STATECOMP: LEARNING WHEN TO COMPRESS HISTORY IN LONG HORIZON AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Long horizon agents continuously accumulate interaction history during task execution, yet the importance of past interactions changes as the agent state evolves. Existing context management methods largely compress history based on fixed windows, periodic schedules, or current relevance, overlooking a more fundamental question: when has a past interaction become safe to replace? Premature compression may remove information still needed for future actions, while overly conservative retention leads to substantial context overhead. To address this, we propose **State Conditioned Compression (STATECOMP)**, a framework that determines when historical interactions can be safely compressed according to the current agent state. STATECOMP constructs KEEP and READY supervision through a two stage annotation procedure and trains an imbalance aware router on hidden representations from a frozen language model. A bounded state representation further reduces the cost of evaluating long histories, while adjacent READY interactions are grouped into continuous spans and replaced with compact summaries during execution. Across four domains of long horizon agent tasks, STATECOMP reduces total agent and summarization tokens by 52.27% while maintaining task performance, and achieves a $12.67\times$ speedup in representation extraction.

1 INTRODUCTION

Long horizon agents continuously accumulate task instructions, intermediate reasoning, tool calls, environment observations, and unsuccessful attempts throughout execution (Yao et al., 2022; Shinn et al., 2023). As the trajectory grows, each new step may require the model to process an increasingly long interaction history (Liu et al., 2024; Bai et al., 2025), resulting in substantial token and inference costs (Jiang et al., 2023; 2024b). However, not every past interaction needs to remain in its original form throughout the entire task. Some interactions continue to determine future actions, while others have already served their purpose and their essential information has been absorbed by later observations or conclusions (Wu et al., 2024; Maharana et al., 2024). The central challenge is therefore not simply how to compress a long prompt, but rather which past interactions have become obsolete at the current agent state and can now be safely compressed during execution.

This decision is inherently state dependent. The same historical interaction may have very different retention requirements at different stages of execution. For example, an error trace can be indispensable while the agent is diagnosing a failure, but its complete details may no longer be necessary once the cause has been identified and the relevant conclusion has been recorded. In contrast, an early task constraint may remain important until the end of the trajectory. Compression opportunities are therefore distributed nonuniformly throughout agent history and cannot be determined reliably from age, position, or a fixed context window alone (Kang et al., 2025b; Hu et al., 2026). At each current state, the agent must reconsider which earlier interactions still require their complete details and which have already become replaceable.

This problem is also strongly asymmetric. In our supervision data, most historical interaction and state pairs remain KEEP, while only a small fraction are labeled READY. Keeping an interaction that could already be compressed mainly incurs additional context cost. Compressing an interaction that is still needed, however, may permanently remove information required by subsequent actions (Pan et al., 2024; Jiang et al., 2024b). This distinction is particularly important in our setting because removed raw interactions are not recovered through external retrieval. We therefore construct compression

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.

supervision with a two stage annotation procedure and train the router with an imbalance aware objective, favoring precise identification of truly compressible history over aggressive compression coverage under highly imbalanced supervision settings.

Motivated by these observations, we introduce **State Conditioned Compression (StateComp)**, a framework that determines when historical interactions can be safely compressed according to the current agent state during long horizon execution. The key idea is to use the current agent state to repeatedly reassess previous interactions and to separate the prediction that some history has become compressible from the decision to execute an actual compression operation. The main contributions of this work are as follows:

- **State dependent compression supervision.** We formulate history compression as a joint decision over a past interaction and the current agent state, and construct KEEP and READY supervision through a two stage annotation procedure that identifies supported compression boundaries.
- **Dynamic compression routing from hidden states.** We train an imbalance aware router on hidden representations from a frozen language model to predict whether each historical interaction can be safely compressed during online inference. A bounded state representation further reduces representation extraction cost.
- **Stateful online compression execution.** We group adjacent READY interactions into continuous spans and summarize only sufficiently large candidates. Committed summaries directly replace the corresponding raw history and become part of the effective context for subsequent agent actions and routing decisions throughout later execution.

We evaluate StateComp on WorkBuddyBench (Team et al., 2026). Across the full set of 260 tasks, total agent and summarization tokens decrease from 698.17M to 333.24M, a reduction of 52.27%, while mean reward changes from 0.6987 to 0.7026. Additional experiments on Eval40, representation efficiency, and multiple acting models further examine the quality and context cost tradeoff of StateComp across different agent settings.

2 Related Work

2.1 Prompt Compression

LLMLingua-2 formulates extractive prompt compression as token classification (Pan et al., 2024). This provides a way to shorten an input while retaining selected information (Jiang et al., 2023; 2024b; Li et al., 2023; Chevalier et al., 2023; Mu et al., 2023). In an agent setting, prompt compression and compression timing address different parts of history management (Ge et al., 2023; Xu et al., 2024; Cheng et al., 2024; Zhang et al., 2025). A representation of what should remain in a shortened input does not, by itself, specify the point in task execution at which the original details can be replaced. StateComp focuses on this temporal decision at the level of complete interactions. Its router selects the history that is ready for compression, while a separate summarizer constructs the content that remains. The selection target is defined jointly by a historical interaction and the current state rather than by the interaction alone.

2.2 Context Management for Long Horizon Agents

SelfCompact gives agents a compaction tool and guidance on when to use it, while ACON studies context compression for long horizon execution (Li et al., 2026b; Kang et al., 2025b). Self-GC, LRE, CoMem, and SAM explore complementary approaches to context management and memory (Hao et al., 2026; Jahan Lia & Mazumder, 2026; Zhang et al., 2026; Hu et al., 2026; Packer et al., 2023; Park et al., 2023; Zhong et al., 2024; Chhikara et al., 2025; Kang et al., 2025a; Xu et al., 2026b; Gutiérrez et al., 2024). These works place context management within the continuing execution of an agent rather than treating history only as a fixed input. StateComp uses a separate router trained on supported compression boundaries. Its prediction and execution decisions are distinct: a positive router prediction can contribute to a candidate span without immediately triggering summarization. Once a summary is committed, later actions and routing use the updated effective history. This makes the timing of replacement, the size of the selected region, and the information retained by the summary separate parts of the compression procedure.

2.3 Adaptive Selection of Historical Information

SWE-Pruner selects useful context for coding agents, and PACE adapts historical context using next step relevance (Wang et al., 2026; Wei et al., 2026). Sculptor equips agents with active context management tools, while ACM also studies context management over long horizon tasks (Li et al., 2026a;c). These approaches are related to selecting information for ongoing decisions. The target in StateComp is specifically whether an interaction still needs to remain in its original form at the current checkpoint. The annotation checks evidence already available in the prefix and uses future dependencies only to reject unsafe candidates. Selected raw spans are replaced by summaries that remain in the effective history. The online controller does not subsequently retrieve the removed originals, so both the decision to compress and the content of each replacement matter to later execution. This distinction separates replaceability from conventional relevance selection: an interaction may remain related to the current task while no longer requiring its original detailed representation during subsequent execution.

3 Supervision for State Dependent Compression

A past interaction does not become unnecessary simply because it is old. For example, an error trace may determine the next debugging action, but its full details may become replaceable once the failure has been diagnosed and the relevant conclusion has been recorded. We define compression with respect to the information available at the current agent state.

3.1 Compression Depends on the Current State

We use a complete interaction as the basic unit, $S_i = (R_i, A_i, O_i)$, containing reasoning, the assistant action, and its observation. Compression respects complete interaction boundaries so that tool calls and their responses remain paired. Before interaction k, the completed history is $P_k = (S_1, \ldots, S_{k-1})$. For $i < k$, the label $y_{i,k} = 0$ means that S_i should remain in full, whereas $y_{i,k} = 1$ means that its original details can be replaced by an accurate summary at that state. Importantly, this label is not an intrinsic property of S_i alone: the same interaction may transition from KEEP to READY as later observations resolve uncertainty, establish conclusions, or make its original details redundant.

The label is a property of the interaction and the current state together. Two interactions at similar positions can have different retention needs. An early task constraint may remain necessary throughout execution, while an intermediate search result may be superseded by a confirmed finding. The annotation therefore checks whether necessary information would remain available after removing the original details, rather than using age as a proxy for relevance.

3.2 The Earliest Safe Compression Point

For each S_i, we identify the earliest checkpoint t_i^* from which its full details are judged unnecessary along the annotated continuation. The resulting labels are

$$y_{i,k} = \begin{cases} 0, & k < t_i^*, \\ 1, & k \geq t_i^*. \end{cases} \tag{1}$$

If no supported point is found, we set $t_i^* = \infty$ and retain the interaction throughout the trajectory. This boundary summarizes the annotation along the observed trajectory, with compression decisions in other continuations requiring assessment of their corresponding states. Using the earliest supported boundary also avoids treating eventual compressibility as evidence that an interaction could have been removed earlier in the trajectory.

This formulation treats compression as a state dependent transition rather than a static importance judgment. An interaction becomes compressible only when the current prefix provides sufficient evidence to replace its raw form.

A positive decision requires that information still needed later is already available elsewhere in the retained history, or that the original detail is no longer needed. The final dataset contains 244,526 interaction and state pairs, of which 15,633 are positive. In total, 1,832 historical interactions receive a supported compression point. Complete statistics appear in Appendix A.

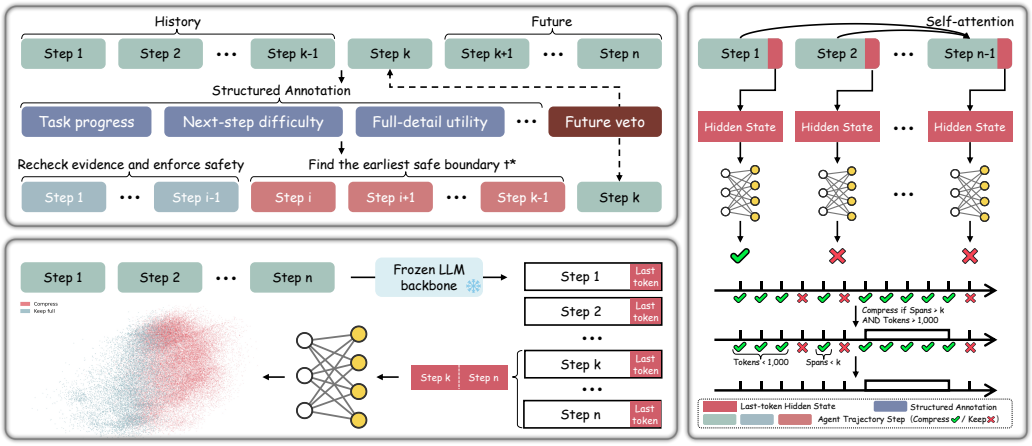

Figure 1: **Overview of StateComp.** Structured annotations and a second boundary review produce supervision. A router learns from hidden states of a frozen representation model. Online predictions are combined with span length and token gates before summaries replace raw interactions. Historical and current state features are used jointly, as described in Section 4.1.

3.3 Two Stage Annotation with Supporting Evidence

Direct annotation from a complete trajectory can confuse eventual resolution with information already available at an earlier checkpoint. We therefore require positive evidence to come from P_k. Future interactions may reveal that deleting S_i would remove a detail needed later, but future success or a later summary cannot establish that an earlier deletion was safe.

Stage 1 independently annotates each checkpoint. It records task progress, observation complexity, expected next subtask requirements, the remaining utility of the target interaction, and any supporting evidence already present in the prefix. Each record includes a reason, evidence locations, an initial KEEP or READY judgment, and any future dependency that vetoes a positive judgment. Decisions at one checkpoint are not inherited by another.

Stage 2 uses these Stage 1 records as its input. For a fixed historical interaction, it reviews the initial decisions across checkpoints together with the original evidence, checks proposed compression points again, and finds the earliest supported boundary. This separates local evidence collection from final boundary determination. It also exposes inconsistent initial judgments instead of propagating the first READY label through all later states. Unresolved cases remain KEEP. This decomposition bases positive labels on explicit state local evidence collected at individual checkpoints, separating evidence collection from boundary determination over the annotated trajectory.

The procedure treats the two errors differently. Retaining a replaceable interaction mainly increases context cost. Removing information that remains necessary can harm subsequent actions, especially when raw history cannot be retrieved later. We therefore require stronger evidence for a positive label and favor precision when selecting a router operating point.

4 StateComp Methodology

StateComp uses the current state to reconsider which historical interactions must remain in full. It makes predictions at interaction granularity, summarizes eligible continuous spans, and updates the history used by later actions. A central design choice is to separate readiness prediction from compression execution: identifying a replaceable interaction does not by itself imply that an immediate rewrite is worthwhile. This separation allows the controller to accumulate local compression opportunities before committing a larger rewrite, reducing unnecessary summary calls and avoiding fragmented updates to the active history. Figure 1 connects labeling, router training, and online compression throughout the subsequent task execution.

4.1 Hidden-State Representations and Router Training

The online agent cannot access future interactions or the annotated boundary. We instead read representations from a frozen language model. In the full context reference, one forward pass over P_k provides a hidden vector h_i at the final token of each historical interaction and a current state vector q_k at the final prompt position. The router uses the historical vector together with the current state to predict $y_{i,k}$ at each checkpoint.

This formulation does not require an earlier token to attend to later tokens. With a causal attention mask, appending new interactions does not by itself change an earlier hidden vector. State dependence is supplied by q_k and its combination with the target interaction, or by a separately constructed input that contains both the target and current context. Thus, the same historical interaction can receive different router scores as the current state changes. Section 4 describes the bounded implementation used to reduce representation extraction cost.

Full context extraction is a useful reference but becomes expensive as history grows (Kwon et al., 2023; Xiao et al., 2024b; Zhang et al., 2023; Jiang et al., 2024a; Xiao et al., 2024a). The recorded trajectories include inputs of approximately 128K tokens and a largest input of 203,675 tokens. Repeatedly processing these histories can make the representation model itself a substantial source of overhead (Liu et al., 2023; Li et al., 2024; Cai et al., 2024; Tang et al., 2024).

The practical construction uses a separate bounded view for each historical interaction under evaluation. It combines the target interaction with the current state and a limited amount of effective history. The frozen model reads this view, and the final prompt position supplies a compact readout for the decision. Every surviving raw interaction remains eligible for assessment. Bounded input construction changes the context used to compute features while preserving the full set of historical candidates available to the router. Historical interactions can therefore be reconsidered regardless of their position within the surviving effective history at the current checkpoint.

The representation budget is at most $5{,}120$ tokens per view. Recent4 is the reference recent context construction in the reported bounded ablation. We distinguish its context window K from the compression span threshold κ below. With N historical targets, candidate specific construction involves N short input sequences, which may be batched. The complete checkpoint cost still depends on N, despite the token bound on each view.

For each remaining historical interaction, the router combines the target representation, current state, and available historical features to produce a score $p_{i,k}$. Features come only from the effective prefix. Aggregating historical vectors within that prefix constructs prediction features solely from the information retained in the effective history. The router is trained on the labels from Section 3 with the representation model frozen throughout all training. Training variants and the grouped selection protocol are documented in Appendix B.

4.2 Interaction Routing and Span Execution Gates

A score at or above τ selects an interaction as a compression candidate. The controller then merges adjacent selected interactions into continuous spans. A KEEP interaction breaks a span; it is not included merely to connect two selected regions. A positive router decision does not immediately trigger a summary call. For a candidate span $B = [s, e]$, the execution rule is

$$\min_{i \in B} p_{i,k} \geq \tau, \qquad |B| > \kappa, \qquad \mathrm{Tokens}(B) \geq B_{\min}, \tag{2}$$

with $\tau = 0.60$, $\kappa = 3$, and $B_{\min} = 1{,}000$. Thus, an eligible span contains at least four complete interactions and at least 1,000 source tokens. The token condition specifies the minimum source length required to trigger a candidate summarization call.

These gates address the cost of executing compression. The score threshold controls prediction confidence, while the span-length and source-token conditions control whether a predicted opportunity is large enough to justify an actual summarization operation. Summarizing tiny fragments can spend more tokens on the auxiliary call than it saves in the active history. Combining adjacent interactions also gives the summarizer a more coherent source and avoids a succession of fragmented replacements. The execution gates therefore convert local readiness predictions into larger operational decisions, separating semantic replaceability from the economic cost of performing a rewrite. Importantly,

rewriting earlier history can invalidate KV reuse from the first changed position onward. Prefix caching depends on an unchanged prefix. Deferring small rewrites is designed to limit repeated cache invalidation and reconstruction. KV reuse remains restricted to the unchanged prefix, while effects on cache hit rate require direct measurement under the actual serving configuration.

Each eligible span produces one summary containing facts, constraints, locations, confirmed conclusions, failure causes, and unresolved items that remain useful. The router determines which original details may leave the active context; the summarizer determines what the replacement retains. The controller commits a replacement only when it preserves interaction structure and actually shortens the effective history. Otherwise, it keeps the original span. Appendix E specifies this procedure.

4.3 Updating the Effective Agent History

After a summary replaces a raw span, the resulting effective history becomes the context used for the next agent action. Later state features and router predictions are computed from this updated history rather than from an untouched copy of the original trajectory. Remaining interactions are therefore reconsidered at every checkpoint under the newly formed agent state. Compression is therefore a closed loop process: each committed replacement changes the context on which subsequent routing decisions are made throughout later execution.

Figure 2 illustrates this process on one trajectory. Early checkpoints retain most of the original history. As execution proceeds, additional continuous regions become eligible for compression and are replaced, while interactions that still require their original details remain in the active history.

This stateful update is important because a compression action changes the context available to all subsequent decisions. A previously selected interaction is therefore not treated as a permanent authorization for future compression. Instead, the router scores the remaining history again from the current effective state.

The system retains the original trajectory separately for auditing, but this canonical record is not used as an external retrieval store by the online controller. The effective history contains only retained raw interactions and committed summaries. If routing or final validation fails, the controller returns the previous valid effective history. A failed summarization proposal leaves its source span unchanged.

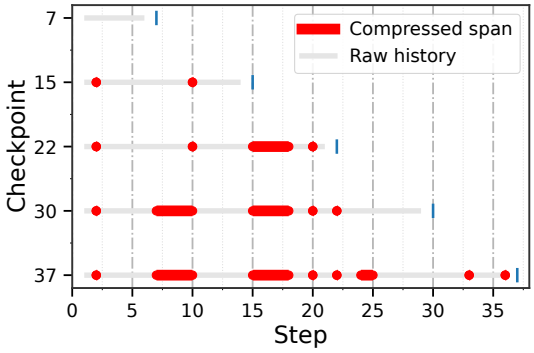

Figure 2: Compression dynamics across checkpoints in an example trajectory.

5 Experiments

We evaluate task reward and model token consumption on complete agent tasks. The full benchmark measures the overall effect, the fixed Eval40 comparison examines alternative context strategies, and cross model runs characterize how the quality and token tradeoff changes with the acting model.

5.1 Experimental Setup

We first examine how labeled compression opportunities evolve, then evaluate prediction on held out trajectories. The strict router study uses frozen Qwen2.5-7B representations and five grouped folds. Each rotation has 200 training, 50 validation, and 50 test trajectories. All pairs from one trajectory stay in the same split. Configuration and threshold selection use training and validation data rather than test labels in every fold.

WorkBuddyBench contains 260 tasks: 80 Code, 50 Office, 60 Security, and 70 Web (Team et al., 2026). The full comparison uses DeepSeek-V4-Flash (Xu et al., 2026a) on the same 260 tasks. Eval40 contains ten fixed tasks from each domain. All method and cross model comparisons on Eval40 use this same 40-task set. We report mean reward multiplied by 100 in tables, with unscaled

Table 1: **Efficiency and quality of bounded state representations.** Max. tokens reports the largest recorded input; runtime and memory are measured on the same seven paired states.

Method	Max. tokens	Wall clock (s)	Peak GPU (GiB)	Workspace (GiB)	Test F1
Full history	203,675	76.40	29.426	11.867	0.7000
Bounded	**5,120** (-97.49%)	**6.03** (12.67×)	**18.620** (-36.72%)	**1.062** (-91.05%)	**0.7818** (+0.08)

reward in the text. Total tokens include recorded agent and summary input and output tokens. Cached input and reasoning output are subsets of those totals and are not added again. Local representation cost is reported separately in Table 1.

The Full260 and Eval40 evaluations use the compression rule in Section 4.2, with $\tau = 0.60$, $\kappa = 3$, and $B_{\min} = 1{,}000$ tokens. The additional published baselines use the same Eval40 token accounting. Operating settings and the fixed Eval40 task identifiers are provided in Appendix G.

5.2 Hidden-State Prediction and Representation Efficiency

Figure 3 shows that compression opportunities emerge progressively during task execution. Across all four domains, only a small fraction of past interactions are labeled READY near the beginning of a trajectory, while the fraction generally increases as execution proceeds. This pattern indicates that historical interactions do not have a fixed retention requirement. Instead, their compression status changes as the agent accumulates new evidence and completes intermediate subtasks.

The trend motivates state conditioned selection rather than compression based only on age or a fixed context window. At the same time, the four domains follow different trajectories, suggesting that task progress alone is not a sufficient decision rule in practice. This variation further supports conditioning on the observed agent state rather than applying a globally shared compression schedule across tasks. The router therefore conditions each compression decision on the current agent state and the target historical interaction. Additional analyses of compression delay, trajectory variation, and trajectory length are reported in Appendix D.

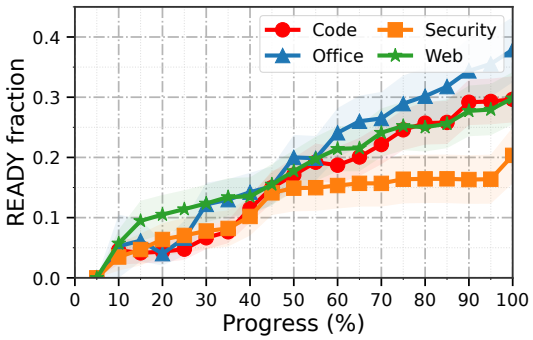

Figure 3: Compression opportunities over different trajectory stages.

We next test whether these state dependent compression decisions can be predicted from hidden representations. Under grouped cross trajectory evaluation, the evaluated configuration obtains AUROC 0.7096 and PR AUC 0.1736. The result indicates that frozen hidden states contain predictive information about whether the full details of a historical interaction should still be retained. This predictive signal motivates precision oriented routing with span and token gates, which determine when candidate interactions enter an online compression operation.

The prediction problem is intentionally asymmetric. Retaining an interaction that could already be compressed mainly increases context cost, whereas compressing an interaction too early can remove information needed by later actions. We therefore use a precision oriented operating policy. A highly selective configuration reaches 72.22% external precision at 0.11% recall. Full precision coverage curves, calibration results, and threshold diagnostics are provided in Appendix D.

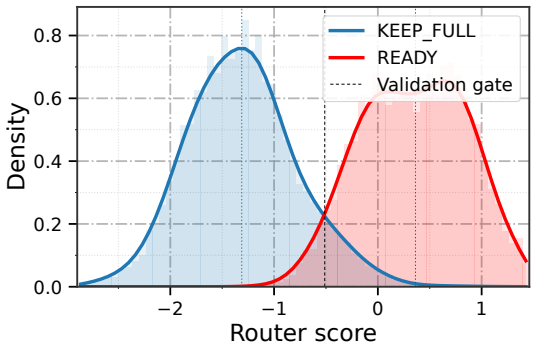

Figure 4: Router score distributions.

Figure 4 shows the learned score distributions for KEEP_FULL and READY examples. The

Table 2: **Full WorkBuddyBench comparison on 260 tasks.** Scores are mean reward multiplied by 100. Tokens include the main agent and summary calls. Colored changes show score differences relative to the baseline for all domains.

Method	Code	Office	Sec.	Web	Avg.	Tokens (M)
Baseline	76.99	81.84	**47.76**	72.14	69.87	698.17
StateComp	**77.12** ↑0.13	**82.77** ↑0.93	47.31 ↓0.45	**73.14** ↑1.00	**70.26** ↑0.39	**333.24**

score visualization characterizes the trained readout, complementing the grouped evaluation of predictive performance above.

Table 1 separates recorded context sizes from paired runtime measurements. On seven states for which both implementations complete, total extraction time falls from 76.40 to 6.03 seconds. Peak GPU memory decreases by 36.72% and forward workspace by 91.05%. The largest full inputs do not complete in this microbenchmark and are not used as timed paired samples. In the separate representation comparison, the best reported bounded configuration has F1 0.7818, compared with 0.7000 for the full context reference. Thus, the bounded configuration combines faster extraction with higher test F1 in this comparison, using an input construction that differs from the full context reference in its retained context. Detailed configurations and measurements are in Appendix C. Taken together, these results show that compression readiness is state varying and exhibits measurable predictability from compact representations, providing the basis for practical online routing.

5.3 Task Performance and Context Cost

Table 2 reports the complete comparison. Mean reward changes from 0.6987 to 0.7026, while total agent and summary tokens fall from 698.17M to 333.24M, a reduction of 52.27%. Average tokens per task therefore fall from approximately 2.69M to 1.28M. The result combines a large reduction in transmitted and generated tokens with a small increase in average task reward. The main empirical benefit in this setting is improved token efficiency with maintained task performance: the method removes substantial context while keeping aggregate task quality close to the uncompressed execution across the full benchmark.

Token reductions occur in all four domains, ranging from 38.89% in Code to 55.75% in Office. Code, Office, and Web have slightly higher reward, while Security changes from 0.4776 to 0.4731. These results show consistent token reductions across domains, with small domain dependent changes in task reward. Detailed token components are provided in Appendix F. The consistent token reductions across all four domains show that the efficiency gain extends across the different task categories covered by this benchmark.

On Eval40, StateComp obtains a mean reward of 0.7154 with 43.76M tokens, compared with 0.6754 and 75.50M for the baseline. This corresponds to 42.04% fewer tokens. The token total includes 0.077M summary tokens.

Sliding windows with $K = 5, 10, 20$ obtain rewards of 0.5639, 0.6212, and 0.6867. The largest window still uses 80.82M tokens. Periodic summarization with $n = 5$ reaches 0.7017 reward but uses 94.85M tokens, including 3.01M summary tokens. PACE uses 44.12M tokens with reward 0.5842. Within this comparison group, StateComp achieves the highest mean reward with 43.76M total tokens. This supports evaluating compression timing together with the cost of generating summaries, rather than reporting the length of a shortened prompt alone. The comparison also shows that total consumption depends on both context length and execution behavior, as changes in execution paths and repeated summary operations can offset prompt level savings.

Table 3 also reports additional published context management baselines evaluated on the same fixed Eval40 tasks for a broader comparative reference.

5.4 Cross-Model Evaluation

Table 4 applies the same compression procedure to four acting models on the same fixed Eval40 tasks. GLM-5.3-Flash (Zeng et al., 2026) improves from 0.6170 to 0.6492 reward while tokens fall from

Table 3: **Context management results on Eval40.** Scores are reward multiplied by 100. Small colored arrows show changes relative to the DeepSeek baseline. Bold and underlined values indicate the best and second best results.

Method	Code	Office	Sec.	Web	Avg.	Tokens (M)
Baseline	72.93	81.64	44.58	71.00	67.54	75.50
Sliding Window ($K = 5$)	66.81 ↓6.12	67.43 ↓14.21	32.32 ↓12.26	59.00 ↓12.00	56.39 ↓11.15	118.08
Sliding Window ($K = 10$)	74.37 ↑1.44	73.05 ↓8.59	40.07 ↓4.51	61.00 ↓10.00	62.12 ↓5.42	79.39
Sliding Window ($K = 20$)	<u>79.31</u> ↑6.38	**85.11** ↑3.47	40.28 ↓4.30	70.00 ↓1.00	68.67 ↑1.13	80.82
Periodic Summary ($n = 3$)	65.44 ↓7.49	67.66 ↓13.98	26.32 ↓18.26	**82.66** ↑11.66	60.52 ↓7.02	88.57
Periodic Summary ($n = 5$)	**83.60** ↑10.67	75.74 ↓5.90	40.03 ↓4.55	<u>81.32</u> ↑10.32	<u>70.17</u> ↑2.63	94.85
LLMLingua-2 (Pan et al., 2024)	64.64 ↓8.29	73.31 ↓8.33	40.10 ↓4.48	70.00 ↓1.00	62.01 ↓5.53	96.18
PACE (Wei et al., 2026)	70.38 ↓2.55	55.75 ↓25.89	43.53 ↓1.05	64.02 ↓6.98	58.42 ↓9.12	44.12
SelfCompact (Li et al., 2026b)	76.83 ↑3.90	79.62 ↓2.02	36.20 ↓8.38	71.00	65.91 ↓1.63	62.70
ACON-Core (Kang et al., 2025b)	71.79 ↓1.14	68.55 ↓13.09	<u>47.95</u> ↑3.37	65.00 ↓6.00	63.32 ↓4.22	56.13
Self-GC (Hao et al., 2026)	63.45 ↓9.48	71.67 ↓9.97	43.71 ↓0.87	71.00	62.46 ↓5.08	66.21
LRE (Jahan Lia & Mazumder, 2026)	62.62 ↓10.31	63.44 ↓18.20	27.14 ↓17.44	68.00 ↓3.00	55.30 ↓12.24	91.64
CoMem (Zhang et al., 2026)	56.43 ↓16.50	59.86 ↓21.78	2.50 ↓42.08	58.00 ↓13.00	44.20 ↓23.34	**30.64**
SAM (Hu et al., 2026)	67.62 ↓5.31	81.63 ↓0.01	47.24 ↑2.66	67.00 ↓4.00	65.87 ↓1.67	53.85
SWE-Pruner (Wang et al., 2026)	63.45 ↓9.48	72.60 ↓9.04	40.87 ↓3.71	65.00 ↓6.00	60.48 ↓7.06	60.07
Sculptor (Li et al., 2026a)	51.19 ↓21.74	81.62 ↓0.02	35.25 ↓9.33	69.00 ↓2.00	59.26 ↓8.28	51.75
ACM (Li et al., 2026c)	38.69 ↓34.24	55.93 ↓25.71	6.20 ↓38.38	34.00 ↓37.00	33.70 ↓33.84	94.56
StateComp (Ours)	76.03 ↑3.10	<u>84.12</u> ↑2.48	**51.01** ↑6.43	75.00 ↑4.00	**71.54** ↑4.00	<u>43.76</u>

Table 4: **Cross model evaluation on Eval40.** Scores are reward multiplied by 100. Colored arrows show changes relative to the same backbone without StateComp. Tokens report the total model usage in millions. Each domain contains ten tasks, so Avg. is the mean of the four domain scores.

Backbone	Method	Code	Office	Sec.	Web	Avg.	Tokens (M)
Hunyuan	Base	55.30	79.24	**60.24**	46.00	60.19	41.00
	StateComp	**64.31** ↑9.01	**82.20** ↑2.96	60.14 ↓0.10	**49.00** ↑3.00	**63.91** ↑3.72	**12.83**
Qwen3.8-Flash	Base	75.25	83.42	45.28	86.00	72.49	49.75
	StateComp	**93.46** ↑18.21	**88.71** ↑5.29	41.20 ↓4.08	**88.00** ↑2.00	**77.84** ↑5.35	38.23
GLM-5.3-Flash	Base	79.49	66.13	34.67	66.51	61.70	47.58
	StateComp	66.15 ↓13.34	69.90 ↑3.77	38.64 ↑3.97	85.00 ↑18.49	64.92 ↑3.22	38.49
MiMo-V2.5	Base	53.76	77.05	22.04	55.00	51.96	53.55
	StateComp	57.65 ↑3.89	71.64 ↓5.41	22.77 ↑0.73	58.00 ↑3.00	52.52 ↑0.56	18.91

47.58M to 38.49M. MiMo-V2.5 changes from 0.5196 to 0.5252 with tokens falling from 53.55M to 18.91M. Both models achieve higher mean reward with lower token consumption.

Qwen3.8-Flash improves reward from 0.7249 to 0.7784, while tokens fall from 49.75M to 38.23M. Hunyuan reduces tokens from 41.00M to 12.83M while reward increases from 0.6019 to 0.6391. Across all four acting models, StateComp reduces token usage and increases mean reward, with the magnitude of these gains and their distribution across domains varying by backbone. These results demonstrate the transfer of the same compression procedure across different acting models. Total token usage summarizes complete-task consumption across requests; distinguishing the contributions of execution length, retries, and request size would require a separate breakdown of the underlying request records. Domain scores are given in Appendix F.

6 Conclusion

We introduced StateComp, which predicts whether past interactions still need their full details at the current agent state. Two stage annotation supplies supported compression boundaries, hidden states provide prediction features, and span and token gates determine when selected history is summarized. Committed summaries become part of the effective history without restoring removed raw interactions. The full benchmark comparison reduces agent and summary tokens by approximately half while maintaining the average task reward. These findings support separating the prediction of compression opportunities from the decision to rewrite a sufficiently large span. The experiments quantify complete task token usage across domains and acting models in diverse settings, including the overhead of auxiliary summary generation.

AI USE STATEMENT

Generative AI tools were used for language editing and literature retrieval and discovery, including identifying potentially relevant papers, search keywords, and related work, as well as surveying the current state of research on relevant topics. An LLM was also used in the annotation procedure to assist in constructing the compression supervision described in the paper and appendix. The annotation procedure and resulting labels were reviewed by the authors. The final selection, interpretation, and presentation of the literature and experimental results were determined by the authors. All AI assisted content was reviewed by the authors, who take full responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

Appendices A to E describe the two stage annotation procedure, grouped training and evaluation splits, representation construction, router objectives, theoretical analysis, and online compression algorithm. Appendices F and G provide the fixed benchmark protocols, supplementary results, operating settings, and the fixed Eval40 task identifiers. Token accounting conventions and the compression configuration are specified alongside the corresponding experiments.

REFERENCES

Yushi Bai, Shangqing Tu, Jiajie Zhang, Hao Peng, Xiaozhi Wang, Xin Lv, Shulin Cao, Jiazheng Xu, Lei Hou, Yuxiao Dong, et al. Longbench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3639–3664, 2025.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Yucheng Li, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Junjie Hu, et al. Pyramidkv: Dynamic kv cache compression based on pyramidal information funneling. *arXiv preprint arXiv:2406.02069*, 2024.

Xin Cheng, Xun Wang, Xingxing Zhang, Tao Ge, Si-Qing Chen, Furu Wei, Huishuai Zhang, and Dongyan Zhao. xrag: Extreme context compression for retrieval-augmented generation with one token. *Advances in Neural Information Processing Systems*, 37:109487–109516, 2024.

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. Adapting language models to compress contexts. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 3829–3846, 2023.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready ai agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*, 2025.

Tao Ge, Jing Hu, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. In-context autoencoder for context compression in a large language model. *arXiv preprint arXiv:2307.06945*, 2023.

Bernal J Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. Hipporag: Neurobiologically inspired long-term memory for large language models. *Advances in neural information processing systems*, 37:59532–59569, 2024.

Xubin Hao, Hongjin Meng, Xin Yin, Jiawei Zhu, and Chenpeng Cao. Self-gc: Self-governing context for long-horizon llm agents. *arXiv preprint arXiv:2607.00692*, 2026.

Yuyang Hu, Hongjin Qian, Shuting Wang, Jiongnan Liu, Ziliang Zhao, Jiejun Tan, Zheng Liu, and Zhicheng Dou. Sam: State-adaptive memory for long-horizon reasoning agent. *arXiv preprint arXiv:2605.24468*, 2026.

Nusrat Jahan Lia and Aritra Mazumder. Learning what not to forget: Long-horizon agent memory from a few kilobytes of learning. *arXiv e-prints*, pp. arXiv–2606, 2026.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. Llmlingua: Compressing prompts for accelerated inference of large language models. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 13358–13376, 2023.

Huiqiang Jiang, Yucheng Li, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Zhenhua Han, Amir H Abdi, Dongsheng Li, Chin-Yew Lin, et al. Minference 1.0: Accelerating pre-filling for long-context llms via dynamic sparse attention. *Advances in Neural Information Processing Systems*, 37:52481–52515, 2024a.

Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. Longllmlingua: Accelerating and enhancing llms in long context scenarios via prompt compression. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1658–1677, 2024b.

Jiazheng Kang, Mingming Ji, Zhe Zhao, and Ting Bai. Memory os of ai agent. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 25972–25981, 2025a.

Minki Kang, Wei-Ning Chen, Dongge Han, Huseyin A Inan, Lukas Wutschitz, Yanzhi Chen, Robert Sim, and Saravan Rajmohan. Acon: Optimizing context compression for long-horizon llm agents. *arXiv preprint arXiv:2510.00615*, 2025b.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pp. 611–626, 2023.

Mo Li, LH Xu, Qitai Tan, Long Ma, Hongyong Song, Ting Cao, and Yunxin Liu. Sculptor: Empowering llms with cognitive agency via active context management. In *International Conference on Learning Representations*, volume 2026, pp. 153411–153440, 2026a.

Tianjian Li, Jingyu Zhang, William Jurayj, Xi Wang, Chuanyang Jin, Mehrdad Farajtabar, Eric Nalisnick, and Daniel Khashabi. Self-compacting language model agents. *arXiv preprint arXiv:2606.23525*, 2026b.

Xiaochuan Li, Ryan Ming, Meng Chu, Shuai Shao, Rong Jin, and Chenyan Xiong. Acm: Agentic context management for long horizon tasks. *arXiv preprint arXiv:2607.23809*, 2026c.

Yucheng Li, Bo Dong, Frank Guerin, and Chenghua Lin. Compressing context to enhance inference efficiency of large language models. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 6342–6353, 2023.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. Snapkv: Llm knows what you are looking for before generation. *Advances in Neural Information Processing Systems*, 37:22947–22970, 2024.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the association for computational linguistics*, 12:157–173, 2024.

Zichang Liu, Aditya Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhuo Xu, Anastasios Kyrillidis, and Anshumali Shrivastava. Scissorhands: Exploiting the persistence of importance hypothesis for llm kv cache compression at test time. *Advances in Neural Information Processing Systems*, 36:52342–52364, 2023.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of llm agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13851–13870, 2024.

Jesse Mu, Xiang Li, and Noah Goodman. Learning to compress prompts with gist tokens. *Advances in Neural Information Processing Systems*, 36:19327–19352, 2023.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G Patil, Ion Stoica, and Joseph E Gonzalez. Memgpt: Towards llms as operating systems. *arXiv preprint arXiv:2310.08560*, 2023.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Rühle, Yuqing Yang, Chin-Yew Lin, et al. Llmlingua-2: Data distillation for efficient and faithful task-agnostic prompt compression. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 963–981, 2024.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pp. 1–22, 2023.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. Quest: Query-aware sparsity for efficient long-context llm inference. *arXiv preprint arXiv:2406.10774*, 2024.

Tencent WorkBuddy Bench Team, Siqi Cai, Shaopeng Chen, Xiang Fei, Yong Mao, Zihan Xu, Zhiheng Lyu, Zhijian Shao, Yuchen Shi, Shuwen Zhang, et al. Tencent workbuddy bench: A multi-domain coding-agent benchmark with contamination-resistant task construction. *arXiv preprint arXiv:2607.20911*, 2026.

Yuhang Wang, Yuling Shi, Mo Yang, Rongrui Zhang, Shilin He, Heng Lian, Yuting Chen, Siyu Ye, Kai Cai, and Xiaodong Gu. Swe-pruner: Self-adaptive context pruning for coding agents. *arXiv preprint arXiv:2601.16746*, 2026.

Lei Wei, Xiao Peng, Guannan Zhang, Chenhao Jiang, Hongyu Li, Lanbo Lin, Yuanwu Xu, Jiayao Liu, Kesu Wang, Bin Wang, et al. Pace: Predictive adaptive context extraction for long-horizon llm agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 27184–27199, 2026.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. Longmemeval: Benchmarking chat assistants on long-term interactive memory. *arXiv preprint arXiv:2410.10813*, 2024.

Chaojun Xiao, Pengle Zhang, Xu Han, Guangxuan Xiao, Yankai Lin, Zhengyan Zhang, Zhiyuan Liu, Song Han, and Maosong Sun. Infllm: Unveiling the intrinsic capacity of llms for understanding extremely long sequences with training-free memory. *arXiv preprint arXiv:2402.04617*, 3(7), 2024a.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations*, volume 2024, pp. 21875–21895, 2024b.

Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, et al. Deepseek-v4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026a.

Fangyuan Xu, Weijia Shi, and Eunsol Choi. Recomp: Improving retrieval-augmented lms with context compression and selective augmentation. In *International Conference on Learning Representations*, volume 2024, pp. 43478–43502, 2024.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for llm agents. *Advances in Neural Information Processing Systems*, 38:17577–17604, 2026b.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, et al. Glm-5: from vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*, 2026.

Peitian Zhang, Zheng Liu, Shitao Xiao, Ninglu Shao, Qiwei Ye, and Zhicheng Dou. Long context compression with activation beacon. In *International Conference on Learning Representations*, volume 2025, pp. 101932–101948, 2025.

Yuwei Zhang, Chengyu Dong, Shuowei Jin, Changlong Yu, Hejie Cui, Hongye Jin, Xinyang Zhang, Hamed Bonab, Colin Lockard, Jianshu Chen, et al. Comem: Context management with a decoupled long-context model. *arXiv preprint arXiv:2605.30842*, 2026.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in neural information processing systems*, 36:34661–34710, 2023.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pp. 19724–19731, 2024.

A COMPRESSION TARGETS AND SUPERVISION

This appendix provides the supervision, training details, mathematical analysis, and supplementary experiments for STATECOMP. The analysis characterizes the prediction and execution rules, while supplementary experiments evaluate the resulting empirical compression quality.

A.1 INTERACTIONS AND EFFECTIVE HISTORY

An interaction $S_i = (R_i, A_i, O_i)$ contains reasoning, an assistant action, and its observation. Tool calls and their responses remain within complete interaction boundaries. Before step k, let $P_k = (S_1, \ldots, S_{k-1})$ denote the original prefix and H_k the effective history after previously committed replacements. Let $\mathcal{I}_k$ index the surviving raw interactions. The agent and router use H_k; stored original trajectories are not available as an online retrieval channel.

The target is defined jointly by a historical interaction and the current state:

$$y_{i,k} = \begin{cases} 0 \quad (\text{KEEP}), & \text{retain the original details,} \\ 1 \quad (\text{READY}), & \text{replace the details by an accurate summary.} \end{cases} \tag{A.1}$$

READY permits replacing original details while requiring information still needed by the task to remain available in the summary or elsewhere in the effective history. The online router predicts

$$p_{i,k} = p_\theta(S_i, H_k), \qquad \widehat{y}_{i,k} = \mathbf{1}\{p_{i,k} \geq \tau\}, \qquad i \in \mathcal{I}_k. \tag{A.2}$$

Selected interactions may occupy several disconnected regions; a recent window is not imposed on candidate eligibility.

Table A1: **Notation for supervision and online compression.**

Symbol	Meaning
S_i	Complete historical interaction
P_k	Uncompressed reference prefix before interaction k
$H_k, \mathcal{I}_k$	Effective history and identifiers of surviving raw interactions
$y_{i,k}$	Reference label: KEEP $= 0$, READY $= 1$
t_i^*	Earliest supported compression checkpoint
F_ϕ, g_θ	Frozen representation model and trained router
$h_i, q_k, r_{i,k}$	Historical, current, and aggregated historical features
$z_{i,k}, p_{i,k}$	Router logit and sigmoid score
τ	Router selection threshold
$K_{\text{ctx}}, K_{\text{sw}}, K_{\text{hist}}$	Recent context size, sliding window size, and same prefix feature selection count
$B, \kappa, B_{\min}$	Continuous source span, interaction threshold, and source token threshold
$M, \text{Tok}(H)$	Per view representation budget and serialized history length

A.2 TWO-STAGE ANNOTATION

Stage 1 independently annotates each checkpoint and produces a structured judgment for every historical interaction at that checkpoint. Each record includes task progress, observation complexity, expected requirements of the next subtask, the remaining utility of the interaction, a KEEP/READY judgment, and the locations of supporting evidence. Positive evidence must already occur in P_k. Future interactions can reveal a missing dependency and veto a proposed replacement, but cannot justify an earlier replacement using information that was not yet available.

Writing $e_{i,k}$ for supporting prefix evidence and $v_{i,k}$ for a future dependency veto, a positive local judgment $b_{i,k}$ obeys

$$b_{i,k} = 1 \implies e_{i,k} = 1 \text{ and } v_{i,k} = 0. \tag{A.3}$$

Stage 2 reviews these records across checkpoints for the same interaction, checks proposed boundaries against their supporting evidence, and selects the earliest supported compression point t_i^*. Unresolved cases remain KEEP. The final labels are

$$y_{i,k} = \mathbf{1}\{k \geq t_i^*\}, \qquad t_i^* \in \{i+1, \ldots, T\} \cup \{\infty\}. \tag{A.4}$$

The value ∞ denotes an interaction with no supported boundary. Monotonicity describes the boundary labels for the annotated continuation. Online predictions are recomputed after the effective history changes, rather than inherited from earlier checkpoints, to reflect the current execution state.

A.3 Data and Grouped Evaluation

The supervision collection contains 300 trajectories and 244,526 interaction and state pairs. READY accounts for 15,633 pairs, giving

$$\pi = \frac{15{,}633}{244{,}526} \approx 0.06393. \tag{A.5}$$

An always KEEP classifier consequently reaches approximately 93.61% accuracy without identifying a single compression opportunity. We use ranking and precision and recall metrics rather than accuracy alone.

Table A2: **Compression supervision and evaluation partitions.**

Quantity	Value
Trajectories	300
Recorded states	7,931
Checkpoint groups	7,631
Interaction and state pairs	244,526
KEEP pairs	228,893
READY pairs	15,633
Supported boundaries	1,832
Interactions without a supported boundary	6,099
Training / validation / test trajectories per rotation	200 / 50 / 50
Grouped folds	5

All pairs from the same trajectory remain in one partition. Model configuration, training checkpoint, class weighting, and prediction threshold are selected using training and validation data. The five rotations evaluate generalization across trajectories rather than across randomly separated pairs from the same trajectory.

B State Representations and Router Learning

B.1 Historical and Current State Features

The grouped router study uses a frozen Qwen2.5-7B representation model. Let $X_k = \mathrm{Serialize}(P_k)$, and let ℓ_i be the last token of S_i. A full context pass of the frozen representation model F_ϕ provides

$$h_i = F_\phi(X_k)_{\ell_i}, \qquad q_k = F_\phi(X_k)_{|X_k|}. \tag{B.1}$$

The first vector describes the historical interaction; the second describes the current state. A notation covering the concatenation based variants is

$$r_{i,k} = \mathrm{Agg}\{h_j : j \in \mathcal{J}_{i,k}\}, \qquad x_{i,k} = [h_i; q_k; r_{i,k}], \qquad p_{i,k} = \sigma(g_\theta(x_{i,k})). \tag{B.2}$$

Here $\mathcal{J}_{i,k}$ lies within the available prefix. The experiments compare features with and without recurrent aggregation and same prefix historical selection. These operations construct prediction features using information retained within the currently available prefix.

Proposition B.1 (Causal prefix invariance). *Consider a deterministic causal Transformer with fixed parameters, unchanged prefix tokens and positional assignments, and no transformation of the prefix that depends on the length of an appended suffix. In exact arithmetic, appending tokens leaves every existing prefix hidden state unchanged.*

Proof. Let $H_j^{(\ell)}$ be the representation at prefix position j and layer ℓ. The input states $H_j^{(0)}$ are identical. For one attention head,

$$A_j^{(\ell)} = \sum_{u \le j} \alpha_{ju}^{(\ell)} W_V^{(\ell)} H_u^{(\ell)}, \qquad \alpha_{ju}^{(\ell)} = \frac{\exp(a_{ju}^{(\ell)})}{\sum_{v \le j} \exp(a_{jv}^{(\ell)})}. \tag{B.3}$$

Assume that the states at all prefix positions are unchanged at layer ℓ. Each query, permitted key, value, and positional contribution in Equation (B.3) is then unchanged. The normalization contains no suffix positions, so every attention output is unchanged. Residual connections, tokenwise normalization, and feed forward layers preserve the equality. Induction over layers proves the result. □

Thus, a historical vector cannot become state dependent merely because new tokens are appended. The router obtains state dependence from q_k or from a freshly constructed input containing both the target interaction and current context. The proposition also characterizes reusable prefix states after an online rewrite.

B.2 Weighted Classification

With READY encoded as one, the router objective is

$$\mathcal{L}(\theta) = -\frac{1}{N} \sum_{n=1}^{N} \left[w_1 y_n \log p_n + w_0 (1 - y_n) \log(1 - p_n) \right]. \tag{B.4}$$

The weighted BCE experiments set $w_0 = 1$ and vary w_1. The representation model remains frozen. For $p = \sigma(z)$ and a binary label y,

$$\frac{\partial \ell}{\partial z} = \big(w_1 y + w_0 (1 - y)\big)(p - y), \tag{B.5}$$

$$\frac{\partial^2 \ell}{\partial z^2} = \big(w_1 y + w_0 (1 - y)\big) p (1 - p). \tag{B.6}$$

These derivatives characterize convexity with respect to the logit, while convexity in nonlinear router parameters requires a separate analysis.

Proposition B.2 (Weighted score and underlying posterior). *Let $\eta(x) = \Pr(Y = 1 \mid x)$ and $w_0, w_1 > 0$. For $0 < \eta(x) < 1$, the unique minimizer of the conditional weighted BCE is*

$$p^*(x) = \frac{w_1 \eta(x)}{w_1 \eta(x) + w_0 (1 - \eta(x))}, \qquad \operatorname{logit} p^* = \operatorname{logit} \eta + \log \frac{w_1}{w_0}. \tag{B.7}$$

Proof. The conditional risk is $R(p) = -w_1 \eta \log p - w_0 (1 - \eta) \log(1 - p)$. Its derivative vanishes when $w_1 \eta (1 - p) = w_0 (1 - \eta) p$, giving the first expression. Moreover,

$$R''(p) = \frac{w_1 \eta}{p^2} + \frac{w_0 (1 - \eta)}{(1 - p)^2} > 0, \tag{B.8}$$

which proves uniqueness. Taking the ratio $p^*/(1 - p^*)$ proves the log odds identity. Boundary cases follow by continuity. If $D = w_1 \eta + w_0 (1 - \eta)$, the excess risk also satisfies

$$R(p) - R(p^*) = D \, \mathrm{KL}\big(\mathrm{Bern}(p^*) \,\|\, \mathrm{Bern}(p)\big), \tag{B.9}$$

obtained by collecting the two logarithmic terms. □

The corresponding inverse transformation is

$$\eta = \frac{w_0 p^*}{w_1 (1 - p^*) + w_0 p^*}. \tag{B.10}$$

This population identity explains how class weighting changes score interpretation; calibration of the fitted router is assessed separately through empirical diagnostics.

B.3 Asymmetric Costs and Operating Thresholds

Let C_{unsafe} denote the cost of selecting a KEEP interaction and C_{miss} the cost of retaining a READY interaction. Both are positive. Correct label decisions have zero cost in this analysis; summary generation errors are considered separately in Appendix E.

Proposition B.3 (Cost Sensitive Selection). *If $\eta(x)$ is known, the minimum conditional label risk is attained by selection whenever*

$$\eta(x) \geq \eta_0 := \frac{C_{\text{unsafe}}}{C_{\text{unsafe}} + C_{\text{miss}}}. \tag{B.11}$$

For the ideal weighted score, the equivalent threshold is

$$\tau_w = \frac{w_1 C_{\text{unsafe}}}{w_1 C_{\text{unsafe}} + w_0 C_{\text{miss}}}. \tag{B.12}$$

Proof. The conditional costs of selection and retention are $C_{\text{unsafe}}(1-\eta)$ and $C_{\text{miss}}\eta$, respectively. Comparing them gives Equation (B.11). The map $\eta \mapsto p^*$ in Equation (B.7) is strictly increasing. Substitution of η_0 therefore gives the equivalent weighted threshold. More explicitly, a decision that disagrees with this cost optimal decision has excess conditional risk

$$(C_{\text{unsafe}} + C_{\text{miss}})|\eta - \eta_0|. \tag{B.13}$$

This follows by subtracting the smaller of the two conditional costs from the larger one. □

Class weighting and conservative decision thresholds have different roles. The former changes the training loss; the latter determines the operating tradeoff. In the experiments, validation data determine operating choices; the posterior and cost ratio specify the corresponding theoretical decision rule.

C Bounded Representation and Efficiency

C.1 Input Construction and Computational Cost

For each surviving historical target, the bounded construction forms

$$V_{i,k} = \mathcal{B}_M(S_i, H_k; K_{\text{ctx}}), \qquad |V_{i,k}| \leq M, \qquad \widetilde{h}_{i,k} = F_\phi(V_{i,k})_{|V_{i,k}|}. \tag{C.1}$$

It combines the target with current context and a limited portion of the effective history. The representation comparison uses a 4,096-token base budget and up to 1,024 additional target tokens, for $M = 5,120$. Recent4 specifies the representation context, while the span length threshold controls eligibility for online compression operations.

Let L_k denote full context length, N_k the number of surviving targets, $\mathcal{F}(n)$ the cost of a length-n model pass, and c_g the router cost per target. Fresh full context extraction and target specific bounded extraction have workloads

$$C_{\text{full}}(k) = \mathcal{F}(L_k) + O(N_k c_g), \qquad C_{\text{bounded}}(k) \leq N_k \mathcal{F}(M) + O(N_k c_g). \tag{C.2}$$

One full pass can supply all historical readouts. Conversely, bounded extraction uses multiple short views, which may be batched; its total work still depends on N_k. For a dense attention model with L layers and width d, the usual arithmetic model is $\mathcal{F}(n) = O(L(nd^2 + n^2 d))$. Across a trajectory,

$$C_{\text{trajectory}} \leq \sum_k N_k \mathcal{F}(M) + \sum_k O(N_k c_g). \tag{C.3}$$

A bounded view controls individual input size, while checkpoint cost depends on the surviving target count.

C.2 Stability of Decisions and Selected Spans

Proposition C.1 (Bounded perturbations preserve routing and spans). *Suppose full and bounded features $x_i, \widetilde{x}_i$ are supplied to the same logit function g, with $|g(x_i) - g(\widetilde{x}_i)| \leq L_g \epsilon_i$. Let $a_\tau = \log(\tau/(1-\tau))$, $0 < \tau < 1$. The decision for target i is unchanged if*

$$|g(x_i) - a_\tau| > L_g \epsilon_i. \tag{C.4}$$

If this holds for every eligible interaction, and the source history and execution gates are unchanged, the two feature constructions produce the same maximal selected spans and the same gate eligible candidates.

Proof. A score more than $L_g \epsilon_i$ above the threshold remains above it after the perturbation; a score more than that amount below it remains below it. Since the sigmoid is strictly increasing, thresholding its output is equivalent to thresholding the logit at a_τ. This proves pointwise decision equality. Equality for every target gives identical binary masks. Maximal runs are determined uniquely by the mask and the fixed boundaries of eligible interactions. Each resulting run consequently has the same source length and token count, so both execution gates return the same answer. □

The result applies to a fixed router under the stated perturbation bound. The ablations measure empirical performance across separately fitted feature variants, while Equation (C.4) characterizes fixed router stability under bounded perturbations.

C.3 Representation Measurements

The input size study covers 110 trajectories and 2,558 states. Mean bounded input is 4,181.5 tokens; 86 trajectories exceed the base budget, 73 reach the 5,120-token limit, and 178 current step inputs are truncated. Timing uses seven states on which both implementations complete on the same device. The representation quality comparison is separate from the boundary label router study in Appendix D.

Table A3: **Representation efficiency.** Input statistics, paired runtime, and classification quality refer to their respective measurements.

Metric	Full history	Bounded
Mean input tokens	n/a	4,181.50
Largest recorded input	203,675	5,120
Representative long input	127,833	5,120
Paired extraction time (s)	76.40	6.03
Mean latency (s/state)	10.914	0.861
Peak GPU memory (GiB)	29.426	18.620
Forward workspace (GiB)	11.867	1.062
Throughput (states/s)	0.0916	1.1611
Separate test F1	0.7000	0.7818

Total extraction time decreases from 76.40 to 6.03 seconds, a $12.67\times$ measured speedup. Peak GPU allocation and forward workspace measure different quantities. The full model memory result comes from a successful 57,401-token input; inputs of 131,576 and 203,675 tokens exhaust GPU memory and are not included in the paired timing. This speedup characterizes representation extraction within the complete-task execution pipeline.

The capped hybrid achieves the highest F1, while capped same prefix selection achieves the highest balanced accuracy. The effects of recurrent and historical features vary across the evaluated input constructions. These comparisons measure predictive performance under each construction, including the restricted context available to the bounded variants relative to the full history representation.

Table A4: **Context and historical feature ablations in the representation comparison.**

Representation	Test F1	Balanced accuracy
Full Context	0.7000	0.5951
Recent2	0.7101	0.5940
Recent4	0.7340	0.6036
Local + GRU	0.7531	0.5979
Local + same prefix Top k	0.7545	0.6022
Hybrid	0.6749	0.6011
Capped Recent4	0.7518	0.5867
+ GRU	0.7465	0.6048
+ same prefix Top k	0.7477	**0.6156**
+ Hybrid	**0.7818**	0.5988

D Compression Dynamics and Prediction Diagnostics

D.1 Readiness and Compression Delay

At checkpoint k, the annotated readiness fraction is

$$a_k = \frac{1}{k-1} \sum_{i<k} y_{i,k}. \tag{D.1}$$

Even under monotone boundary labels, this fraction need not increase at every step. If u_k old interactions newly become READY and the new interaction has label $b_k = y_{k,k+1}$, then

$$a_{k+1} - a_k = \frac{u_k + b_k - a_k}{k}. \tag{D.2}$$

Indeed, the previous READY count is $(k-1)a_k$, and the next count is $(k-1)a_k + u_k + b_k$. Dividing by the new denominator k and subtracting a_k proves the identity. New KEEP interactions can lower the fraction without reversing any existing boundary label.

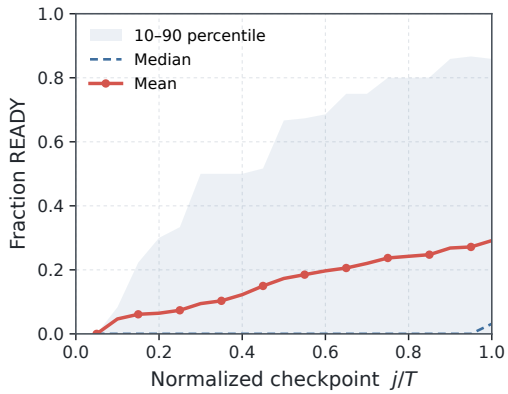

Figure A1: **Variation in compression opportunities.** Mean and median READY fractions and the 10th to 90th percentile range across trajectories.

The mean, median, and percentile range summarize variation across trajectories. The percentile band spans the central range of trajectory level READY fractions at each execution stage. Compression delay additionally describes how long an interaction remains in its original form before a supported boundary. For normalized delays d_i in a finite boundary group G, its empirical distribution is

$$\widehat{F}_G(u) = |G|^{-1} \sum_{i \in G} \mathbf{1}\{d_i \le u\}. \tag{D.3}$$

This distribution is conditional on a finite supported boundary.

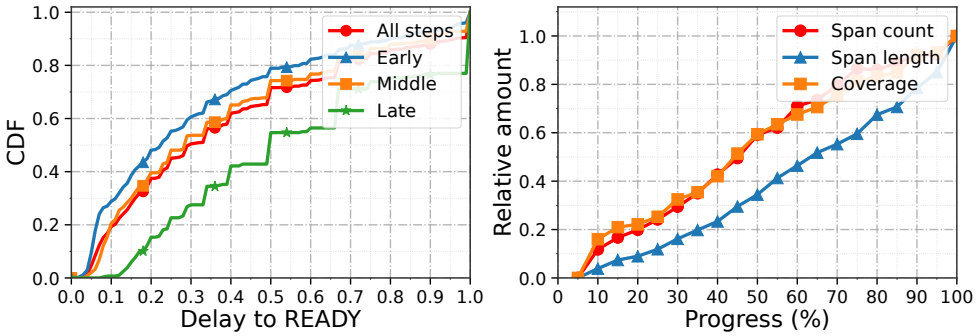

Figure A2: **Compression delay and span dynamics.** Left: normalized delay to READY by interaction position. Right: relative span count, span length, and coverage over task progress.

For a binary selection mask $a_1, \ldots, a_N$ within a consecutive eligible region, let $a_0 = 0$. Then

$$J = \sum_{i=1}^{N} a_i(1 - a_{i-1}), \qquad c = N^{-1} \sum_{i=1}^{N} a_i, \qquad \bar{\ell} = \frac{\sum_i a_i}{J} \quad (J > 0). \tag{D.4}$$

Each maximal selected run contributes one transition from zero to one, which proves the expression for J. Protected messages and existing summaries separate eligible regions. Thus, coverage, number of spans, and average span length describe different aspects of the selected history.

D.2 Router Configurations

Table A5: **Router configurations and operating points.** PR AUC and AUROC are ranking metrics; precision and recall are percentages at the reported operating point. n/a denotes an unreported metric.

Configuration	PR AUC	AUROC	Precision	Recall
BCE baseline	0.1685	0.6870	n/a	n/a
Weighted BCE ($w_1 = 10$)	0.1736	0.7096	n/a	n/a
Random oversampling 1:4	0.1674	n/a	n/a	n/a
Linear	0.1425	n/a	n/a	n/a
MLP 128	0.1455	n/a	n/a	n/a
MLP 2048	0.1655	n/a	n/a	n/a
Candidate + current + Top4	0.1455	0.7171	28.571	0.081
CB Focal ($\beta = 0.9999, \gamma = 3$)	0.1624	n/a	n/a	n/a
Weighted BCE ($w_1 = 5$)	n/a	n/a	72.222	0.111
Positive ratio sweep	0.1669	n/a	58.970	n/a
100% training trajectories	n/a	n/a	62.319	0.092

Weighted BCE with $w_1 = 10$ reaches AUROC 0.7096 and PR AUC 0.1736. The separate $w_1 = 5$ operating point has precision 72.222% at recall 0.111%, illustrating the limited coverage of a highly selective rule. The ranking and high precision results come from separately fitted configurations. These measurements assess held-out prediction; online compression additionally depends on span selection and summary fidelity.

The trajectory length groups contain 100 short, 100 medium, and 100 long trajectories. Prediction varies across these groups, whereas the displayed training fraction trend is comparatively flat. These comparisons characterize performance across trajectory length groups together with their associated domain composition and label prevalence.

D.3 Precision, Coverage, and Calibration

For READY as the positive class, precision is $TP/(TP + FP)$, recall is $TP/(TP + FN)$, and selected coverage is $(TP + FP)/N$. If positive prevalence is π, true positive rate is r, and false

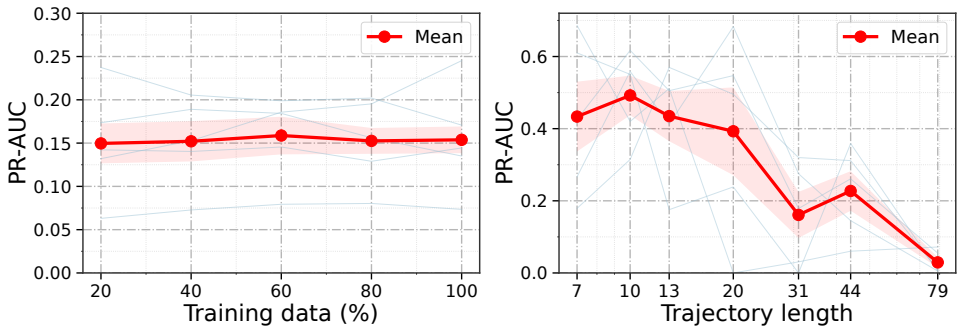

Figure A3: **Prediction across training fractions and trajectory length groups.** Thin curves show individual folds and the thick curve their mean.

positive rate is f, then

$$\text{Precision} = \frac{\pi r}{\pi r + (1-\pi)f}. \tag{D.5}$$

This follows by writing $TP = N\pi r$ and $FP = N(1-\pi)f$. In a low prevalence setting, false positives therefore strongly affect precision.

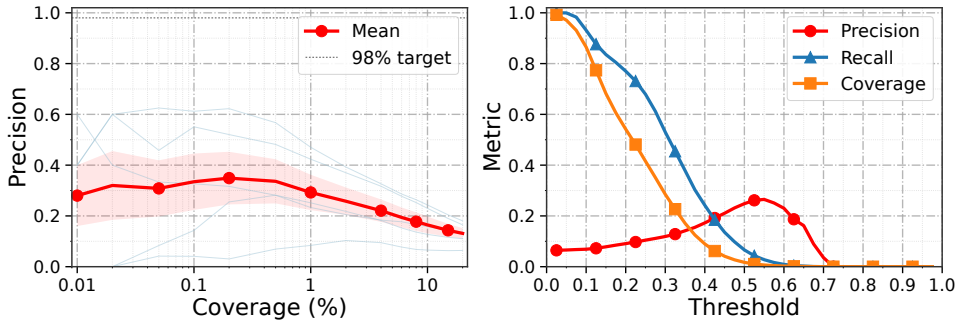

Figure A4: **Held out precision and coverage.** The 98% line marks the reference precision target for the held-out comparison.

Raising a threshold produces nested selected sets, so coverage and FPR cannot increase on fixed scores. Precision need not be monotone: removing a true positive can reduce the precision of the remaining set. These curves characterize the precision and coverage tradeoff against the 98% reference target.

For score bins $\mathcal{B}_b$, the calibration summary is

$$\text{ECE} = \sum_b \frac{|\mathcal{B}_b|}{N} \left| \frac{1}{|\mathcal{B}_b|} \sum_{n \in \mathcal{B}_b} y_n - \frac{1}{|\mathcal{B}_b|} \sum_{n \in \mathcal{B}_b} p_n \right|. \tag{D.6}$$

High score precision and aggregate ECE characterize different aspects of predictive reliability, particularly in sparsely populated score regions. Calibration and selective prediction are therefore evaluated as complementary properties.

E Online Compression and Analytical Properties

E.1 Candidate Spans and Execution

The online controller uses a frozen Qwen3.5-9B representation model with Layer 32 last token readouts. It concatenates the candidate interaction vector, the current checkpoint vector, and the mean of the same prefix Top4 historical vectors, giving a router input with 12,288 dimensions. The

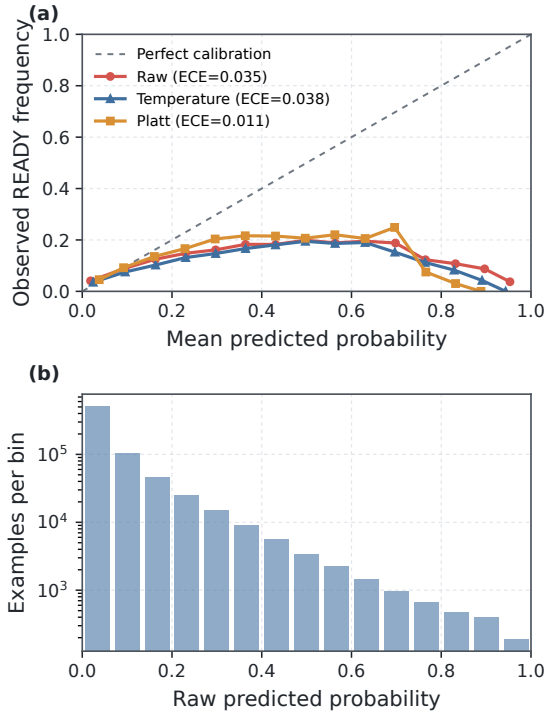

Figure A5: **Calibration diagnostics.** The displayed ECE values are 0.035, 0.038, and 0.011 for raw, temperature scaled, and Platt calibrated scores.

router has one hidden layer of width 2,048 and dropout 0.3. Online compression uses the threshold $\tau = 0.60$.

At every checkpoint, the router scores the surviving complete interactions in the effective history. Adjacent selected interactions form maximal spans; a KEEP interaction, protected message, or committed summary breaks a span. A candidate B is eligible when

$$\min_{i \in B} p_{i,k} \geq \tau, \qquad |B| > \kappa, \qquad \mathrm{Tok}(B) \geq B_{\min}. \tag{E.1}$$

The compression configuration is $(\tau, \kappa, B_{\min}) = (0.60, 3, 1{,}000)$, as specified in Section 4.2. The same execution rule is used for Full260, Eval40, and the cross model comparisons.

Each eligible span is summarized into facts, constraints, locations, confirmed conclusions, failure causes, and unresolved items useful to later actions. Replacements are accepted only if the complete serialized history remains protocol valid and becomes shorter. Later routing uses the resulting history, with no external recovery of removed originals.

E.2 Structural Preservation and Call Bounds

Proposition E.1 (Valid and nonexpanding updates). *Suppose the input history is protocol valid, candidate spans contain complete interactions, and each accepted replacement is protocol valid and shorter under the same serialization and tokenizer. A checkpoint returns a valid history no longer than its input. It is strictly shorter when a nonempty replacement transaction is committed.*

Proof. Let $H^{(0)} = H_k$ and let $H^{(j)}$ be the trial after the jth accepted replacement. The acceptance rule gives

$$\mathrm{Valid}(H^{(j)}) = 1, \qquad \mathrm{Tok}(H^{(j)}) < \mathrm{Tok}(H^{(j-1)}). \tag{E.2}$$

Induction proves validity and decreasing length for every accepted trial. A rejected proposal leaves the current trial unchanged. The final transaction either returns the last valid trial or the unchanged input.

Algorithm 1 One checkpoint of STATECOMP

Input: Valid effective history H_k; frozen model F_ϕ; trained router g_θ; threshold τ; span threshold κ; source budget $B_{\min}$.
Output: A valid history for the next agent request.

```
Preserve H_k as the fallback history
Construct bounded inputs for surviving complete raw interactions
Extract features and compute p_{i,k}
if feature extraction or routing fails then
    return H_k
end
Form maximal eligible spans satisfying Equation (E.1)
Initialize H̃_k ← H_k
foreach eligible span B, in reverse history order do
    Generate a summary m_B from B
    if summary generation succeeds then
        Construct a trial history by replacing B in H̃_k with m_B
        if the trial is protocol valid and shorter than H̃_k then
            Accept the trial as the new H̃_k and record source identifiers
        end
    end
end
if the complete proposed history passes final protocol checks then
    Commit H̃_k
    return H̃_k
end
return H_k
```

This proves the claim. This establishes non-expansion at the compression checkpoint; subsequent interactions may add tokens to the history. □

For a nonnegative integer κ, if a checkpoint has n_{sel} selected interactions, each eligible span consumes at least $\kappa + 1$ disjoint interactions. Consequently,

$$J_{\mathrm{eligible}} \le \left\lfloor \frac{n_{\mathrm{sel}}}{\kappa + 1} \right\rfloor . \tag{E.3}$$

This bounds summarization proposals under the one call per span algorithm, with transport retries outside the proposal count.

E.3 Interaction Errors and Summary Fidelity

Proposition E.2 (Conditional span error decomposition). *For a selected span B and available history H, let E_i be the event that interaction $i \in B$ is actually KEEP, and suppose $\Pr(E_i \mid H, B) \le \epsilon_i$. Let Q_B denote a harmful joint replacement despite all selected interactions having correct READY labels. Suppose $\Pr(Q_B \mid H, B, \bigcap_{i\in B} E_i^c) \le \delta_B$ whenever this conditioning event has positive probability. Then*

$$\Pr\left(\bigcup_{i\in B} E_i \;\cup\; \left[Q_B \cap \bigcap_{i\in B} E_i^c \right] \middle| H, B \right) \le \min\left\{ 1, \sum_{i\in B} \epsilon_i + \delta_B \right\}. \tag{E.4}$$

Proof. The two bracketed failure cases are disjoint. By the union bound, the probability of at least one incorrect READY label is at most $\sum_i \epsilon_i$. The second case has probability

$$\Pr\left(\bigcap_i E_i^c \middle| H, B \right) \Pr\left(Q_B \middle| H, B, \bigcap_i E_i^c \right) \le \delta_B. \tag{E.5}$$

If the first factor is zero, this case has zero probability. Adding the bounds and using that a probability cannot exceed one proves the result. Independence between interaction errors is not required. □

The term δ_B includes summary omissions and dependencies between joint replacements. For example, replacing both copies of a necessary identifier can lose information even when either copy would be replaceable alone. The bound is conditional on span level error probabilities, while aggregate precision provides a complementary empirical diagnostic. Minimum span size controls execution overhead; semantic reliability depends on routing accuracy, joint dependencies, and summary fidelity.

E.4 Amortized Tokens and Prefix Caching

Proposition E.3 (Token break even condition). *Fix an otherwise unchanged sequence of requests and outputs. Suppose a committed replacement saves Δ_r serialized input tokens at each subsequent request r, and its auxiliary summary call costs C_{sum} input and output tokens. Its net token saving is*

$$\Delta T = \sum_{r=1}^{R} \Delta_r - C_{\mathrm{sum}}. \tag{E.6}$$

For a constant saving $\Delta > 0$, the saving is positive exactly when $R\Delta > C_{\mathrm{sum}}$.

Proof. Under the fixed request sequence, all unchanged input and output terms cancel between the two accounts. The remaining input differences sum to $\sum_r \Delta_r$, and the additional summary call contributes C_{sum} only to the compressed account. Subtraction gives the identity and the break even condition. □

Actual online runs can change actions, retries, and termination times, so complete-task token use is measured separately. Monetary cost also depends on cached-input, uncached-input, and output prices.

By Proposition B.1, tokens before the first rewritten position keep the same causal prefix. Tokens at and after that position generally require recomputation. A span gate regulates rewrite frequency, while cache hit rate and end to end latency additionally depend on the serving configuration and prefix caching implementation.

F Benchmark Protocol and Supplementary Results

F.1 Task Sets and Metrics

WorkBuddyBench contains 80 Code, 50 Office, 60 Security, and 70 Web tasks. Eval40 contains ten tasks from each domain. With task reward r_t,

$$\overline{r} = \frac{1}{N}\sum_{t=1}^{N} r_t = \sum_{d} \frac{N_d}{N}\overline{r}_d. \tag{F.1}$$

Tables in the main text multiply reward by 100; the detailed tables below use the original zero to one scale. The overall score is the task weighted aggregate used in this paper. Failed task outcomes remain in fixed set comparisons.

For disjoint agent and summary request sets,

$$T_{\mathrm{total}} = \sum_{r\in\mathcal{R}_{\mathrm{agent}}} (I_r + O_r) + \sum_{r\in\mathcal{R}_{\mathrm{summary}}} (I_r + O_r). \tag{F.2}$$

Cached input and reasoning output are already included in the respective input and output totals. Local representation computation is reported separately. Monetary savings depend on the prices of token categories.

F.2 Full260 Results

Combined tokens decrease from 698,167,232 to 333,239,366, corresponding to 52.2694%. Mean reward changes from 0.698701 to 0.702556. Security reward decreases slightly while the other three domain rewards increase. Token reductions occur in every domain.

Table A6: **Full260 task rewards.** Differences use the original reward scale.

Domain	Tasks	Baseline	StateComp	Difference
Code	80	0.769865	0.771199	+0.001334
Office	50	0.818394	0.827696	+0.009302
Security	60	0.477555	0.473065	-0.004490
Web	70	0.721429	0.731429	+0.010000
Total	260	0.698701	0.702556	+0.003855

Table A7: **Full260 token consumption.** Agent and summary columns include input and output tokens.

Domain	Baseline	Agent	Summary	Combined	Reduction
Code	109,624,178	66,919,305	77,381	66,996,686	38.8851%
Office	59,924,344	26,475,291	43,653	26,518,944	55.7460%
Security	358,590,823	159,586,199	267,966	159,854,165	55.4216%
Web	170,027,887	79,703,907	165,664	79,869,571	53.0256%
Total	698,167,232	332,684,702	554,664	333,239,366	52.2694%

F.3 Eval40 Controls and Summary Overhead

The following comparisons use the same fixed 40 tasks, with ten tasks per domain. StateComp uses the compression configuration in Section 4.2. Summary tokens are reported alongside total tokens to show the auxiliary cost of compression.

Table A8: **Eval40 task rewards.** Each domain contains ten tasks; Overall covers all 40 tasks.

Method	Code	Office	Security	Web	Overall
Baseline	0.7293	0.8164	0.4458	0.7100	0.6754
StateComp	0.7603	0.8412	0.5101	0.7500	0.7154
Sliding Window $K = 5$	0.6681	0.6743	0.3232	0.5900	0.5639
Sliding Window $K = 10$	0.7437	0.7305	0.4007	0.6100	0.6212
Sliding Window $K = 20$	0.7931	0.8511	0.4028	0.7000	0.6867
Periodic Summary $n = 3$	0.6544	0.6766	0.2632	0.8266	0.6052
Periodic Summary $n = 5$	0.8360	0.7574	0.4003	0.8132	0.7017
PACE	0.7038	0.5575	0.4353	0.6402	0.5842

Using the unrounded recorded token totals, the token reduction is

$$100\left(1 - \frac{43{,}760{,}684}{75{,}496{,}039}\right) \approx 42.04\%. \tag{F.3}$$

The compression run uses approximately 0.077M summary tokens, compared with 3.01M for periodic summarization at $n = 5$. The full control table shows both the performance losses of short windows and the overhead of repeated summarization.

Additional published baselines. The main comparison also includes published context management methods. LLMLingua-2, SelfCompact, ACON-Core, Self-GC, LRE, CoMem, SAM, SWE-Pruner, Sculptor, and ACM are evaluated on the same fixed Eval40 tasks. Their token totals use the same accounting rule as the other Eval40 methods. ACM uses a released policy, rather than the same API based acting model.

F.4 Cross-Model Results

All acting models are evaluated on the same fixed Eval40 tasks with the compression rule in Section 4.2. Each domain contains ten tasks, so overall reward is the equally weighted mean of the four domain rewards. Summary usage describes the auxiliary cost of compression.

Table A9: **Eval40 total tokens and summary overhead.** Counts are in millions. Summary tokens are included in the total; totals follow the precision of the main text comparison.

Method	Total tokens (M)	Summary tokens (M)
Baseline	75.50	0
StateComp	43.76	0.077
Sliding Window $K = 5$	118.08	0
Sliding Window $K = 10$	79.39	0
Sliding Window $K = 20$	80.82	0
Periodic Summary $n = 3$	88.57	4.50
Periodic Summary $n = 5$	94.85	3.01
PACE	44.12	4.82

Table A10: **Cross model Eval40 rewards.** Rewards use the zero to one scale. Each domain contains ten tasks.

Model	Method	Code	Office	Security	Web	Overall
Hunyuan	Base	0.5530	0.7924	0.6024	0.4600	0.6019
Hunyuan	StateComp	0.6431	0.8220	0.6014	0.4900	0.6391
Qwen3.8-Flash	Base	0.7525	0.8342	0.4528	0.8600	0.7249
Qwen3.8-Flash	StateComp	0.9346	0.8871	0.4120	0.8800	0.7784
GLM-5.3-Flash	Base	0.7949	0.6613	0.3467	0.6651	0.6170
GLM-5.3-Flash	StateComp	0.6615	0.6990	0.3864	0.8500	0.6492
MiMo-V2.5	Base	0.5376	0.7705	0.2204	0.5500	0.5196
MiMo-V2.5	StateComp	0.5765	0.7164	0.2277	0.5800	0.5252

All four acting models use fewer total tokens and achieve higher mean reward with compression. Domain level reward changes vary by backbone. The totals summarize complete-task usage, with summary tokens already included as the auxiliary cost of the compression procedure.

G Parameter Analysis and Evaluation Tasks

G.1 Operating Settings

The probability threshold τ, representation window K_{ctx}, sliding window baseline K_{sw}, source token threshold $B_{\min}$, and span length threshold κ control distinct operations. Full260, Eval40, and the cross model comparisons use $\tau = 0.60$, $\kappa = 3$, and $B_{\min} = 1{,}000$ tokens. Thus, a span must contain at least four complete interactions and satisfy the source token threshold before summarization. The representation budget remains at most $5{,}120$ tokens per target view.

Table A11: **Cross model total tokens and summary overhead.** Counts are in millions. Summary tokens are included in the total.

Model	Method	Total tokens (M)	Summary tokens (M)
Hunyuan	Base	41.00	0
Hunyuan	StateComp	12.83	0.130
Qwen3.8-Flash	Base	49.75	0
Qwen3.8-Flash	StateComp	38.23	0.236
GLM-5.3-Flash	Base	47.58	0
GLM-5.3-Flash	StateComp	38.49	0.101
MiMo-V2.5	Base	53.55	0
MiMo-V2.5	StateComp	18.91	0.028

G.2 Fixed Eval40 Task Identifiers

The fixed task list contains ten tasks from each domain. It specifies the benchmark subset used for the Eval40 comparisons.

Table A12: **Fixed Eval40 task identifiers.**

No.	Domain	Task identifier
1	Code	api_contract-hard-markup_errors
2	Code	api_contract-hard-openapi_params
3	Code	api_contract-hard-token_errors
4	Code	api_contract-hard-validation_errors
5	Code	bug_fix-easy-a_crash_in_local
6	Code	bug_fix-easy-filtered_relation_queryset_arg
7	Code	bug_fix-easy-invalid_filterwarnings_regex_error
8	Code	bug_fix-medium-error_key_uses_data_key
9	Code	bug_fix-medium-errors_from_earlier_indices
10	Code	bug_fix-medium-incorrect_linenos_on_fstring
11	Office	analyst-forecast-extract-L3-018
12	Office	api-usage-explain-cli-l3-001
13	Office	board-material-update-timeline-excel
14	Office	calendar-dida-sync-state
15	Office	channel-period-compare-L4-017
16	Office	cloudagent-sdk-doc-validation-report
17	Office	contract-extract-L3-014
18	Office	cross-week-dashboard-migration
19	Office	crypto-backtest-chain-L4-002
20	Office	daily-creation-checkpoint-recovery
21	Security	agent-to-agent-injection-hard-multistep
22	Security	apt-multi-source-correlation-hard-multistep
23	Security	bb-bin-dns-parse-010
24	Security	bb-bin-firmware-audit-007
25	Security	bb-bin-format-log-004
26	Security	bb-bin-int-length-005
27	Security	bb-bin-ipc-cache-001
28	Security	bb-bin-media-parse-008
29	Security	bb-bin-oob-read-003
30	Security	bb-bin-parse-crash-006
31	Web	animated-explainer-L3-028
32	Web	atmosphere-game-L4-035
33	Web	blog-editor-draft-recovery-L4-059
34	Web	browser-clipper-extension-L4-005
35	Web	canvas-webgl-scene-L4-026
36	Web	chart-generation-L2-025
37	Web	checkout-incident-analysis-L4-049
38	Web	city-article-theme-variants-L4-066
39	Web	claims-drawer-state-review-report-L3-071
40	Web	cohort-retention-dashboard-L4-054