SAGE: Online Continual Knowledge Distillation via Boundary Detection and Low-Rank Adaptation

Anonymous ACL submission

Abstract

Small language models are widely deployed for their efficiency, but their limited capacity leads to performance degradation on unseen data after deployment. While knowledge distillation transfers capabilities from large models, it is typically performed offline, preventing continuous teacher supervision after deployment. We propose SAGE, an online continual distillation framework that enables a deployed small model to receive targeted supervision from a teacher model under streaming conditions. SAGE integrates three modules: a Trigger module that detects when the student exceeds its competence boundary through label-free fusion of teacher–student discrepancies; a Buffer module that organizes triggered samples into semantically coherent knowledge regions via streaming clustering with stability filtering; and a Debugger module that performs two-stage low-rank parameter search to produce adaptation configurations for targeted knowledge injection. Experiments demonstrate that SAGE achieves substantial improvements in sample efficiency and adaptation accuracy.

1 Introduction

Large language models have demonstrated performance on reasoning and knowledge-intensive tasks (Brown et al., 2020; Touvron et al., 2023; Bai et al., 2023), but their high computational cost limits their deployment in resource-constrained settings (Kaplan et al., 2020; Dettmers et al., 2022, 2023). Small language models are widely adopted, yet degrade under distribution shifts (Sanh, 2019; Koh et al., 2021), motivating knowledge distillation (KD) as a bridge (Hinton et al., 2015). However, KD is inherently static, covering the training distribution and leaving models blind to evolving inputs (Fig. 1(a)). We call this boundary saturation, where inputs move beyond supervised regions and cause capability decay while the model stays fixed.

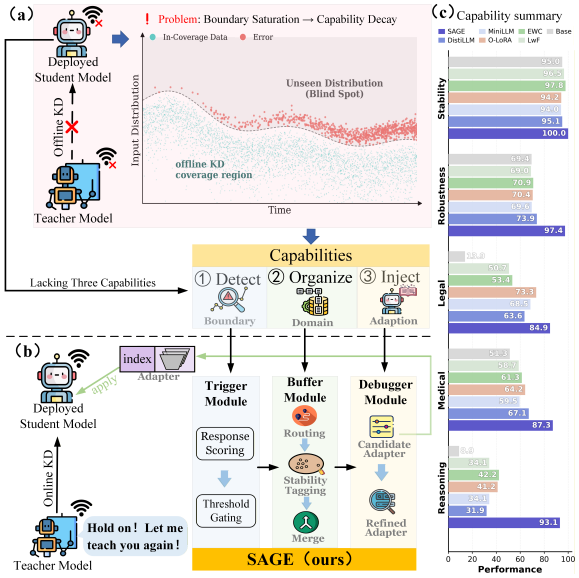

Figure 1: (a) Offline KD leads to boundary saturation, degrading performance. (b) SAGE (ours). An online loop with three modules: Trigger (detect), Buffer (organize), and Debugger (inject).(c) SAGE resolves boundary saturation via online KD, improving performance.

While existing continual learning methods are not specifically designed for boundary saturation. For example, regularization-based methods such as Elastic Weight Consolidation (Kirkpatrick et al., 2017) constrain parameter updates to preserve prior knowledge but introduce no external supervision (Parisi et al., 2019). Replay-based methods (Lopez-Paz and Ranzato, 2017; Rolnick et al., 2019) retain samples at high memory cost and remain unaware of the model's competence boundary. Distillation-based methods (e.g., LwF (Li and Hoiem, 2017) and more recent variants such as MiniLLM (Gu et al., 2024) and DistillM (Ko et al., 2024, 2025)) incorporate teacher supervision, but remain largely offline and insensitive to real-time distribution shifts (De Lange et al., 2021). These methods rely on predefined training supervision and cannot provide continuous teacher guidance after deployment, leaving them unable to correct

failures under distribution shift. Consequently, they lack three key capabilities: *Detect*, identifying inputs beyond the student's competence; *Organize*, grouping failures into domains; and *Inject*, applying targeted updates with teacher supervision.

To this end, we propose SAGE in Fig. 1(b), an online continual distillation framework for dynamic data streams. Specifically, the *Trigger* module identifies competence boundary violations by aggregating teacher–student discrepancies across lexical, semantic, and confidence-based dimensions into a anomaly score. Then, the *Buffer* module processes these anomaly samples, which are then organized into semantically coherent and temporally stable knowledge regions via streaming clustering to ensure that adaptation targets converged data rather than transient noise. Finally, the *Debugger* module executes a two-stage low-rank adaptation search by combining a coarse configuration exploration of hyperparameters with local fine-grained refinement to produce higher-performance updates. Together, these modules enable the continuous and structured internalization of teacher supervision under evolving streaming conditions.

To evaluate SAGE's core adaptation capability, we conduct experiments on models of different sizes and from different families. We evaluate SAGE across five key capability dimensions, including Stability, Robustness, Legal knowledge, Medical knowledge, and Reasoning, as summarized in Fig. 1(c). SAGE consistently achieves strong state-of-the-art performance and outperforms all baselines across all settings. Furthermore, performance improves steadily as the number of distillation cycles increases, demonstrating that SAGE enables genuine continual learning rather than one-shot adaptation.

2 Related Work

2.1 Continual Learning and Self-Adaptive Mechanisms

Regularization-based methods (e.g., EWC (Kirkpatrick et al., 2017)) constrain parameter updates but degrade over long sequences. Replay-based approaches (e.g., GEM (Lopez-Paz and Ranzato, 2017)) use memory buffers, incurring storage and task dependence. Some methods (e.g., Learning without Forgetting (Li and Hoiem, 2017)) preserve prior knowledge by constraining predictions, but rely on an incomplete teacher signal. In LLMs, self-adaptive mechanisms provide a complementary direction. FFN layers implicitly encode key–value memory (Geva et al., 2021; Berges et al., 2024; Li et al., 2025), while Memory^3 (Yang et al., 2024) and MemoryLLM (Wang et al., 2024) support modular and incremental updates. Recent work on self-improvement shows that models can refine representations without external labels (Huang et al.; Prasad et al.; Shafayat et al., 2025). However, these approaches are limited to static adaptation, while our work enables online detection, organization, and adaptation.

2.2 Boundary-Aware Adaptation with Structured Efficiency

Boundary-aware adaptation requires identifying failure regions and updating parameters efficiently. OOD detection methods (e.g., Mahalanobis scoring (Lee et al., 2018), energy-based models (Liu et al., 2020)) identify inputs requiring adaptation, while clustering methods (e.g., DenStream (Cao et al., 2006), MuDi-Stream (Amini et al., 2016), HDBSCAN (McInnes et al., 2017)) group evolving data distributions. For adaptation, parameter-efficient fine-tuning methods such as LoRA enable efficient updates via low-rank adapters, with extensions exploring sparsity, modularity, and dynamic selection (Shazeer et al., 2017; Yeasar et al., 2025; Liao et al., 2025; Ostapenko et al., 2024; Valipour et al., 2023; Xu et al., 2024; Kong et al., 2024; Wang et al., 2023b). However, these approaches are typically studied in isolation, without a unified framework under streaming conditions.

2.3 Knowledge Distillation

Once failure regions are identified and clustered, a reliable teacher signal is needed to guide parameter updates. Existing KD approaches can be broadly categorized into response-based methods, which align output distributions (Hinton et al., 2015), and feature-based methods, which match intermediate representations (Srivastava et al., 2015). Recent work extends KD to large language models, focusing on generative and reasoning capabilities (e.g., DistillLM (Ko et al., 2024, 2025), MiniLLM (Gu et al., 2024)). Other extensions incorporate retrieval (Lewis et al., 2020; Liu et al., 2024; Borgeaud et al., 2022) or long-context modeling (Wang et al., 2023a), which are largely complementary to distillation. In contrast, existing KD methods are largely offline and lack post-deployment adaptation, while our work enables online continual distillation.

3 Design of SAGE

Overview. To address the three missing capabilities: *Detect*, *Organize*, and *Inject*, SAGE instantiates a corresponding three-module framework: Trigger, Buffer, and Debugger. As shown in Fig. 2, these modules form a closed adaptation loop around the deployed student model. At each time step t, an incoming input x^t is first passed through the Trigger module $\mathcal{T}$, which decides whether it lies within the student's competence boundary. Inputs that pass are answered normally; those that fail are converted into structured teacher-student pairs and enter the Buffer module $\mathcal{B}$, where they are routed and accumulated until a semantically coherent, temporally stable cluster forms. Only upon stabilization does the cluster advance to the Debugger module $\mathcal{U}$, which searches for an optimal low-rank adapter and injects the distilled knowledge back into the student. The student thus grows stronger precisely where it was weakest. The overall system evolution is formulated as:

$$M^{t+1} = M^t + \mathcal{U}\big(\mathcal{B}(\mathcal{T}(x^t; M^t, T))\big). \quad (1)$$

3.1 Trigger

The trigger module is responsible for initiating dynamic adaptation by detecting student model competence boundary violations. As shown in Figure 2, our Trigger module consists of two components: response scoring and threshold gating.

Response Scoring To improve detection accuracy, we use three evaluation levels: token-level confidence from logits (*Logits Margin*), sequence-level overlap (*BLEU* and *ROUGE-L*), and semantic similarity from embeddings (*Embedding Similarity*). As shown in Fig. 2, after the input $x^t \in \mathbb{R}^d$ at time t is forwarded to both the teacher and student model, the student model produces output logits, from which the Logits Margin (LM) (Liu et al., 2016) is computed based on the Linear layer output in Fig. 2 to evaluate the student's confidence in predicting the next token. Assume that the logits of the student model output form a vector $z \in \mathbb{R}^V$, where V is the vocabulary size. Then we have,

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

The next token obtained by applying the softmax function in Fig. 2 is treated as the student's predicted token, which is then compared against the teacher's reference answer. We use BLEU (Papineni et al., 2002) for local exact matching and ROUGE-L (Lin, 2004) for global sequence matching. Specifically, we adopt a variant of BLEU-4 to measure the 4-gram overlap between the student's predicted answer and the teacher's reference answer, defined using 4-gram precision p_4 and weights w_4:

$$B_4 = \exp\left(\sum_{n=1}^{N} w_4 \log p_4\right) \quad (3)$$

ROUGE-L, by contrast, measures the longest common subsequence regardless of contiguity. We use the F_1 score, computed as the harmonic mean of precision P and recall R, and define it,

$$F_1 = \frac{2 \cdot P \cdot R}{P + R} \quad (4)$$

Finally, the student's predicted answer and the teacher's reference answer are encoded into embeddings in Fig. 2 to measure their semantic similarity. Let $\mathbf{Emb}_{pred}(x^t)$, $\mathbf{Emb}_{teacher}(x^t)$ denote the embedding vectors of the student's predicted answer and the teacher's reference answer, respectively. The embedding similarity (Lin et al., 2017) ES is,

$$\text{ES} = \frac{\mathbf{Emb}_{pred}(x^t) \cdot \mathbf{Emb}_{true}(x^t)}{\|\mathbf{Emb}_{pred}(x^t)\| \cdot \|\mathbf{Emb}_{true}(x^t)\|} \quad (5)$$

To obtain the final Anomaly Score $\mathcal{AS}$, where higher values indicate greater discrepancy between the teacher and student outputs, we perform normalization and integration of Equation (2), Equation (3), Equation (4) and Equation (5) as follows,

$$\mathcal{AS}^t = \sum_{i=1}^{4} w_i(1 - s_i) \quad (6)$$

where $s_1 = LM/LM_m$, $s_2 = B_4$, $s_3 = F_1$, and $s_4 = ES$. Here, LM_m denotes the upper bound used to normalize the margin score, which is set to 5.0 in our experiments. The weight w_i corresponds to each evaluation metric.

Threshold Gating. The input x^t is processed by the Response Scoring to produce the anomaly score $\mathcal{AS}^t$. A threshold τ is applied to define the gating decision function g^t: when $g^t = 1$, the input x^t is identified as exceeding the student's competence boundary and is forwarded to the Buffer module for structured knowledge transfer. The formula is,

$$g^t = \begin{cases} 1, & \mathcal{AS}^t \geq \tau \\ 0, & \mathcal{AS}^t < \tau \end{cases} \quad (7)$$

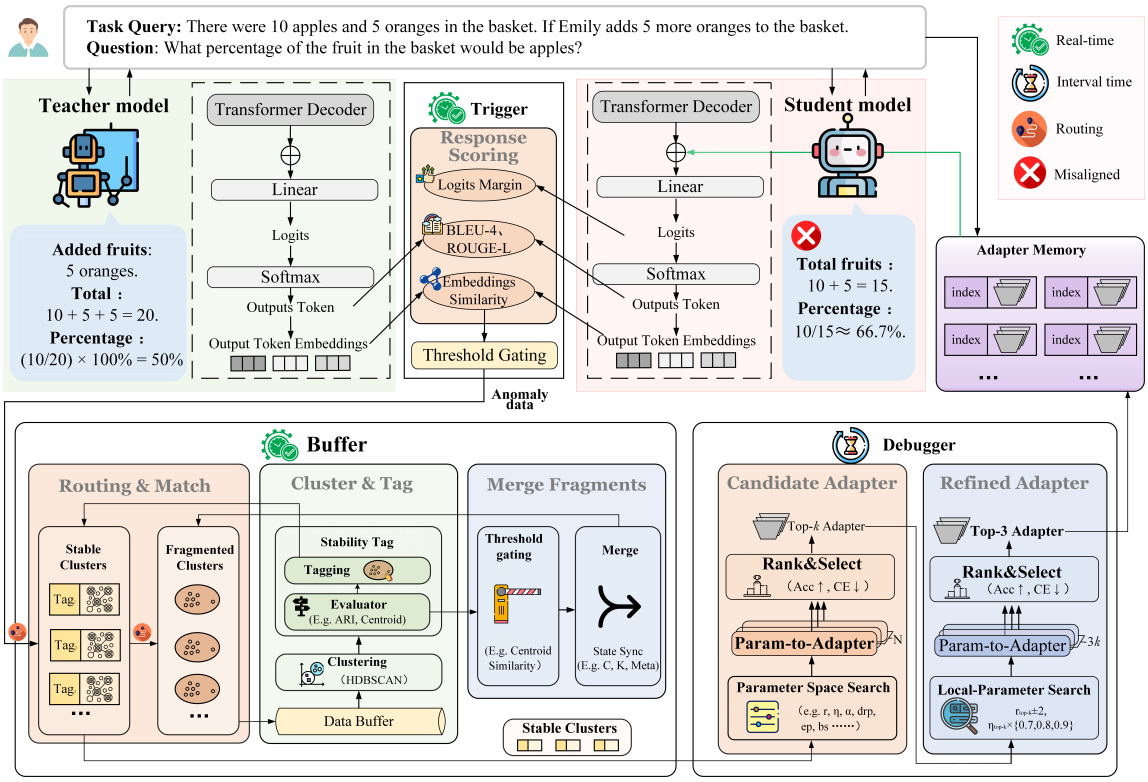

Figure 2: Overview of our SAGE framework. The Trigger module detects anomaly data by comparing predictions between student model and teacher model, injects data into the Buffer module for clustering, and commits the clusters to the Debugger for fine-tuning. The resulting adapter is finally applied to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

3.2 Buffer

Upon receiving an anomaly pair from Trigger, Buffer decides where to place it and when to trigger adaptation, through three steps: routing and matching; clustering and tagging; and fragment merging.

Routing and Match We partition the data into buckets (knowledge scope) and, for each input x^t, identify its bucket and retrieve stability labels $\{s_i^t\}$, where $s_i^t = 1$ indicates a stable cluster (maintained from previous clustering passes). If no stable clusters exist, x^t is stored in the buffer. Otherwise, it is matched to existing clusters via:

$$Match = \alpha \cdot \cos(\mathbf{e}_x, \mathbf{e}_c) + (1-\alpha) \cdot \frac{|k_{x^t} \cap k_{\mathcal{C}^t}|}{\max |k_{\mathcal{C}}|} \tag{8}$$

where $\mathbf{e}_x$ and $\mathbf{e}_c$ are embeddings of x^t and cluster center $\mathcal{C}_c^t$. If the Match score exceeds a threshold, x^t is assigned to the cluster; otherwise, a new cluster is created (fragment).

Cluster and Tag We check whether the number of samples in the bucket of x^t exceeds a threshold; otherwise, clustering is skipped. Once triggered, we apply HDBSCAN (Campello et al., 2013) to group samples under evolving data distributions. However, due to streaming inputs, the resulting clusters may be unstable, which can affect subsequent LoRA fine-tuning. To address this, we introduce two metrics to quantify clustering stability:

1. Adjusted Rand Index (ARI), measuring label consistency across successive clusterings:

$$ARI = \frac{RI - \mathbb{E}[RI]}{\max(RI) - \mathbb{E}[RI]} \tag{9}$$

2. Average Cosine Similarity, measuring semantic consistency between cluster centroids:

$$Emb_C = \cos(\mathbf{e}_c^{t-m}, \mathbf{e}_c^t) \tag{10}$$

Given ARI and Emb_C, we assign a cluster as stable ($s_i^t = 1$) if either exceeds its threshold:

$$ARI > \tau_{ARI} \ \cup \ Emb_C > \tau_{Emb} \ \Rightarrow \ s_i^t = 1 \tag{11}$$

If the stability label s_i^t of a cluster changes, we update as part of the Cluster-Stability Routing for the next input x^{t+1}. Clusters that remain stable but contain few samples are treated as fragmented.

Merge Fragments Fragment clusters $|\mathcal{F}^t|$, if left unmerged, dilute stable regions and introduce routing noise. We merge them when $|\mathcal{F}^t|$ exceeds a threshold and two clusters are merged if:

$$\cos(\mathbf{e}_c^i, \mathbf{e}_c^j) > \tau \tag{12}$$

If no pair satisfies this condition, clustering terminates; otherwise, merged clusters are updated (e.g., centroids, keywords, metadata) synchronously so that the Routing and Match stage can correctly handle the subsequent input x^{t+1}.

3.3 Debugger

Candidate Adapter After a fixed interval T, we fine-tune the student on each stable cluster $\mathcal{C}_s$. We define a hyperparameter space Θ over LoRA rank, learning rate, and other training parameters, and sample N configurations:

$$\Theta = \{\theta_1, \theta_2, \dots\}, \quad \mathcal{P}_n \sim \Theta,\ n = 1, \dots, N \tag{13}$$

Each configuration yields an adapter y_n^T, evaluated by accuracy Acc and cross-entropy loss CE on $\mathcal{C}_s$. Adapters are ranked by Acc (descending) and CE (ascending), and the top-k are selected:

$$\arg \mathit{Top}_k^{\prec lex}\{Acc(y_n^T), -CE(y_n^T)\}_{n=1}^{N} \tag{14}$$

Refined Adapter The coarse search may miss the local optimum. Starting from the top-k configurations, we perform a local search over rank and learning rate (most sensitive hyperparameters). For each configuration, we construct a neighborhood:

$$\begin{aligned}\mathcal{N}_{k_i} =& \{\theta_{r,k_i} - 2, \theta_{r,k_i}, \theta_{r,k_i} + 2\} \\ & \times \{0.7\theta_{l,k_i}, 0.8\theta_{l,k_i}, 0.9\theta_{l,k_i}\}\end{aligned} \tag{15}$$

This yields $3k$ candidate configurations. We fine-tune and rank them using the same criterion, selecting the top-3 adapters:

$$\arg \mathit{Top}_3^{\prec lex}\{Acc(y_n^T), -CE(y_n^T)\}_{n=1}^{3k} \tag{16}$$

The selected adapters are stored for future reuse. When a new input x^{t+m} is routed to a matching cluster, the corresponding adapter is loaded to enhance the student's response.

4 Experiments

4.1 Experimental Settings

Model and Implementation Details Our experiments are conducted on a single NVIDIA A100 (80GB) GPU. We use Qwen2.5-1.5B, Qwen2.5-3B, Qwen2.5-7B, LLaMA-2-7B, and Mistral-7B as base models. For trigger detection and clustering, we adopt BGE-large-en-v1.5 as the sentence encoder, since decoder-only representations are less effective for sentence-level similarity. Teacher signals from the OpenAI GPT-5 API (85.71% OOD accuracy) are relatively unstable. Therefore, we use ground-truth labels to simulate an ideal teacher signal for fair comparison.

Datasets and Evaluation Setup We evaluate both in-distribution (ID) and out-of-distribution (OOD) performance on four datasets. TriviaQA (Joshi et al., 2017) serves as the ID benchmark. GSM8K (math) (Cobbe et al., 2021), PubMedQA (biomedical) (Jin et al., 2019), and LexGLUE (legal) (Chalkidis et al., 2022) are used as OOD benchmarks. GSM8K is evaluated using Exact Match (EM), Mean Absolute Error (MAE), and Numeric Extraction Rate (NER). PubMedQA and LexGLUE are evaluated using Accuracy (Acc) and F1. For PubMedQA and LexGLUE, we apply Chain-of-Thought decomposition to break each instance into shorter reasoning steps, which are then used by SAGE.

Baselines We compare SAGE with representative methods from three categories: *(1) Parameter-efficient adaptation:* Base Model (no adaptation), LoRA (Hu et al., 2022), O-LoRA (Wang et al., 2023b), and CA-LoRA (offline clustering-based adaptation); *(2) Continual learning:* EWC (Kirkpatrick et al., 2017) and LwF (Li and Hoiem, 2017); *(3) Distillation-based methods:* MiniLLM (Gu et al., 2024) and DistilLM (Ko et al., 2024). CA-LoRA is trained on statically clustered data, while all other baselines use the original datasets. Data are split into train/validation/test sets with ratios of 80%/10%/10%.

4.2 Main Results

Baseline Comparison The results in Table 1 confirm that SAGE's three-module design effectively addresses the failure modes of static distillation. SAGE consistently outperforms all baselines across datasets and model scales, with larger gains under OOD settings. For example, on Qwen2.5-1.5B, SAGE improves GSM8K EM from 36.46% (O-LoRA) to 47.70% (+11.24%), and boosts PubMedQA accuracy from 69.48% to 85.64% (+16.16%). On Qwen2.5-7B, SAGE further achieves 91.32 EM on GSM8K, outperforming the strongest baseline (81.36%) by +9.96%. These gains come from boundary-aware adaptation. Trigger resolves the *Detect* gap, focusing updates on failure regions (+28.32% PubMedQA). Buffer resolves the *Organize* gap, reducing noisy supervision (MAE 1515 vs. 4126). Debugger resolves the *Inject* gap, enabling fine-grained updates (99.80% EM on LLaMA-2-7B). Therefore, SAGE achieves consistent improvements by replacing static distillation with targeted, online adaptation.

Table 1: 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) Adp_1	**47.70** ↑	**1226**↓	**100**↑	76.92	75.71	**86.67**↑	**86.67**↑
	SAGE (ours) Adp_2	36.47	2019	100	81.11	83.42	74.47	74.47
	SAGE (ours) Adp_3	32.46	1805	100	**85.64**↑	**85.67**↑	83.35	83.35
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) Adp_1	57.31	1301	100	76.92	75.71	**86.67**↑	**86.67**↑
	SAGE (ours) Adp_2	58.12	1352	99.80	91.11	93.42	74.47	74.47
	SAGE (ours) Adp_3	**58.23** ↑	**1515**↓	**100**↑	**95.64**↑	**95.67**↑	83.35	83.35
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) Adp_1	86.20	196	94.39	**89.12**↑	**87.56**↑	83.99	84.56
	SAGE (ours) Adp_2	86.95	160	95.19	78.81	74.29	**92.00**↑	**92.13**↑
	SAGE (ours) Adp_3	**91.32** ↑	**45**↓	92.38	78.64	75.60	86.67	86.67
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) Adp_1	84.95	28185	100	90.17	87.33	**87.14**↑	**87.14**↑
	SAGE (ours) Adp_2	99.80	0.05	99.40	89.36	81.67	84.52	84.52
	SAGE (ours) Adp_3	**99.80**↑	**0.01**↓	**100**↑	**95.70**↑	**95.37**↑	86.29	86.29
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) Adp_1	95.56	143576	99.20	76.92	75.71	**86.67**↑	**86.67**↑
	SAGE (ours) Adp_2	**97.59** ↑	**7.29**↓	**99.60**↑	91.11	93.42	74.47	74.47
	SAGE (ours) Adp_3	96.17	25.52	99.40	**95.64**↑	**95.67**↑	83.35	83.35

Continual Learning and Retention 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

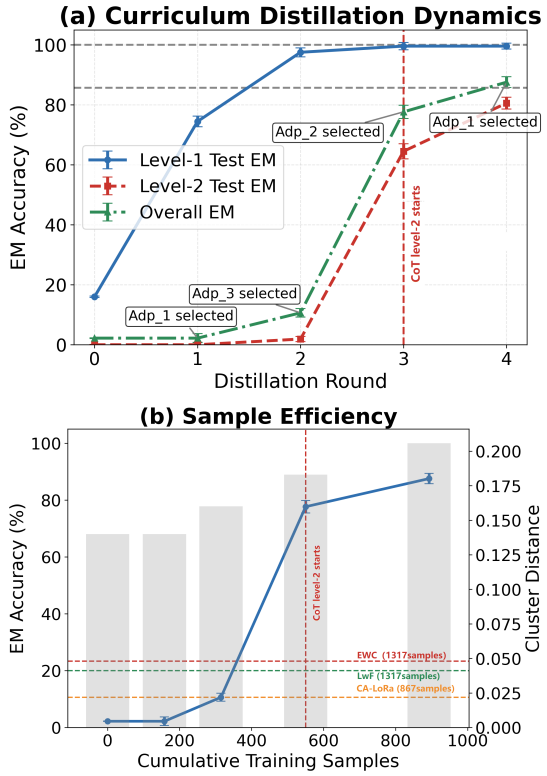

Figure 3: Curriculum learning and sample-efficient adaptation under SAGE. (a) EM accuracy across distillation rounds, highlighting the transition to Level-2 reasoning. (b) Overall accuracy versus cumulative training samples, with improving cluster structure quality over time.

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.

5 Ablation Study

5.1 Component Ablation

Table 2 presents component-wise ablation results on GSM8K Level-1 under identical settings. The debugger is not ablated here, as its effect is analyzed at the parameter level. *w/o Trigger.* Performance collapses, showing that boundary-aware filtering is essential; without it, mixed data produce non-separable training signals and unstable adaptation. *w/o Buffer.* Removing or replacing the buffer with static clustering yields unstable performance across adapters, indicating that consistent

Table 2: Ablation Study of SAGE on the GSM8K Dataset.

Method		Dataset Size	GSM8K(%)		
			EM	MAE	NER
w/o Trigger (random)	Adp_1	499	6.44	11609	99.17
	Adp_2		8.89	11594	100
	Adp_3		6.72	14995	99.17
w/o Buffer	Adp_1	499	92.32	95	93.99
	Adp_2		81.27	473	63.13
	Adp_3		72.13	13917	57.52
w/o Buffer (HDBSCAN)	Adp_1	484	89.77	309	98.97
	Adp_2		78.70	21812	89.26
	Adp_3		96.69	429	100
w/o Buffer (K-means)	Adp_1	329	97.55	14	99.09
	Adp_2		74.47	377	100
	Adp_3		92.40	646	100
w/o Trigger w/op Buffer	Adp_1	499	92.32	95	93.99
	Adp_2		81.27	473	63.13
	Adp_3		72.13	13917	57.52
SAGE (full)	Adp_1	315	84.95	28185	78.58
	Adp_2		**99.80**	**0.05**	**99.40**
	Adp_3		**99.80↑**	**0.01↓**	**100↑**

knowledge organization is required under streaming settings. *Full SAGE.* The full model achieves near-perfect and stable performance. This confirms that all three capabilities (*Detect*, *Organize*, and *Inject*) are jointly necessary for reliable continual distillation.

5.2 Sensitivity Analysis

Trigger *Separability Across ID/OOD Sources.* To evaluate whether the 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 reliable boundary detection. *Robustness to Threshold and Indicator Weights.* To assess whether the trigger requires careful tuning, we further 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 that the trigger is robust to hyperparameter choices and generalizes well across settings.

Buffer *Clustering Quality Across Streaming Variants.* To evaluate whether the buffer produces reliable clusters under streaming conditions, we compare different clustering strategies. Fig. 5(a) shows that HDBSCAN alone is unstable, while adding stability tagging and fragment merging improves

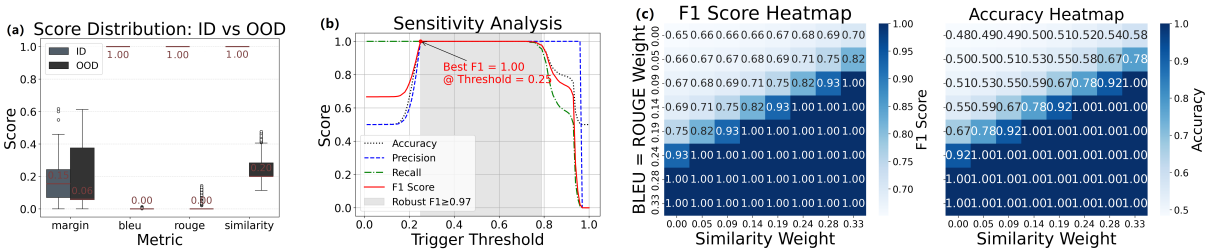

Figure 4: Parameter analysis of the Trigger Module. (a) False positive rates on ID vs. OOD samples. Red lines in boxplots indicate the mean; the logits margin metric is normalized. (b) Threshold sensitivity analysis; the gray area marks the optimal threshold range. (c) Heatmap of score weight sensitivity.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

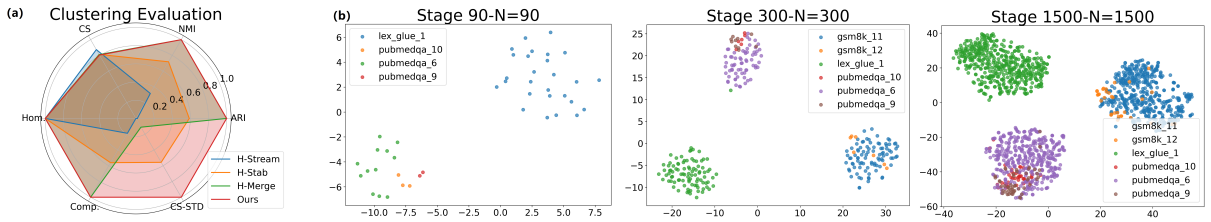

Figure 5: Results of the Buffer Module evaluation. (a) Evaluation of Streaming HDBSCAN Variants on Clustering Metrics: Homogeneity, ARI, NMI, Completeness, Central Similarity (CS), and Cluster Sample Std. (CS-STD). (b) Dynamic t-SNE Visualization as Data Volume (N) Increases.

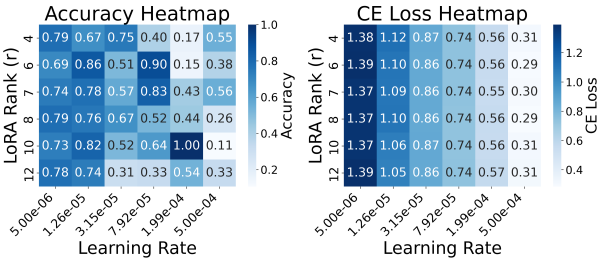

Figure 6: Heatmap of accuracy and cross-entropy loss for LoRA parameter: examining the necessity of of rank and learning rate for exploitation optimization.

consistency. The full SBC achieves the best overall clustering quality, indicating that all of 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 clear convergence to a stable structure.

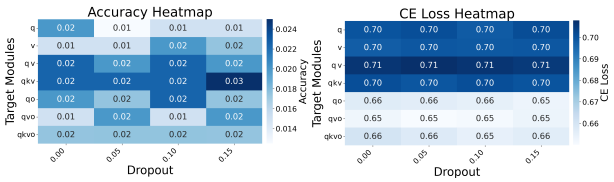

Figure 7: Heatmap of accuracy and cross-entropy loss for LoRA parameter: examining the necessity of dropout and target modules for exploitation optimization.

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 consistently 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 further analyze dropout and target module selection. Fig. 7 shows minimal variation across configurations, indicating consistently 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 more efficiently while fixing other parameters.

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.

Limitations

We acknowledge the following limitations of our work:

- SAGE relies on task atomicity and stable low-rank gradient directions. For highly entangled reasoning tasks, the clustering process may become unstable, reducing the effectiveness of selective LoRA adaptation.

- Although the trigger-buffer mechanism effectively prevents noisy updates from contaminating the student model, the current framework mainly focuses on local and sparse adaptation. Streaming environments may still introduce memory interference and accumulated structural drift across clusters.

- While sparse triggering significantly reduces the average adaptation cost, the per-cluster LoRA optimization overhead remains relatively high. Improving asynchronous execution, routing efficiency, and scalable memory organization remains important future work.

References

Amineh Amini, Hadi Saboohi, Tutut Herawan, and Teh Ying Wah. 2016. Mudi-stream: A multi density clustering algorithm for evolving data stream. *Journal of Network and Computer Applications*, 59:370–385.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, and 1 others. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Vincent-Pierre Berges, Barlas Oğuz, Daniel Haziza, Wen-tau Yih, Luke Zettlemoyer, and Gargi Gosh. 2024. Memory layers at scale. *arXiv e-prints*, pages arXiv–2412.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, and 1 others. 2022. Improving language models by retrieving from trillions of tokens. In *International conference on machine learning*, pages 2206–2240. PMLR.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Ricardo JGB Campello, Davoud Moulavi, and Jörg Sander. 2013. Density-based clustering based on hierarchical density estimates. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 160–172. Springer.

Feng Cao, Martin Estert, Weining Qian, and Aoying Zhou. 2006. Density-based clustering over an evolving data stream with noise. In *Proceedings of the 2006 SIAM international conference on data mining*, pages 328–339. SIAM.

Ilias Chalkidis, Abhik Jana, Dirk Hartung, Michael Bommarito, Ion Androutsopoulos, Daniel Katz, and Nikolaos Aletras. 2022. 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)*, pages 4310–4330.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

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

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

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

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. 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.

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

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

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.

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*.

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua Lu. 2019. 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)*, pages 2567–2577.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. 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)*, pages 1601–1611.

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

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, and 1 others. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526.

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

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

Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, and 1 others. 2021. Wilds: A benchmark of in-the-wild distribution shifts. In *International conference on machine learning*, pages 5637–5664. PMLR.

Rui Kong, Qiyang Li, Xinyu Fang, Qingtian Feng, Qingfeng He, Yazhu Dong, Weijun Wang, Yuanchun Li, Linghe Kong, and Yunxin Liu. 2024. Lora-switch: Boosting the efficiency of dynamic llm adapters via system-algorithm co-design. *CoRR*.

Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. 2018. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. *Advances in neural information processing systems*, 31.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

Dongfang Li, Zetian Sun, Xinshuo Hu, Baotian Hu, and Min Zhang. 2025. Cmt: A memory compression method for continual knowledge learning of large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 24413–24421.

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

Xiaoxuan Liao, Chihang Wang, Shicheng Zhou, Jiacheng Hu, Hongye Zheng, and Jia Gao. 2025. Dynamic adaptation of lora fine-tuning for efficient and task-specific optimization of large language models. pages 120–125.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. In *International Conference on Learning Representations*.

Jingyu Liu, Jiaen Lin, and Yong Liu. 2024. How much can rag help the reasoning of llm? *arXiv preprint arXiv:2410.02338*.

Weitang Liu, Xiaoyun Wang, John Owens, and Yixuan Li. 2020. Energy-based out-of-distribution detection. *Advances in neural information processing systems*, 33:21464–21475.

Weiyang Liu, Yandong Wen, Zhiding Yu, and Meng Yang. 2016. Large-margin softmax loss for convolutional neural networks. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning-Volume 48*, pages 507–516.

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

Leland McInnes, John Healy, Steve Astels, and 1 others. 2017. hdbscan: Hierarchical density based clustering. *J. Open Source Softw.*, 2(11):205.

Oleksiy Ostapenko, Zhan Su, Edoardo Maria Ponti, Laurent Charlin, Nicolas Le Roux, Lucas Caccia, and Alessandro Sordoni. 2024. Towards modular llms by building and reusing a library of loras. In *Proceedings of the 41st International Conference on Machine Learning*, pages 38885–38904.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

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

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*.

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

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

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

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*.

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

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Mojtaba Valipour, Mehdi Rezagholizadeh, Ivan Kobyzev, and Ali Ghodsi. 2023. Dylora: Parameter-efficient tuning of pre-trained models using dynamic search-free low-rank adaptation. pages 3274–3287.

Weizhi Wang, Li Dong, Hao Cheng, Xiaodong Liu, Xifeng Yan, Jianfeng Gao, and Furu Wei. 2023a. Augmenting language models with long-term memory. *Advances in Neural Information Processing Systems*, 36:74530–74543.

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

Yu Wang, Yifan Gao, Xiusi Chen, Haoming Jiang, Shiyang Li, Jingfeng Yang, Qingyu Yin, Zheng Li, Xian Li, and Bing Yin. 2024. Memoryllm: Towards self-updatable large language models. *ICML*.

Jingwei Xu, Junyu Lai, and Yunpeng Huang. 2024. Meteora: Multiple-tasks embedded lora for large language models. *arXiv e-prints*, pages arXiv–2405.

Hongkang Yang, Zehao Lin, Wenjin Wang, Hao Wu, Zhiyu Li, Bo Tang, Wenqiang Wei, Jinbo Wang, Zeyun Tang, Shichao Song, and 1 others. 2024. Memory3: Language modeling with explicit memory. *arXiv preprint arXiv:2407.01178*.

Arnob Samin Yeasar, Zhan Su, Minseon Kim, Oleksiy Ostapenko, Doina Precup, Lucas Caccia, and Sordoni Alessandro. 2025. Exploring sparse adapters for scalable merging of parameter efficient experts. In *ICLR 2025 Workshop on Modularity for Collaborative, Decentralized, and Continual Deep Learning*.

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 modulesq_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 η_{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'

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

1: $s \leftarrow$ InferStructure(x)
2: $e \leftarrow$ GetEmbedding(x)
3: $k \leftarrow$ ExtractKeywords(x)
4: **if** IsClusterStable(s) == True **then**
5: $(\mathcal{C}^*, \gamma) \leftarrow$ FindBestCluster(s, e, k)
6: # γ: weighted score combining embedding similarity e and keyword overlap k
7: **if** $\gamma \geq \tau$ **then** ▷ Exceeding threshold τ
8: Assign x to cluster $\mathcal{C}^*$
9: **return** cluster assignment
10: **end if**
11: **else**
12: AddToBuffer$(x, e, k, \mathcal{D}_s)$
13: **end if**
14: **if** $|\mathcal{D}_s| \geq T$ **then** ▷ Exceeding clustering threshold T
15: $e \leftarrow$ ExtractEmbeddings$(\mathcal{D}_s)$
16: $h \leftarrow$ HDBSCAN(e)
17: ▷ Density clustering without preset count
18: $(\text{ARI}, \text{Sim}) \leftarrow$ EvalStability$(h, h^{(t-1)}, e, e^{(t-1)})$
19: **if** ARI $\geq \theta_{\text{ari}} \vee$ Sim $\geq \theta_{\text{sim}}$ **then**
20: CreateNewClusters$(h, \mathcal{D}_s)$
21: $\mathcal{D}_s \leftarrow \emptyset$ ▷ Clear buffer
22: StableFlag$_s \leftarrow$ True
23: **if** $|\mathcal{C}_s| \geq 3$ **then**
24: MergeSimilarClusters$(\mathcal{C}_s)$
25: ▷ Merge clusters with centroid similarity
26: **end if**
27: **else**
28: **return** ▷ Clustering unstable, wait more
29: **end if**
30: **end if**
31: **return** x as "Unassigned" ▷ No matching cluster

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:

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

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$\mathcal{A}$

1: $\mathcal{P}_n \leftarrow \text{InitialConfigs}(\Theta, n)$ ▷ Initialize n seed configs
2: $\mathcal{R}_0 \leftarrow []$ ▷ Initial training results
3: **for all** $\theta \in \mathcal{P}_n$ **do**
4: $\quad r \leftarrow \text{TrainLoRA}(\theta, \mathcal{C}, \mathcal{M}, \mathcal{T})$
5: $\quad \mathcal{R}_0 \leftarrow \mathcal{R}_0 \cup \{r\}$
6: **end for**
7: $\mathcal{R}_0 \leftarrow \text{SortByScore}(\mathcal{R}_0)$
8: # maximize accuracy, break ties by minimizing loss
9: $\mathcal{P}_{\text{top}} \leftarrow \text{Top-KConfigs}(\mathcal{R}_0, k)$
10: $\mathcal{R}_1 \leftarrow []$ ▷ Refined training results
11: **for all** $\theta_{\text{top}} \in \mathcal{P}_{\text{top}}$ **do**
12: $\quad \mathcal{P}_{\text{opt}} \leftarrow \text{LocalParamSearch}(\theta_{\text{top}})$
13: #local search over rank and learning rate near θ_{top}
14: $\quad$ **for all** $\theta' \in \mathcal{P}_{\text{opt}}$ **do**
15: $\quad\quad r' \leftarrow \text{TrainLoRA}(\theta', \mathcal{C}, \mathcal{M}, \mathcal{T})$
16: $\quad\quad \mathcal{R}_1 \leftarrow \mathcal{R}_1 \cup \{r'\}$
17: $\quad$ **end for**
18: **end for**
19: $\mathcal{R}_{\text{all}} \leftarrow \mathcal{R}_0 \cup \mathcal{R}_1$
20: $\mathcal{R}_{\text{all}} \leftarrow \text{SortByScore}(\mathcal{R}_{\text{all}})$
21: $\mathcal{A}_{\text{LoRA}} \leftarrow \text{Top-KResults}(\mathcal{R}_{\text{all}}, 3)$
22: $\text{SaveBuffer}(\mathcal{A}_{\text{LoRA}})$
23: **return** $\mathcal{A}_{\text{LoRA}}$ and θ^*

C Discussion

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

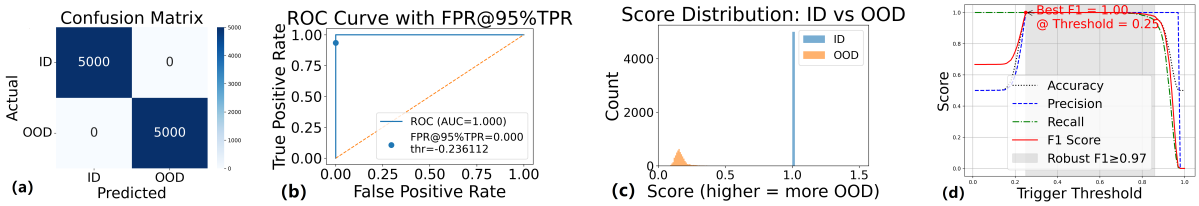

Figure 8: Trigger module evaluation on long-form OOD datasets with TriviaQA as ID. (a) Confusion matrix. (b) ROC curve. (c) Anomaly score distributions. (d) Threshold sensitivity.

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 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 9, 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 9(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.

D.2 Details for Buffer Module

Figure 10 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.

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 11 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 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 al-

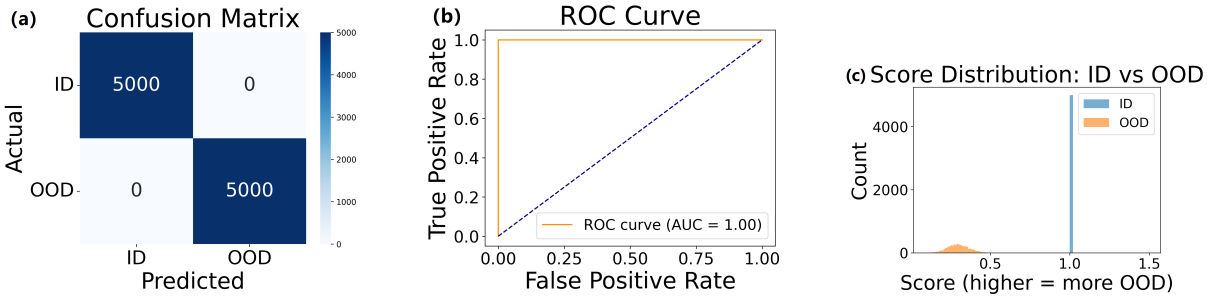

Figure 9: Results of the Trigger Module evaluation under formal test parameters. (a) Confusion matrix for ID and OOD samples. (b) ROC curve illustrating detection performance. (c)Anomaly score of ID datasets and OOD datasets.

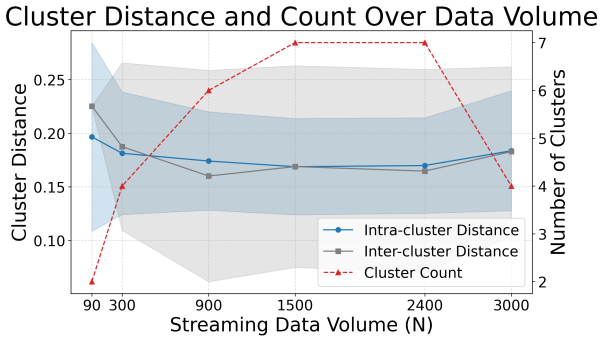

Figure 10: Cluster distance and cluster count over streaming data volume.

gorithm 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 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 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 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 14 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.3 Details for Debugger Module

To examine stability, we shuffle the dataset with different random seeds. As shown in Table 3, SAGE maintains high performance with EM $= 97.16_{\pm 4.65}$, indicating weak sensitivity to input or-

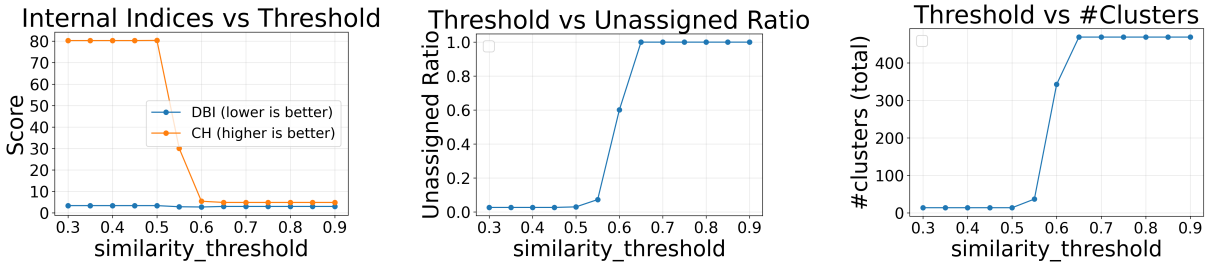

Figure 11: Similarity threshold sensitivity analysis for threshold gating after match clusters module. (left) Davies–Bouldin Index and Calinski–Harabasz Index for internal indices. (middle) Outlier Ratio for proper threshold. (right) Cluster Stability.

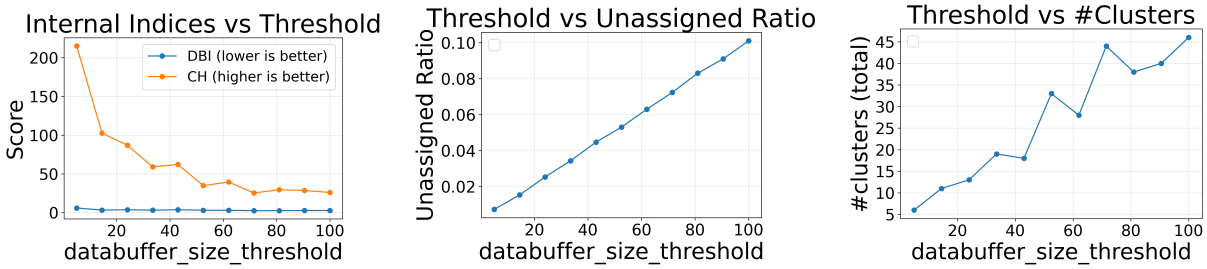

Figure 12: Data buffer size threshold sensitivity analysis for threshold gating before density cluster module. (left) Davies–Bouldin Index and Calinski–Harabasz Index for internal indices. (middle) Outlier Ratio for proper threshold. (right) Cluster Stability.

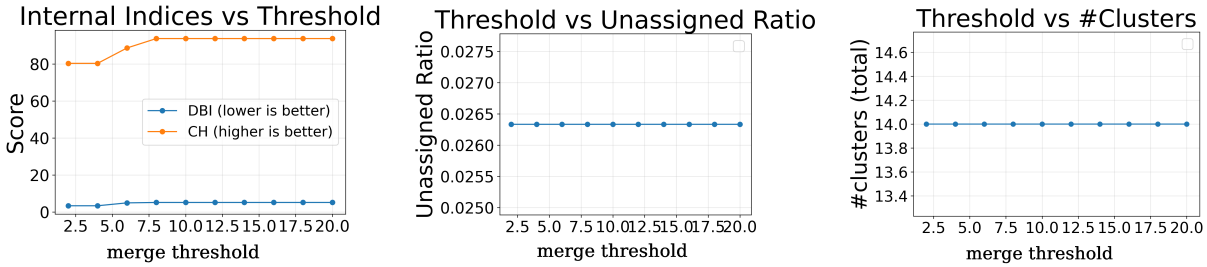

Figure 13: Merge threshold sensitivity analysis for threshold gating before merge clusters module.(left) Davies–Bouldin Index and Calinski–Harabasz Index for internal indices. (middle) Outlier Ratio for proper threshold. (right) Cluster Stability.

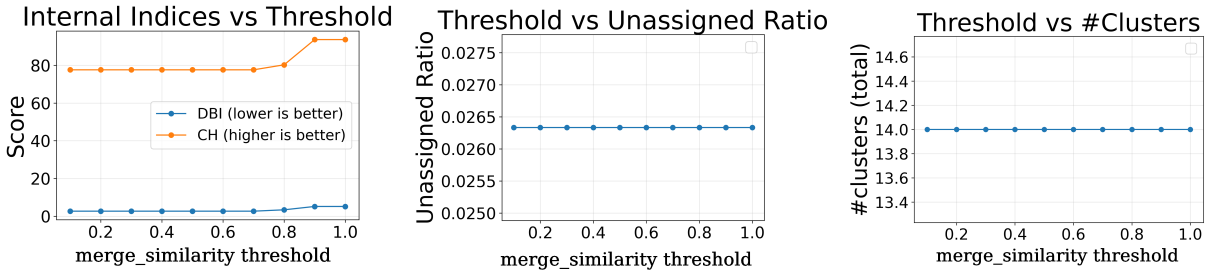

Figure 14: Cluster center similarity threshold sensitivity analysis for merge clusters module.(left) Davies–Bouldin Index and Calinski–Harabasz Index for internal indices. (middle) Outlier Ratio for proper threshold. (right) Cluster Stability.

der. 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.

D.3.1 Cross-Round Retention Analysis

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-

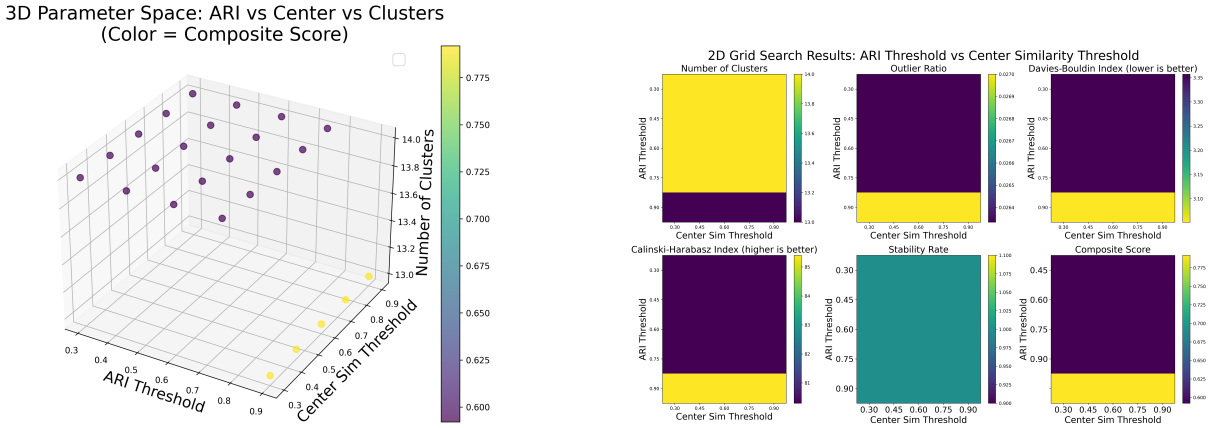

Figure 15: Ajusted Random Index and average Cosine Similarity thresholds sensitivity analysis for density cluster module. (left) 3d parameter for cluster stability of threshold. (right) 2d parameter results for cluster stability, outlier ratio, DBI, CH, stability rate and composite score.

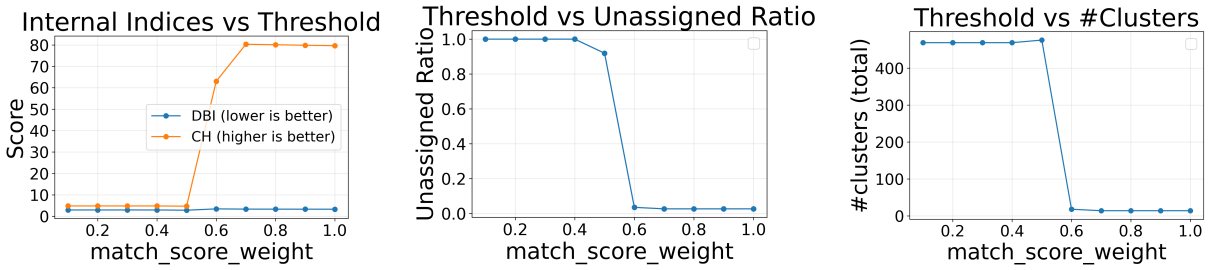

Figure 16: Match score weight sensitivity analysis for match cluster module.(left) Davies–Bouldin Index and Calinski–Harabasz Index for internal indices. (middle) Outlier Ratio for proper threshold. (right) Cluster Stability.

Table 3: 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	0.02	0.86	99.40	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

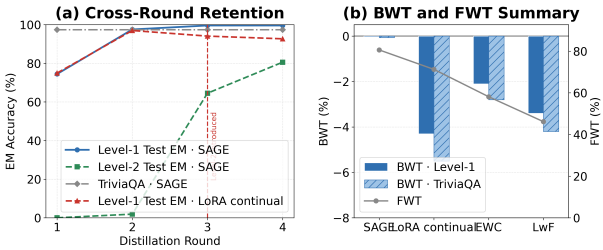

Figure 17: Cross-round knowledge retention and transfer under SAGE. (a) EM accuracy across rounds, showing stable retention and capability expansion.(b) BWT and FWT across methods, where SAGE achieves near-zero forgetting with the highest forward transfer.

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 17(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 17(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.4 Computational Efficiency Analysis

A common concern for any online adaptation framework is whether parameter updates introduce meaningful latency during inference. Accordingly, we evaluate the per-sample latency of each component, as reported in Table 4. Signals required by the Trigger are obtained directly from the forward pass, incurring only 24.01 ms on average. The Buffer module is asynchronous, so it does not block model inference, and its amortized per-sample cost is only 12.63 ms, independent of the frequency of updates or the number of clusters formed. The Debugger is the most expensive component (7726.05 ms), but it is triggered only when: ① a cluster becomes stable; ② it exceeds the minimum size threshold; and ③ the cluster's gradient structure satisfies the atomicity criterion. As a result, updates are sparse events rather than per-sample operations. Accounting for the sparsity of updates, the effective amortized cost of SAGE is only 493.11 ms per sample. During pure inference (no update events), loading an active low-rank adapter adds negligible overhead, increasing the model's forward latency from 9.44 ms to only 9.62 ms. In summary, this design ensures that the computational cost of online adaptation remains bounded while still enabling effective endogenous memory formation.

Table 4: The average latency for SAGE to process one piece of data.

Latency	Trigger	Buffer	Debugger	SAGE	Inference
(ms)	24.01	12.63	7726.05	493.11	9.62

E 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

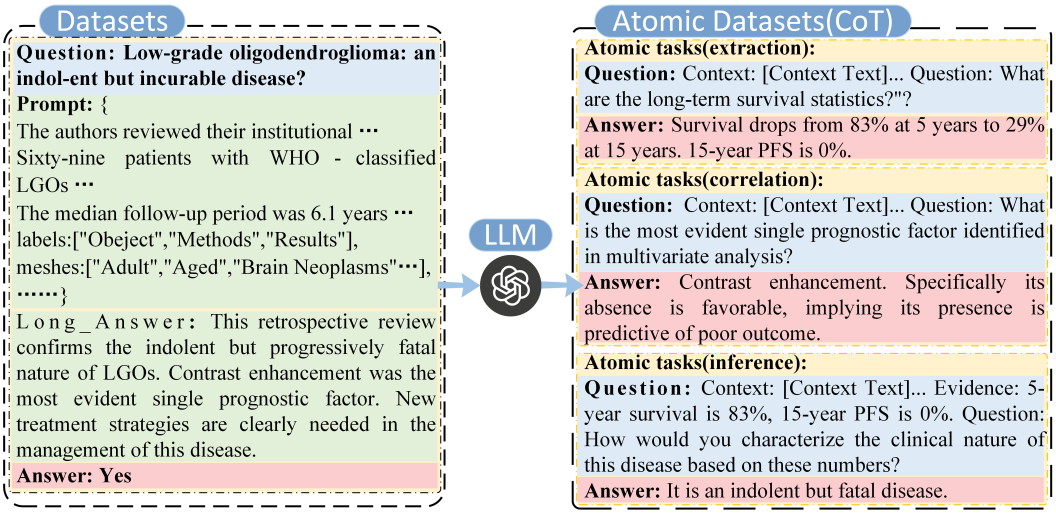

Figure 18: Case of inducing atomicity via Chain-of-Thought.

the authors.