

Figure 11 further indicates that inter-cluster distances remain stable while the total cluster count converges, suggesting that the buffer does not fragment indefinitely but stabilizes into a fixed set of distinct knowledge regions. This convergence ensures that teacher supervision is progressively consolidated into stable, non-overlapping targets, enabling increasingly precise low-rank adaptation over time.

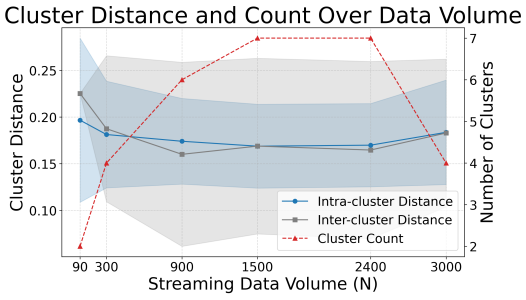

Figure 11: Cluster distance and cluster count over streaming data volume.

To cope with streaming data, we designed multiple thresholds. Below is a sensitivity analysis of each threshold to verify the rationality of our SBC algorithm design.

Figure 12 shows the results of the similarity threshold in the threshold gating module, which comes after the match clusters module in the routing and matching process. Data values greater than this threshold are considered similar to a cluster and can be added to it. Since a lower DBI and a higher

CH are desirable, and moderate values of unassigned rate and cluster number are preferred, the similarity threshold should be in the range of 0.55 to 0.65. This threshold is relatively sensitive and can serve as the primary factor for adjusting the clustering process.

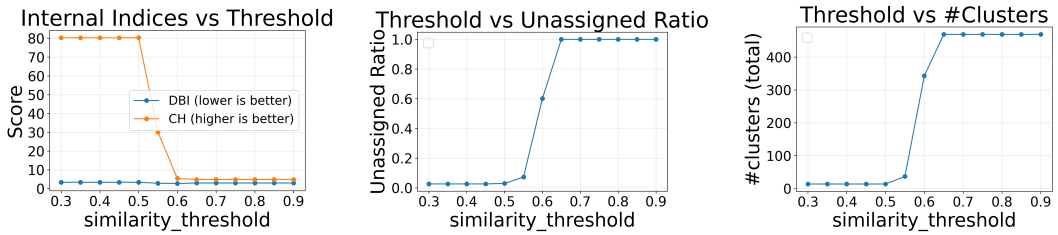

Figure 12: Similarity threshold sensitivity analysis for threshold gating after match clusters module. (left) Davies–Bouldin Index and Calinski–Harabasz Index for internal indices. (middle) Outlier Ratio for proper threshold. (right) Cluster Stability.

Figure 13 shows the results of the data buffer size threshold for the threshold gating module, which precedes the density clustering module in the cluster-and-tag process. When the threshold exceeds this value, the SBC algorithm triggers the density clustering module, and the HDBSCAN algorithm performs the clustering. The results show that a data buffer size between 5 and 40 yields the best performance. Adjusting this parameter can slightly improve clustering results, but the effect is not significant, indicating that it is suitable for secondary parameter tuning.

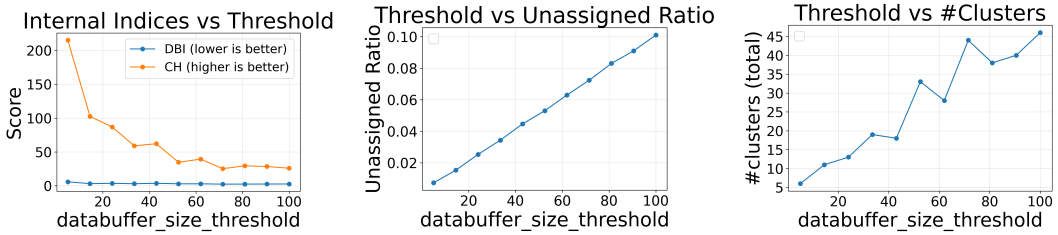

Figure 13: Data buffer size threshold sensitivity analysis for threshold gating before density cluster module. (left) Davies–Bouldin Index and Calinski–Harabasz Index for internal indices. (middle) Outlier Ratio for proper threshold. (right) Cluster Stability.

Figure 14 shows the results of the merge threshold for the threshold gating module, which precedes the merge clusters module in the merge fragments module. When the threshold exceeds this value, the SBC algorithm enters the fusion region to determine cluster centers. The results show that a threshold value greater than 6 is most effective, and the clustering performance remains largely unchanged. Therefore, this parameter can be used to enable or disable the merge operation, and adjustment is unnecessary unless the merge operation needs to be disabled.

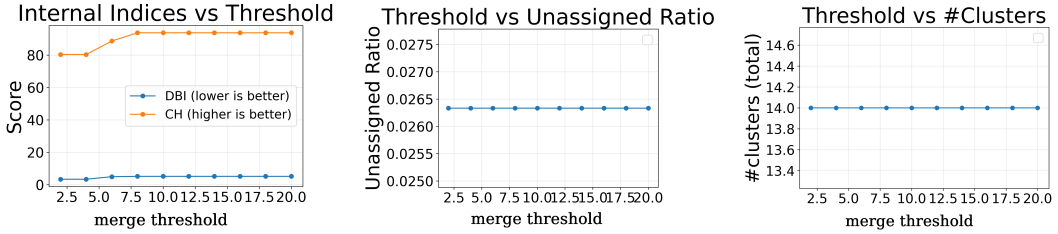

Figure 14: Merge threshold sensitivity analysis for threshold gating before merge clusters module.(left) Davies–Bouldin Index and Calinski–Harabasz Index for internal indices. (middle) Outlier Ratio for proper threshold. (right) Cluster Stability.

Figure 15 shows the results of the cluster center similarity threshold in the merge clusters module, which is part of the merge fragments process. When the threshold exceeds this value, the SBC algorithm executes the fragmented cluster fusion operation. The results show that setting the threshold between 0.8 and 1.0 yields the best performance, with clustering remaining largely stable across this

range. Therefore, this parameter can be set as a fixed value within the algorithm and does not require further adjustment.

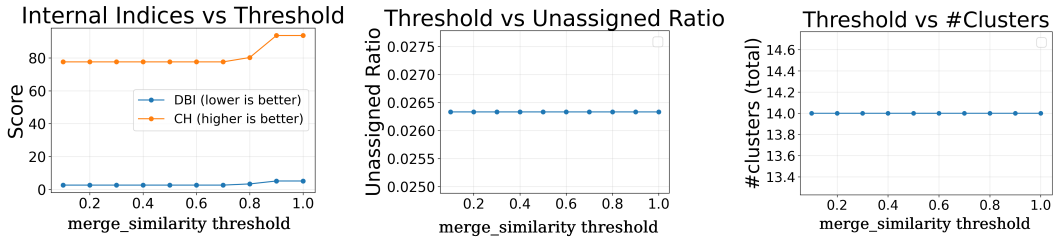

Figure 15: Cluster center similarity threshold sensitivity analysis for merge clusters module.(left) Davies–Bouldin Index and Calinski–Harabasz Index for internal indices. (middle) Outlier Ratio for proper threshold. (right) Cluster Stability.

Figure 16 shows the results of a sensitivity analysis of the Adjusted Random Index and Average Cosine Similarity thresholds. These thresholds are used in the density module of the cluster and tag process. The SBC algorithm requires two thresholds, and both must simultaneously exceed their respective values for the algorithm to assign a stability label to a cluster. This requirement also ensures that the cluster can be matched with the data. Therefore, we conducted a detailed threshold analysis of these two parameters in a three-dimensional space. The results show that an Adjusted Random Index (ARI) threshold above 0.8 performs well. An Average Cosine Similarity (ACS) threshold between 0.5 and 0.9 also performs well. Both thresholds are very robust. Therefore, no adjustment of these threshold parameters is recommended.

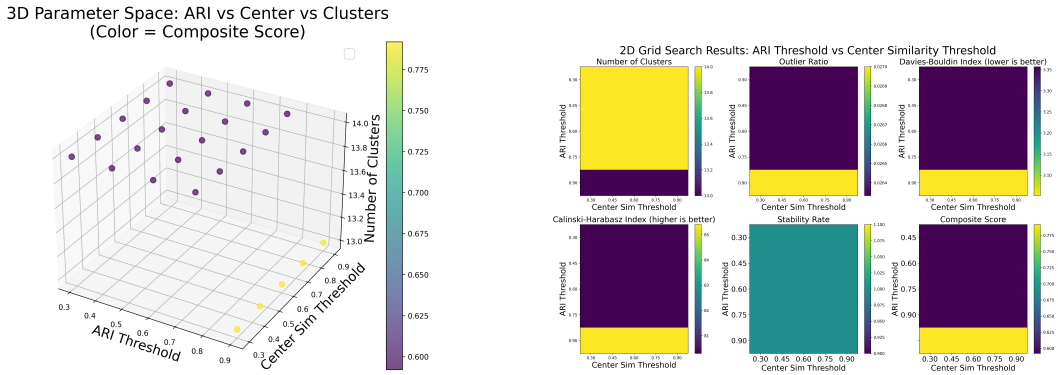

Figure 16: Ajusted Random Index and average Cosine Similarity thresholds sensitivity analysis for density cluster module. (left) 3d parameter for cluster stability of threshold. (right) 2d parameter results for cluster stability, outlier ratio, DBI, CH, stability rate and composite score.

Figure 15 shows the results of the match score weight in the match cluster module for routing and matching. This threshold determines the score weight parameter. Because there are only two weights that sum to 1, only one weight parameter was tested. The weight parameter shown in the figure is the coefficient of embedding similarity between the data and the cluster. The results show that setting this threshold between 0.5 and 0.7 performs best and clustering results are sensitive to changes in this weight. Therefore, this threshold should be prioritized as the primary factor for clustering adjustments.

D.4 Details for Debugger Module

To examine stability, we shuffle the dataset with different random seeds. As shown in Table 6, SAGE maintains high performance with EM $= 97.16_{\pm 4.65}$, indicating weak sensitivity to input order. While Adp_1 shows higher variance, Adp_2 and Adp_3 consistently converge across seeds, suggesting that buffer-based clustering and delayed adaptation help stabilize learning under different data permutations.

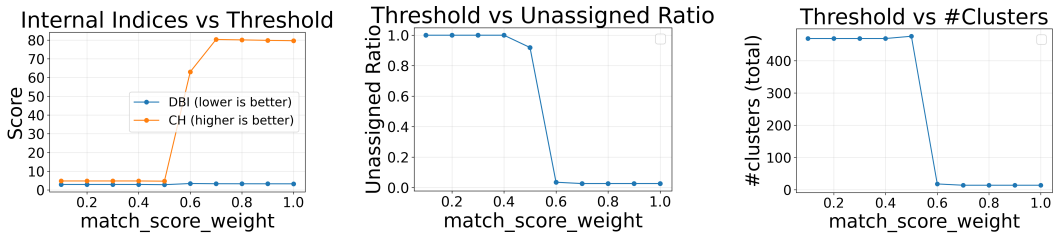

Figure 17: Match score weight sensitivity analysis for match cluster module.(left) Davies–Bouldin Index and Calinski–Harabasz Index for internal indices. (middle) Outlier Ratio for proper threshold. (right) Cluster Stability.

Table 6: Impact of Random Seed Variation on SAGE

Seed	123				42				7			
	EM	MAE	MSE	NER	EM	MAE	MSE	NER	EM	MAE	MSE	NER
Adp_1	84.95	29185	1.5×10^{10}	78.58	98.28	11	3.9×10^{4}	81.36	94.92	547	9.6×10^{7}	94.59
Adp_2	99.80	0.05	1.36	99.40	95.63	1561	5.1×10^{8}	96.39	99.80	0.01	0.05	100
Adp_3	99.80	0.01	0.16	100	99.60	0.05	1.36	100	98.99	4.56	5.7×10^{3}	99.00
Seed	256				1234				898			
	EM	MAE	MSE	NER	EM	MAE	MSE	NER	EM	MAE	MSE	NER
Adp_1	100	0	0	98.79	96.83	6.48	1.0×10^{3}	100	90.45	484	2.3×10^{5}	87.59
Adp_2	96.38	4.96	1.2×10^{3}	99.80	99.60	0.02	0.86	100	100	0	0	100
Adp_3	100	0	0	98.79	97.58	5.55	2.4×10^{3}	98.79	94.58	11.67	1.7×10^{4}	93.27

E INDUCING ATOMICITY VIA CHAIN-OF-THOUGHT

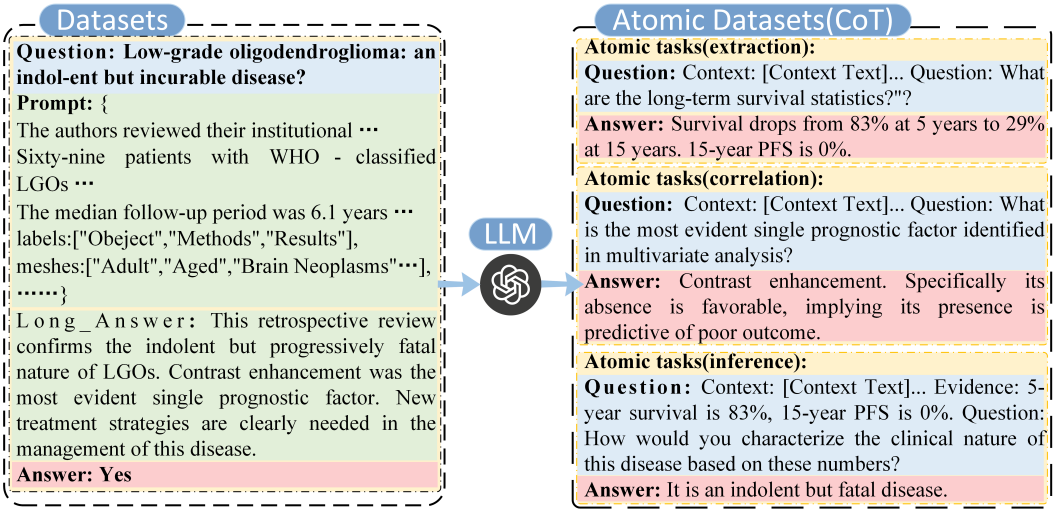

Figure 18: Case of inducing atomicity via Chain-of-Thought.

To achieve strong adaptation with limited data, the quality and structure of training samples become critical. In particular, effective updates require data with clear and decomposable reasoning paths, rather than entangled supervision signals. However, many real-world tasks, such as biomedical reasoning in PubMedQA and legal reasoning in LexGLUE, involve complex and intertwined decision processes, which hinder efficient learning. To address this issue, we introduce a Chain-of-Thought (CoT) induction step to restructure the data. Specifically, for each instance, we decompose the reasoning process into a sequence of explicit intermediate steps, forming a more structured and fine-grained supervision signal. This transformation effectively converts entangled tasks into more atomic reasoning units, making them more suitable for targeted adaptation. We apply this CoT-based preprocessing to both PubMedQA and LexGLUE, where SAGE originally underperforms. Empirically, using the induced data leads to significantly improved performance, validating the

importance of structured reasoning for efficient adaptation. A representative example of such atomic decomposition is shown in Figure 18.

F THE USE OF LLMS

LLMs were used in a limited capacity to (i) suggest related work, which was manually verified by the authors for accuracy and relevance, and (ii) proofread the manuscript for grammar and tense. The models were not involved in research design, experiments, or interpretation. All scientific contributions and claims are solely the responsibility of the authors.