ENCODING CONTEXT INTO LLM WEIGHTS: WHAT MAKES IT CONTEXT?

Anonymous authors
Paper under double-blind review

ABSTRACT

Longer contexts allow large language models (LLMs) to condition their predictions on more information, but inference costs grow prohibitively with context length. An emerging alternative uses test-time updates to encode context as temporary changes to model weights. However, encoding this information into weights that already represent general knowledge obscures its role as context, so the LLM may fail to use it properly to condition subsequent inference. To restore this contextual role, we first associate each context with an identifier during encoding, thereby distinguishing the newly encoded information from pre-existing knowledge. Invoking the identifier at inference then enables the LLM to recite the encoded context. Nevertheless, successful recitation does not ensure that the LLM will use the recalled information as context. We therefore further introduce a prompting framework that explicitly guides the LLM to use this information as context. Across diverse controlled tasks and practical benchmarks, our method outperformed the baseline using only off-the-shelf LLMs, showing its advances in context encoding.

1 INTRODUCTION

Large language models (LLMs) have advanced substantially in recent years (Brown et al., 2020; Grattafiori et al., 2024; Guo et al., 2025), and a key foundation of this progress is their expanding capacity to process long contexts (Gemini Team et al., 2024) and to scale inference-time computation (Muennighoff et al., 2025). The former allows LLMs to ingest larger volumes of diverse inputs, including elaborate prompts, documents, multimodal data, dialogue histories, and tool-interaction records (Achiam et al., 2023). The latter supports test-time scaling through extended reasoning trajectories involving iterative refinement and verification (Wei et al., 2022; Snell et al., 2024).

These benefits, however, come with limitations. As context grows, key-value (KV) cache memory scales linearly and causal attention computation scales quadratically, imposing prohibitive hardware demands and deployment costs (Pope et al., 2023). Models may also lose track of relevant information in longer contexts (Liu et al., 2024; Du et al., 2025), which has been linked to boundary-token biases (Hsieh et al., 2024b), limited length generalization (Zhong et al., 2025), and attention dispersion (Veličković et al., 2024). LLMs are therefore constrained by their input context windows, reaching only the million-token scale even in the most advanced models (Yang et al., 2025; OpenAI, 2026), while their effectively usable length can be shorter (Hsieh et al., 2024a; An et al., 2025).

Taken together, the above observations reveal a tension: LLMs benefit from consuming more tokens at inference, yet an ever-growing context eventually exhausts the available context window. This tension motivates recent work on *test-time context encoding* (TCE) (Chen et al., 2023; Sun et al., 2025; Tandon et al., 2025; Zhang et al., 2025b; Behrouz et al., 2025b; Feng et al., 2026; Charakorn et al., 2026), which encodes prior context through temporary weight updates rather than retaining it in the input (Figure 1). By encoding context without occupying KV cache positions, TCE can potentially mitigate the computational costs and performance degradation associated with long input contexts. Ideally, an LLM should condition its predictions on encoded information as reliably as if that information remained within its effective input window. Prior work has reported promising TCE gains using autoregressive encoding (Chen et al., 2023; Tandon et al., 2025; Chen et al., 2026), often paired with additional meta-training-based fine-tuning, given its "*learning to (learn at test time)*" nature (Sun et al., 2025), to improve the use of encoded information.

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.

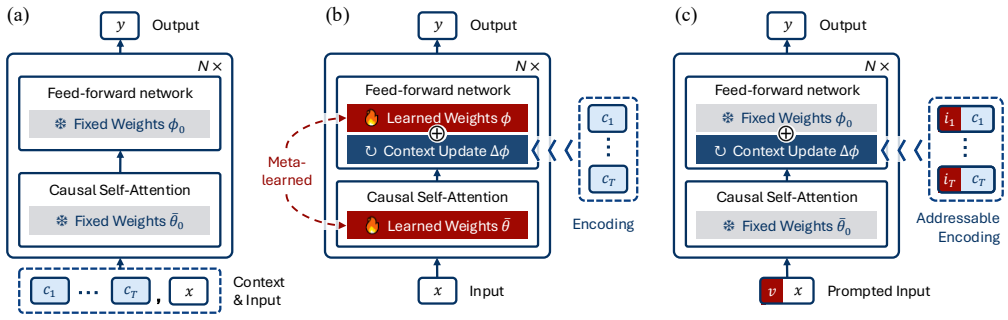


Figure 1: Comparison of (a) full-context inference, (b) encoded-context inference with meta-trained TCE, and (c) encoded-context inference with **ACE**. Full-context inference jointly processes $c_{1:T}$ and x, becoming infeasible when their combined length exceeds the input context window. TCE and **ACE** instead encode the chunks $c_{1:T}$ into LLM weights, illustrated here as FFN weights. Prior TCE methods often rely on additional meta-training-based fine-tuning, whereas **ACE** operates directly on off-the-shelf LLMs without such fine-tuning.

However, additional meta-training can itself improve downstream performance, making it difficult to attribute the reported gains specifically to context encoding. This leaves open whether TCE can work effectively with off-the-shelf LLMs. Although autoregressive objectives can enable memorization (Chen et al., 2023), our experiments in Section 4 showed that TCE without such fine-tuning remained far inferior to *full-context inference*, which retains the context in the input. This gap highlights the distinction between memorization and contextual use, as also discussed in prior work (Eyuboglu et al., 2025; Chen et al., 2026). We therefore investigate how off-the-shelf LLMs can reliably use encoded information as context without such task-specific meta-training.

This problem is nevertheless challenging because newly encoded context and pre-existing general knowledge share the same weights, while current TCE objectives provide no explicit signal that distinguishes the encoded information as active context. This lack of distinction, in turn, makes the influence of such information on subsequent inference difficult to control, leaving it uncertain whether the resulting LLM behavior will be reliably conditioned on that context. To this end, we propose *Addressable Context Encoding* (**ACE**), which makes encoded context distinguishable from pre-existing knowledge through a simple addressing mechanism. Specifically, during encoding, we associate each context with a *unique identifier* (UID), such as a randomly generated alphanumeric string. At inference, invoking the relevant UIDs directs the LLM to recite the associated content, thereby providing an explicit access path to the encoded information. Moreover, going beyond mere recitation, we wrap this addressing mechanism in a flexible prompting framework (Kojima et al., 2022) that further guides the LLM to selectively use and reason over the encoded information when responding to subsequent requests (Figure 1), thus making its behavior more controllable.

We stress-tested **ACE** on controlled tasks covering various aspects of context use, observing substantial improvements over classical TCE. For example, on two such tasks, Recall and Recognition, **ACE** increased accuracy from 1.5% to 75.6% and from 0.0% to 98.0%, respectively, on Qwen3.5-27B (Qwen, 2026). These benefits extended to long-context benchmarks, including RULER (Hsieh et al., 2024a), widely used in prior TCE studies (Tandon et al., 2025; Feng et al., 2026), and the more challenging LongBench (Bai et al., 2024) and LongBench v2 (Bai et al., 2025), which have received less attention in this setting. With Muse-Glimmer-30B (Meta, 2026), **ACE** improved RULER NIAH-3 accuracy from 8% to 63% and LongBench answer F1 from 14.0 to 19.2.

2 Background

Let p_{θ_0} denote the conditional distribution of a well-trained LLM with weights θ_0 and an $L_{\max}$-token input context window. Let C denote prior context, x a subsequent request, and y the corresponding response. When C and x fit within the window, full-context inference generates

$$y \sim p_{\theta_0}(\cdot \mid C, x), \qquad |C| + |x| \leq L_{\max}, \tag{1}$$

where $|\cdot|$ denotes token length. In this setting, the model can directly use the information and interaction state in C when generating y in response to x.

2.1 Long-context Inference

As $|C| + |x|$ approaches $L_{\max}$, full-context inference incurs increasing memory and compute costs (Dao et al., 2022), and the model may use relevant information less reliably (Liu et al., 2024). Mainstream approaches alleviate these problems through context compression (Jiang et al., 2024) or KV-cache eviction (Zhang et al., 2023) but risk discarding task-relevant information. *Test-time context encoding* (TCE) (Chen et al., 2023; Sun et al., 2025; Tandon et al., 2025; Zhang et al., 2025b; Behrouz et al., 2025b) offers a promising alternative by storing context in a high-capacity *parametric state*, potentially preserving more information than explicit context-reduction methods.

We consider a general long-context setting in which accumulated interaction histories may yield $|C| \gg L_{\max}$, as in long-running agentic deployments (Wu et al., 2025). We therefore partition C into T temporally ordered, contiguous chunks $c_{1:T}$ such that

$$C = c_1 \| \cdots \| c_T, \qquad |c_t| \leq B, \tag{2}$$

where $\|$ denotes concatenation. The chunk budget B ensures that each encoding input fits within $L_{\max}$. The context C may include factual information, dialogue history, instructions, preferences, pending queries, or unfinished reasoning, while x may be a query, instruction, or continuation cue.

2.2 Test-time Context Encoding

A subset of the model weights, referred to as *fast weights* (Feng et al., 2026), serves as the parametric state into which the context is encoded. We write $\theta_0 = (\bar{\theta}_0, \phi_0)$, where ϕ_0 denotes the initial fast weights, to which we reset them before encoding each independent context C, while $\bar{\theta}_0$ remains frozen during TCE. For each chunk c_t, we minimize the autoregressive loss with respect to ϕ, i.e.,

$$\mathcal{L}(\phi; c_t) = -\log p_{\bar{\theta}_0, \phi}(c_t). \tag{3}$$

Starting from ϕ_0, we encode the chunks $c_1, \ldots, c_T$ in temporal order. Let $\phi_t^{(k)}$ denote the fast weights after k optimization steps on the chunk c_t. We initialize $\phi_t^{(0)} = \phi_{t-1}$ from the state produced by the preceding chunks and perform K updates as follows:

$$\phi_t^{(k+1)} = \mathrm{Update}\left(\phi_t^{(k)}, \nabla_\phi \mathcal{L}(\phi_t^{(k)}; c_t)\right), \qquad k = 0, \ldots, K-1, \qquad \phi_t = \phi_t^{(K)}. \tag{4}$$

Update denotes one optimizer step, such as gradient descent (Robbins & Monro, 1951) or Adam (Kingma & Ba, 2014). Thus, ϕ_t accumulates updates from the ordered context prefix $c_{1:t}$, yielding the context-encoded model weights $\theta_C = (\bar{\theta}_0, \phi_T)$ with respect to C after all T chunks.

Once encoded, C is removed from the input, and the context-encoded LLM generates

$$y \sim p_{\theta_C}(\cdot \mid x), \qquad |x| \leq L_{\max}. \tag{5}$$

Information from C is expected to influence subsequent predictions through the updated fast weights ϕ_T. The goal is for the resulting LLM to use this information as reliably as it would if C remained directly available in the input (Chen et al., 2023; Eyuboglu et al., 2025; Chen et al., 2026). When C is no longer needed, such as after a conversation ends, its context-specific updates can be discarded by resetting the fast weights from ϕ_T to ϕ_0, thereby restoring the LLM weights from θ_C to θ_0.

2.3 Scope of This Study

Prior work has explored TCE objectives other than the autoregressive loss (Sun et al., 2025; Behrouz et al., 2025b; Feng et al., 2026; Charakorn et al., 2026), as discussed in Section 5. However, these alternatives may perform worse than autoregressive encoding (Tandon et al., 2025), generalize poorly to practical use (Caccia et al., 2025), or incur substantial computational costs (Behrouz et al., 2025b). We therefore primarily focus on autoregressive TCE.

Most prior work reporting promising results relies on additional meta-training (Chen et al., 2023; Behrouz et al., 2025b; Tandon et al., 2025; Feng et al., 2026; Kuratov et al., 2026; Chen et al., 2026). Such fine-tuning may introduce reliance on task-specific correlations, making it difficult to separate its contribution from that of the underlying TCE mechanism, as we discuss later. We therefore restrict our study to off-the-shelf LLMs, aiming to establish a clearer understanding of TCE reliability.

Moreover, TCE requires backpropagation and thus incurs additional computation. Nevertheless, fast-weight updates are typically restricted to selected LLM weights (Tandon et al., 2025; Chen et al., 2026; Feng et al., 2026), reducing the number of trainable weights. Further improvements in efficiency and implementation are beyond our focus on reliability and are left for future work.

3 New Method

Without task-specific meta-training, the TCE framework introduced in Section 2.2 performed far worse than full-context inference, even in relatively simple experiments with very short contexts (Section 4.1). We attribute these failures in part to the fact that autoregressive encoding updates LLM weights much like pretraining and supervised fine-tuning, without explicitly identifying the encoded information as context. Although prior work reports a tendency for weight updates to bias subsequent predictions toward newly encoded information (Meng et al., 2022; Malladi et al., 2023; Zhang et al., 2025a), this implicit bias does not reliably ensure that the LLM recalls the encoded information, let alone treats it as context. These observations suggest two challenges: **(C1)** making encoded information reliably accessible and **(C2)** specifying how it should function as context.

Similar views have been raised in prior work (Chen et al., 2023; Tandon et al., 2025; Chen et al., 2026), yet these approaches do not resolve the fundamental limitation but instead turn to task-specific meta-training to sidestep it. We fill this gap by tackling **C1** and **C2** through two components, respectively: an addressable mechanism for making encoded information reliably accessible and explicitly distinguishing it from pre-existing LLM knowledge (Section 3.1), and a prompting framework for specifying how it should function as context, thus guiding its proper use in subsequent predictions (Section 3.2). Combining these two components, we propose *Addressable Context Encoding* (**ACE**), an enhanced TCE method that uses only off-the-shelf LLMs, as detailed in Section 3.3.

3.1 Addressable Mechanism

Enabling LLMs to reliably use encoded context first requires making the encoded information explicitly distinguishable from pre-existing knowledge, as specified by **C1**. One way to pursue this goal is to make the encoded context *addressable* by binding the context to a *unique identifier* (UID) during encoding. Conditioning on the UID provides an explicit cue for accessing the corresponding information during inference, thereby helping distinguish it from other knowledge.

Formally, we assign each context chunk c_t a UID i_t. For example, in `ctx-573277c6`, `ctx-` indicates context and the suffix is a random alphanumeric string. This format is illustrative, and we impose no particular format or design rule beyond requiring uniqueness. To associate each chunk with its UID, we modify the autoregressive objective in Eq. (3) to predict the chunk conditioned on its UID, i.e.,

$$\mathcal{L}_{\mathrm{A}}(\phi; c_t, i_t) = -\log p_{\bar{\theta}_0,\phi}(c_t \mid i_t), \tag{6}$$

which learns an association between the UID i_t and its context chunk c_t. We retain the sequential fast-weight updates as in Eq. (4), replacing only the basic autoregressive loss with $\mathcal{L}_{\mathrm{A}}$.

At inference, we remove the original context and supply the corresponding UIDs in temporal order. Let $I_C = i_1 \| \cdots \| i_T$ denote this ordered sequence. Given a subsequent input x, the model generates

$$y \sim p_{\theta_C}(\cdot \mid I_C, x), \qquad |I_C| + |x| \leq L_{\max}. \tag{7}$$

Since each UID is short, the added token overhead is small for a moderate number of chunks, leaving most of the input budget for the new request x. I_C references the encoded context without reproducing its content in the input, providing an explicit cue for the LLM to recall the encoded information and potentially facilitating its use in responding to the request x. Appendix B further provides mathematical insights into selective access to encoded context and a proof-of-concept analysis showing how UID conditioning can improve encoding dynamics.

3.2 Prompting Enhancement

Through this addressing mechanism, the LLM can recall the corresponding context when prompted with its UID. When further given a request x, it is expected to prioritize relevant encoded information over pre-existing knowledge. However, this behavior depends on inherent model capabilities that may

vary across models and are difficult to characterize, leaving implicit how the encoded information should guide subsequent inference. This motivates explicitly specifying its contextual role and use, as required by **C2**. To address this limitation, we build a prompting framework on top of the addressing mechanism. It leverages the instruction-following capabilities of off-the-shelf LLMs to explicitly specify the role and use of encoded information as context during both encoding and inference.

During encoding, we make the role and intended use of each context chunk c_t explicit in natural language through a *role-binding prompt*. It identifies the chunk as prior conversational context, specifies its temporal order, highlights its assigned UID, and indicates that relevant information should be used when that UID is invoked. We use the following template:

> **Role-Binding Prompt**
>
> *The following text is a context chunk from the current conversation at s_t, identified by i_t. It has been encoded into the model and should be treated as conversational context whenever i_t is invoked: c_t*

Here s_t denotes the timestamp assigned to chunk c_t, indicating its temporal order. These instructions clarify the contextual role of the chunk and provide metadata about its identity and temporal order. To encode this information alongside the chunk, we introduce an additional autoregressive objective that complements the UID-conditioned objective in Eq. (6). Specifically, let b_t denote the prompt prefix preceding c_t in the template above. We define the role-binding objective as

$$\mathcal{L}_{\mathrm{B}}(\phi; c_t, b_t) = -\log p_{\bar{\theta}_0,\phi}(b_t \| c_t). \tag{8}$$

During inference, we offer a more direct prompt to guide the use of encoded information. As in the addressing mechanism, we supply the UIDs I_C of the encoded chunks C, but further instruct the model to selectively use their associated content. Specifically, we ask the LLM to recall only information relevant to the current request x and use it to formulate the response, without reciting the entire context, which would otherwise consume the limited context window and undermine the original purpose of TCE. We introduce the *context-invocation prompt*, which is given by

> **Context-Invocation Prompt (`ci_v1`)**
>
> *Parts of the conversation have been encoded as the context chunks, ordered by time: i_1 at s_1, . . ., i_T at s_T. Treat these chunks as prior conversational context in subsequent inference. Selectively use information relevant to the current request without reproducing unrelated details.*

We label this template `ci_v1` to distinguish it from other templates evaluated in our experiments and shown in Appendix A.2. We denote the *context-invocation prompt* for the encoded context C with UIDs I_C by v_C. Then, given a subsequent input x, we prepend v_C and generate

$$y \sim p_{\theta_C}(\cdot \mid v_C, x), \tag{9}$$

providing more explicit control over the use of encoded information than the addressing mechanism alone. Note that we still assume that the context window is large enough to accommodate v_C and x.

3.3 Overall Algorithm

We summarize the overall **ACE** algorithm, which consists of encoding and inference procedures. During encoding, the model sequentially encodes the context chunks. Starting from $\phi_t^{(0)} = \phi_{t-1}$, we perform the following update for $k = 0, \ldots, K-1$:

$$\phi_t^{(k+1)} = \mathrm{Update}\left(\phi_t^{(k)}, \nabla_\phi \left[\mathcal{L}_{\mathrm{A}}(\phi; c_t, i_t) + \mathcal{L}_{\mathrm{B}}(\phi; c_t, b_t)\right]\big|_{\phi=\phi_t^{(k)}}\right). \tag{10}$$

After K steps, we set $\phi_t = \phi_t^{(K)}$. Once all chunks have been encoded, we obtain $\theta_C = (\bar{\theta}_0, \phi_T)$. Then, given x, the encoded LLM generates $y \sim p_{\theta_C}(\cdot \mid v_C, x)$. Unlike full-context inference, v_C does not reproduce the original chunks. Instead, it provides a compact index of the encoded chunks and specifies how they should guide inference. Moreover, when the context C is no longer needed, we restore the original model weights θ_0 by resetting the fast weights from ϕ_T to ϕ_0.

4 EXPERIMENTS

We first evaluated **ACE** in controlled experiments covering different aspects of context use. We then assessed its practical effectiveness on standard long-context benchmarks: RULER (Hsieh et al., 2024a), widely used in prior work (Tandon et al., 2025; Feng et al., 2026), and LONGBENCH (Bai et al., 2024) and LONGBENCH V2 (Bai et al., 2025), which have received less attention in prior work.

4.1 SANDBOX TASKS

To begin with, we constructed a controlled benchmark to stress-test TCE across different forms of context use and examine whether **ACE** improved its reliability. The benchmark used template-based synthesis and factual data curation, pairing short contexts with queries. The contexts averaged approximately 11 tokens under the Qwen3.5 tokenizer. We targeted 200 instances for each context-use task or subtask and removed invalid candidates through further screening. We also included a separately constructed Retention test. These tasks are briefly described below.

- **Recall.** We constructed 197 synthetic entity–attribute–value statements, such as "*The registry code of Korvustre Wharf is GC-483.*" Each served as context for a recall query that provided the entity and attribute to request the value, e.g., "*What is the registry code of Korvustre Wharf?*" A response was scored as correct if it retrieved the right value.
- **Recognition.** Using the same 197 statements as in Recall, we asked, "*What information were you just told?*", which relies on recency as the only cue in the request. A response was scored as correct if it retrieved not only the correct value, but also the correct entity and attribute. Recall and Recognition distinguish content-cued retrieval from identifying recently encoded information.
- **Continuation.** We created 168 self-contained factual and arithmetic questions with short, automatically verifiable answers. Each question served as the context, and the LLM was asked to continue the conversation by answering the unresolved question. We measured accuracy, evaluating whether the model can recognize an encoded question as a pending request and act on it.
- **Conflict.** We tested conflicts between newly encoded context and pre-existing knowledge across three subtasks: Counterfactual (200 instances), Dispute (43 instances), and Outdated Information (17, 20, and 33 instances for Qwen3.5-4B, 9B, and 27B, respectively). These data were model-specific because pre-existing knowledge varies across models. By default, we measured accuracy against the contextual answer rather than that from pre-existing knowledge.
- **Retention.** As an additional check, we evaluated accuracy on 100 random query instances unrelated to the encoded context, covering arithmetic, general knowledge, language, and temporal reasoning. This aimed to assess whether TCE impaired general LLM performance.

We offer illustrative examples of our curated sandbox datasets in Appendix A.1. The context of each instance forms a single encoding chunk, i.e., $T = 1$, because it is sufficiently short. Note that all these tasks are relatively easy for full-context inference, which achieves near-perfect performance, but remain challenging for TCE, as we will demonstrate later.

By default, we updated only the weights of the final feed-forward network (FFN) layer, following prior work on selective FFN updates (Mitchell et al., 2021; Tandon et al., 2025; Feng et al., 2026) and motivated by its interpretation as a knowledge store (Geva et al., 2021; Dai et al., 2022; Meng et al., 2022). We used Adam (Kingma & Ba, 2014) with an initial learning rate of 10^{-4} and gradient clipping with a maximum norm of 1 (Pascanu et al., 2013), without further hyperparameter tuning. Moreover, we adopted a stopping threshold τ, set to 0.1 by default, and stopped encoding when the maximum context-token loss fell below τ for $\mathcal{L}_{\text{B}}$ in **ACE** or after 256 update steps, whichever occurred first. Beyond the context-invocation prompt introduced in Section 3.2, termed `ci_v1`, we evaluated two other variants, `ci_v2` and `ci_short`, to examine the impact of prompting on the reliability of **ACE**. Details of all three prompts can be found in Appendix A.2.

We first evaluated and compared classical TCE discussed in Section 2.2 and our proposed enhancement **ACE** on the Recall, Recognition, and Continuation tasks. We also reported results on Retention as a supplement to assess the impact of context encoding on general model performance. We tested both methods on the 4B, 9B, and 27B versions of the Qwen3.5 model family. Table 1 reports accuracy results. As shown, classical TCE fails to reliably use encoded context even on the relatively easy Recall task. In contrast, **ACE** substantially improves performance across the context-use tasks,

Table 1: Comparison between TCE and **ACE** on Recall, Recognition, Continuation, and Retention. We reported accuracy (%) on the Qwen3.5 family. Bold marks the best result within each model and task.

LLM	Method	Prompt	Recall	Recognition	Continuation	Retention
Qwen3.5-4B	TCE	–	6.1	0.5	0.0	**100.0**
	ACE	`ci_short`	52.3	**91.9**	46.4	99.0
	ACE	`ci_v1`	55.3	81.7	**48.8**	**100.0**
	ACE	`ci_v2`	**60.4**	80.7	28.0	**100.0**
Qwen3.5-9B	TCE	–	4.6	2.0	1.2	**100.0**
	ACE	`ci_short`	**40.1**	**96.4**	9.5	**100.0**
	ACE	`ci_v1`	35.5	95.4	**13.7**	**100.0**
	ACE	`ci_v2`	38.1	94.4	5.4	**100.0**
Qwen3.5-27B	TCE	–	1.5	0.0	0.0	**100.0**
	ACE	`ci_short`	**75.6**	**98.0**	**20.8**	**100.0**
	ACE	`ci_v1`	39.1	58.4	11.9	**100.0**
	ACE	`ci_v2`	45.2	50.8	2.4	**100.0**

Table 2: Comparison between TCE and **ACE** on the three categories of Conflict: Counterfactuals (Counterfact), Disputes (Dispute), and Outdated Information (Outdated). We further reported two-sided Counterfactual accuracy in the column labeled "Counterfact (two-sided)."

LLM	Method	Prompt	Counterfact	Dispute	Outdated	Counterfact (two-sided)
Qwen3.5-4B	TCE	–	3.5	7.0	5.9	0.5
	ACE	`ci_short`	86.5	83.7	88.2	20.0
	ACE	`ci_v1`	92.5	**95.3**	**100.0**	27.5
	ACE	`ci_v2`	**93.0**	90.7	94.1	**28.0**
Qwen3.5-9B	TCE	–	14.0	11.6	10.0	6.0
	ACE	`ci_short`	94.0	95.3	**100.0**	29.0
	ACE	`ci_v1`	94.5	95.3	**100.0**	**30.5**
	ACE	`ci_v2`	**95.0**	**97.7**	**100.0**	**30.5**
Qwen3.5-27B	TCE	–	11.0	18.6	54.5	8.0
	ACE	`ci_short`	**96.0**	**97.7**	97.0	**36.5**
	ACE	`ci_v1`	93.0	88.4	**100.0**	29.0
	ACE	`ci_v2`	92.0	93.0	97.0	31.5

making context encoding more usable with off-the-shelf LLMs. Larger LLMs do not necessarily perform better, which is plausible given that these models were not specifically trained for TCE.

Performance also depends on the invocation prompt: `ci_short` achieves the best results in many cases, possibly because the contexts are short and the tasks are relatively simple. This advantage does not consistently extend to more challenging benchmarks such as RULER, as shown in Appendix E, highlighting the importance of appropriate prompting (Du et al., 2025). Finally, with only the final FFN serving as fast weights, both TCE and **ACE** maintain high accuracy on the Retention task, suggesting that the evaluated general capabilities remain largely intact after encoding.

Table 2 reports results separately for the Counterfactual, Dispute, and Outdated Information categories of Conflict. We further reported two-sided accuracy for the Counterfactual category, where an instance was scored as correct only if the LLM correctly answered both queries: one using the encoded context and the other explicitly asking it to disregard that context and use pre-existing knowledge instead. Interestingly, TCE sometimes performed better on Conflict than on the tasks in Table 1, and **ACE** even achieved near-perfect accuracy in several settings. This suggests potential for resolving knowledge conflicts, though further validation is needed. However, both methods performed worse on two-sided Counterfactual, which requires query-dependent source selection, but **ACE** still substantially

Table 3: Comparison between TCE and **ACE** on RULER, LONGBENCH, and LONGBENCH V2. We reported retrieval accuracy (%) on RULER, answer F1 (×100) on LONGBENCH, and summed judge-assigned scores over 100 instances on LONGBENCH V2. By default, we use `ci_v2`. Gold Standard refers to full-context inference with the context retained in the input. Bold marks the better result between TCE and **ACE**.

LLM	Method	RULER			LONGBENCH	LONGBENCH V2
		NIAH-1	NIAH-2	NIAH-3		
Qwen3.5-4B	Gold Standard	100	100	100	20.8	2.5
	TCE	98	7	0	8.8	0.0
	ACE	**100**	**35**	**15**	**9.0**	**2.0**
Qwen3.5-9B	Gold Standard	100	100	100	20.3	3.0
	TCE	88	7	1	8.2	2.5
	ACE	**100**	**17**	**27**	**9.6**	**3.5**
Qwen3.5-27B	Gold Standard	100	100	100	18.9	4.0
	TCE	85	7	0	7.9	**2.5**
	ACE	**100**	**40**	**29**	**9.8**	**2.5**
Muse-Glimmer-30B	Gold Standard	100	100	100	21.2	3.0
	TCE	27	33	8	14.0	2.0
	ACE	**89**	**43**	**63**	**19.2**	**3.0**

outperformed TCE. Taken together, these results suggest that TCE struggles to distinguish encoded context from pre-existing knowledge, while **ACE** offers a promising way to mitigate this issue. However, there remains room for improvement, which may motivate further research.

4.2 LONG-CONTEXT TASKS

Finally, we evaluated the effectiveness of **ACE** on long-context benchmarks to assess whether its benefits extend beyond sandbox settings. In addition to the Qwen3.5 family, we included Muse-Glimmer-30B (Meta, 2026) to examine the generalizability across model families. Alongside RULER (Hsieh et al., 2024a), which was also used in prior work, we considered two more realistic and challenging long-context benchmarks, LONGBENCH and LONGBENCH V2 (Bai et al., 2024; 2025). We briefly introduce them before presenting the results.

- **RULER.** We evaluated three *Needle-in-a-Haystack* (NIAH) tasks, NIAH-1, NIAH-2, and NIAH-3, from RULER, using 100 instances per task. They require the model to retrieve a target value embedded in irrelevant text, with increasing difficulty through more realistic distractors and more complex target values. In our experiments, each instance used a 1,024-token context, and we measured performance by the accuracy of retrieving the needle.
- **LongBench.** We evaluated 100 document question-answering instances on LONGBENCH, using its length-balanced version. These questions require the model to locate and use relevant information in the supplied documents. The contexts ranged from 2k to 7k tokens, and we measured answer F1, which captures token overlap between the generated and ground-truth answers.
- **LongBench v2.** We considered LONGBENCH V2 as a more challenging benchmark requiring deeper understanding and reasoning over longer and more complex contexts. We evaluated 100 instances with contexts ranging from about 10k to 15k tokens. These tasks require integrating information across documents, understanding detailed document content, or learning from examples provided in context, posing challenges even for full-context inference. GPT-5.5 (OpenAI, 2026) judged responses given the question, candidate answers, and gold answer, with method identities withheld. It assigned 1, 0.5, or 0 points to correct, borderline, or incorrect responses, respectively.

Most experimental settings and hyperparameters were the same as in the sandbox experiments, including Adam, the learning rate, and gradient clipping. However, to accommodate longer contexts, we adopted a maximum chunk size of 1,024 tokens and encoded the chunks sequentially. For example,

a $7k$-token context was divided into seven chunks. For LONGBENCH and LONGBENCH V2, we set $\tau = 8$ rather than 0.1 to limit encoding time and reduce the risk of overfitting.

Table 3 reports all results. On the relatively simple NIAH-1 task in RULER, even TCE performed well with the Qwen3.5 models. However, its accuracy dropped sharply on the harder NIAH variants, while **ACE** always improved over TCE across models and tasks. Performance nevertheless remained far from perfect on NIAH-2 and NIAH-3, demonstrating both the benefits of **ACE** and room for further improvement. On LONGBENCH and LONGBENCH V2, performance remained limited even with full-context inference, reflecting the difficulty of these tasks. Nevertheless, **ACE** improved over TCE on LONGBENCH and matched or exceeded it on LONGBENCH V2.

5 RELATED WORK

Our work falls broadly within *test-time training* (TTT) (Sun et al., 2020), which updates model weights at test time to adapt to distribution shifts, as studied in the broader domain-adaptation literature (Sugiyama et al., 2007; Ben-David et al., 2010). Recent work has extended this paradigm to LLMs (Hardt & Sun, 2024; Bansal et al., 2026; Zhu et al., 2026), typically by optimizing an autoregressive loss over the available context. However, these methods primarily adapt the model to condition more effectively on the available context while retaining the original context or its KV cache during subsequent generation, which differs from our setup.

Going beyond this setting, more recent TTT methods use weight updates to replace the input context entirely (Chen et al., 2023), closely matching our context-encoding setup for long-context inference. This setting is substantially more challenging than domain adaptation because the model must faithfully retain the information in its weights and use this implicit representation to condition subsequent inference. This lies beyond conventional TTT methods (Lu et al., 2026; Song et al., 2026), and enhanced methods reporting strong performance after context removal typically rely on task-specific meta-training (Chen et al., 2023; 2026). This makes it unclear whether the gains reflect general context encoding or adaptation to the meta-training distribution, and, to our knowledge, no ablation study has formally examined this distinction.

Beyond the autoregressive loss, prior work has explored alternative TTT objectives under the same setting, with the input context removed after encoding. One prominent direction uses KV association objectives (Sun et al., 2025; Zhang et al., 2025b; Behrouz et al., 2025a;b; Tang et al., 2026; Feng et al., 2026), which encode context as a bias for subsequent queries, guiding outputs toward values associated with relevant contextual keys. Despite their simple update rules, KV association inherently optimizes token matching rather than information encoding, potentially increasing reliance on meta-training and typically underperforming autoregressive encoding (Tandon et al., 2025; Liu et al., 2026). Another direction distills full-context behavior into a model operating without the context (Caccia et al., 2025; Zheng et al., 2026). Although less reliant on further fine-tuning, it requires representative context-query pairs spanning diverse, unpredictable user requests and optimization over substantially more test-time data, raising practical cost and efficiency concerns.

6 SUMMARY

TCE offers a promising approach to long-context inference by encoding context into model weights instead of retaining it in the input. Prior work often relies on additional task-specific meta-training, making it difficult to distinguish general context-encoding capability from adaptation to the meta-training distribution. We therefore studied TCE with only off-the-shelf LLMs. Our results highlight that encoding information into model weights does not ensure its reliable use as context, limiting the practical effectiveness of TCE. To address this limitation, we proposed **ACE**, which combines UID-conditioned encoding for explicit access to encoded information with a prompting framework that further specifies its contextual role and intended use. Experiments on controlled sandbox tasks and long-context benchmarks demonstrated improvements over classical TCE. These findings establish **ACE** as a promising step toward reliable context encoding, while highlighting substantial room for future improvement, particularly in source selection and more complex long-context reasoning.

AI USE STATEMENT

We used AI assistants (GPT and Claude) to polish the manuscript, check mathematical formulations and derivations, and review code. We also used GPT as an LLM judge for the LongBench v2 evaluation. The authors take responsibility for the correctness, originality, and integrity of the final manuscript, including all AI-assisted content.

ETHICS STATEMENT

This work studies a general machine learning methodology for context encoding in LLMs, using synthetic data and established benchmarks. We do not identify ethical concerns specific to this study beyond those generally associated with LLM research and deployment.

REPRODUCIBILITY STATEMENT

The encoding objectives and update procedure are described in Section 3. The experimental sections describe the models, evaluation tasks, metrics, and optimization settings. Appendix B provides the assumptions and proofs underlying our mathematical analysis. Additional component ablations and stopping-threshold results are reported in Appendices C–E. Our source code is available in an anonymized repository at https://anonymous.4open.science/r/Context-encoding-4D9E/ and will be fully open-sourced upon acceptance.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Chenxin An, Jun Zhang, Ming Zhong, Lei Li, Shansan Gong, Yao Luo, Jingjing Xu, and Lingpeng Kong. Why does the effective context length of llms fall short? In *International Conference on Learning Representations*, 2025.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench: A bilingual, multitask benchmark for long context understanding. In *Annual Meeting of the Association for Computational Linguistics*, 2024.

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 *Annual Meeting of the Association for Computational Linguistics*, 2025.

Rachit Bansal, Aston Zhang, Rishabh Tiwari, Lovish Madaan, Venkata Sai Surya Subramanyam Duvvuri, Devvrit Khatri, David Brandfonbrener, David Alvarez-Melis, Prajjwal Bhargava, Mihir Kale, et al. Let's (not) just put things in context: Test-time training for long-context llms. In *International Conference on Learning Representations*, 2026.

Ali Behrouz, Zeman Li, Praneeth Kacham, Majid Daliri, Yuan Deng, Peilin Zhong, Meisam Razaviyayn, and Vahab Mirrokni. Atlas: Learning to optimally memorize the context at test time. *arXiv preprint arXiv:2505.23735*, 2025a.

Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. In *Advances in Neural Information Processing Systems*, 2025b.

Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine Learning*, 79(1):151–175, 2010.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, 2020.

Lucas Caccia, Alan Ansell, Edoardo Ponti, Ivan Vulić, and Alessandro Sordoni. Training plug-n-play knowledge modules with deep context distillation. *arXiv preprint arXiv:2503.08727*, 2025.

Rujikorn Charakorn, Edoardo Cetin, Shinnosuke Uesaka, and Robert Tjarko Lange. Doc-to-lora: Learning to instantly internalize contexts. *arXiv preprint arXiv:2602.15902*, 2026.

Zeming Chen, Gail Weiss, Eric Mitchell, Asli Celikyilmaz, and Antoine Bosselut. Reckoning: reasoning through dynamic knowledge encoding. In *Advances in Neural Information Processing Systems*, 2023.

Zeming Chen, Angelika Romanou, Gail Weiss, and Antoine Bosselut. Perk: Long-context reasoning as parameter-efficient test-time learning. In *International Conference on Learning Representations*, 2026.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. Knowledge neurons in pretrained transformers. In *Annual Meeting of the Association for Computational Linguistics*, 2022.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. In *Advances in Neural Information Processing Systems*, 2022.

Yufeng Du, Minyang Tian, Srikanth Ronanki, Subendhu Rongali, Sravan Bodapati, Aram Galstyan, Azton Wells, Roy Schwartz, Eliu A Huerta, and Hao Peng. Context length alone hurts llm performance despite perfect retrieval. *arXiv preprint arXiv:2510.05381*, 2025.

Sabri Eyuboglu, Ryan Ehrlich, Simran Arora, Neel Guha, Dylan Zinsley, Emily Liu, Will Tennien, Atri Rudra, James Zou, Azalia Mirhoseini, et al. Cartridges: Lightweight and general-purpose long context representations via self-study. *arXiv preprint arXiv:2506.06266*, 2025.

Guhao Feng, Shengjie Luo, Kai Hua, Ge Zhang, Wenhao Huang, Di He, and Tianle Cai. In-place test-time training. In *International Conference on Learning Representations*, 2026.

Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. In *Conference on Empirical Methods in Natural Language Processing*, 2021.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025.

Moritz Hardt and Yu Sun. Test-time training on nearest neighbors for large language models. In *International Conference on Learning Representations*, 2024.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. Ruler: What's the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*, 2024a.

Cheng-Yu Hsieh, Yung-Sung Chuang, Chun-Liang Li, Zifeng Wang, Long Le, Abhishek Kumar, James Glass, Alexander Ratner, Chen-Yu Lee, Ranjay Krishna, et al. Found in the middle: Calibrating positional attention bias improves long context utilization. In *Findings of the Association for Computational Linguistics: ACL 2024*, 2024b.

W Ronny Huang, Steve Chien, Om Thakkar, and Rajiv Mathews. Detecting unintended memorization in language-model-fused asr. *arXiv preprint arXiv:2204.09606*, 2022.

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 *Annual Meeting of the Association for Computational Linguistics*, 2024.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, 2022.

Yuri Kuratov, Matvey Kairov, Aydar Bulatov, Ivan Rodkin, and Mikhail Burtsev. Gradmem: Learning to write context into memory with test-time gradient descent. *arXiv preprint arXiv:2603.13875*, 2026.

Junchen Liu, Sven Elflein, Or Litany, Zan Gojcic, and Ruilong Li. Test-time training with kv binding is secretly linear attention. *arXiv preprint arXiv:2602.21204*, 2026.

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.

Sidi Lu, Zhenwen Liang, Dongyang Ma, Yan Wang, Haitao Mi, and Dong Yu. Locas: Your models are principled initializers of locally-supported parametric memories. *arXiv preprint arXiv:2602.05085*, 2026.

Pratyush Maini, Michael Curtis Mozer, Hanie Sedghi, Zachary Chase Lipton, J Zico Kolter, and Chiyuan Zhang. Can neural network memorization be localized? In *International Conference on Machine Learning*, 2023.

Sadhika Malladi, Alexander Wettig, Dingli Yu, Danqi Chen, and Sanjeev Arora. A kernel-based view of language model fine-tuning. In *International Conference on Machine Learning*, 2023.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. In *Advances in Neural Information Processing Systems*, 2022.

Meta. Muse Glimmer Model Card. https://huggingface.co/meta-models/Muse-Glimmer-30B, 2026.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. Fast model editing at scale. *arXiv preprint arXiv:2110.11309*, 2021.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori B Hashimoto. s1: Simple test-time scaling. In *Conference on Empirical Methods in Natural Language Processing*, 2025.

OpenAI. Introducing GPT-5.5. https://openai.com/index/introducing-gpt-5-5/, 2026.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, 2013.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently scaling transformer inference. *Proceedings of Machine Learning and Systems*, 5:606–624, 2023.

Qwen. Qwen3.5: Towards native multimodal agents, 2026. URL https://qwen.ai/blog?id=qwen3.5.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, pp. 400–407, 1951.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Xiangchen Song, Zhenhao Chen, Lingjing Kong, Shaoan Xie, Xinshuai Dong, Guangyi Chen, and Kun Zhang. Beyond perplexity: A behavioral evaluation framework for deployment-memory claims in llm test-time training. *arXiv preprint arXiv:2607.00368*, 2026.

Masashi Sugiyama, Matthias Krauledat, and Klaus-Robert Müller. Covariate shift adaptation by importance weighted cross validation. *Journal of Machine Learning Research*, 8(5), 2007.

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *International Conference on Machine Learning*, 2020.

Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, Tatsunori Hashimoto, and Carlos Guestrin. Learning to (Learn at test time): RNNs with expressive hidden states. In *International Conference on Machine Learning*, 2025.

Arnuv Tandon, Karan Dalal, Xinhao Li, Daniel Koceja, Marcel Rød, Sam Buchanan, Xiaolong Wang, Jure Leskovec, Sanmi Koyejo, Tatsunori Hashimoto, et al. End-to-end test-time training for long context. *arXiv preprint arXiv:2512.23675*, 2025.

Bohao Tang, Zhen Qin, Yuqi Pan, Zheng Li, Pengfei Liu, and Ya Zhang. Modular ttt: Rethinking test-time training as composable modules. *arXiv preprint arXiv:2608.07110*, 2026.

Petar Veličković, Christos Perivolaropoulos, Federico Barbero, and Razvan Pascanu. Softmax is not enough (for sharp size generalisation). *arXiv preprint arXiv:2410.01104*, 2024.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, 2022.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. Longmemeval: Benchmarking chat assistants on long-term interactive memory. In *International Conference on Learning Representations*, 2025.

An Yang, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoyan Huang, Jiandong Jiang, Jianhong Tu, Jianwei Zhang, Jingren Zhou, et al. Qwen2.5-1m technical report. *arXiv preprint arXiv:2501.15383*, 2025.

Mengqi Zhang, Xiaotian Ye, Qiang Liu, Shu Wu, Pengjie Ren, and Zhumin Chen. Uncovering overfitting in large language model editing. In *International Conference on Learning Representations*, 2025a.

Tianyuan Zhang, Sai Bi, Yicong Hong, Kai Zhang, Fujun Luan, Songlin Yang, Kalyan Sunkavalli, William T Freeman, and Hao Tan. Test-time training done right. *arXiv preprint arXiv:2505.23884*, 2025b.

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. In *Advances in Neural Information Processing Systems*, 2023.

Ziyang Zheng, Zeju Li, Xiangyu Wen, Jianyuan Zhong, Junhua Huang, Lei Chen, Mingxuan Yuan, and Qiang Xu. Context distillation as latent memory management. *arXiv preprint arXiv:2605.28889*, 2026.

Meizhi Zhong, Chen Zhang, Yikun Lei, Xikai Liu, Yan Gao, Yao Hu, Kehai Chen, and Min Zhang. Understanding the rope extensions of long-context llms: An attention perspective. In *International Conference on Computational Linguistics*, 2025.

Xinyu Zhu, Zhe Xu, Xiaohan Wei, Yunchen Pu, Fei Tian, Chonglin Sun, Kaushik Rangadurai, Hua Zhi, Frank Shyu, Sandeep Pandey, et al. Self-guided test-time training for long-context llms. *arXiv preprint arXiv:2607.09415*, 2026.

A Dataset Examples and Prompt Templates

We provide examples and prompt templates to clarify the experimental setup. The dataset examples illustrate the encoded contexts, subsequent queries, and expected answers for each sandbox task and subtask. We then present the three context-invocation prompts used to guide the models in accessing and using encoded information during inference.

A.1 Sandbox Dataset Examples

To give a more concrete picture of the sandbox datasets, we provide an example for each task and subtask: Recall, Recognition, Continuation, the three Conflict categories (Counterfactual, Dispute, and Outdated Information), and finally Retention. The Outdated Information example comes from the Qwen3.5-4B dataset, with values based on the Wikidata snapshot collected on August 22, 2026. We retain the original wording "context above" in the Conflict queries, although the context itself is removed from the input after encoding.

Recall and Recognition

Context: The registry code of Korvustre Wharf is GC-483.

Recall query: What is the registry code of Korvustre Wharf? Return only the value.
Reference answer: GC-483

Recognition query: What information were you just told? Return the complete statement.
Reference fields: Entity: Korvustre Wharf; attribute: registry code; value: GC-483.

Continuation

Factual context: What is the capital of Norway?
Reference answer: Oslo

Arithmetic context: What is 26 squared?
Reference answer: 676

Subsequent request for either context: Continue the preceding conversation. It ended with an unanswered question, which is stored in the encoded context that is available to you. Recall that question and give your answer to it now. Do not reply that you lack access to it. End your reply with a line of the form "Final answer: <answer>".

Counterfactual

Context: The capital of Peru is Arequipa.

Contextual-answer query: Answer using only the context above, even if it contradicts what you believe to be true. According to that context, what is the capital of Peru? Return only the value.
Reference answer: Arequipa

Prior-knowledge query: Setting aside the current context, what is the generally known capital of Peru? Return only the value.
Reference answer: Lima

Dispute

Context: The inventor of the telephone is Antonio Meucci.

Contextual-answer query: Answer using only the context above, even if it contradicts what you believe to be true. According to that context, what is the inventor of the telephone? Return only the value.
Reference answer: Antonio Meucci

Outdated Information

Context: The head coach of FC Bayern Munich is Vincent Kompany.

Contextual-answer query: Answer using only the context above, even if it contradicts what you believe to be true. According to that context, what is the head coach of FC Bayern Munich? Return only the value.
Reference answer: Vincent Kompany

Outdated value: Thomas Tuchel

Retention

Encoded context: The registry code of Trilbane Outpost is RV-807.

Unrelated question: What is 17 multiplied by 4? Return only the number.
Reference answer: 68

Unrelated question: What is the capital of Portugal? Return only the value.
Reference answer: Lisbon

A.2 Context-Invocation Prompts

In addition to the context-invocation prompt introduced in Section 3.2, denoted by `ci_v1`, we evaluated two variants while keeping the encoding procedure and role-binding prompt fixed. The short variant, `ci_short`, uses a minimal instruction to invoke the encoded context. The extended variant, `ci_v2`, additionally instructs the model to continue an unresolved request or incomplete task in the latest chunk. We provide all three prompts below. Let i_t and s_t denote the UID and timestamp of context chunk c_t, respectively. Each prompt is followed by a blank line and the task query, and the resulting text is supplied as a user message.

Context-Invocation Prompt (`ci_short`)

Recall context $i_1, \ldots, i_T$ to continue.

Context-Invocation Prompt (`ci_v1`)

Parts of the conversation have been encoded as the context chunks, ordered by time: i_1 at s_1, . . ., i_T at s_T. Treat these chunks as prior conversational context in subsequent inference. Selectively use information relevant to the current request without reproducing unrelated details.

Context-Invocation Prompt (`ci_v2`)

Parts of the conversation have been encoded as the context chunks, ordered by time: i_1 at s_1, . . ., i_T at s_T. Treat these chunks as prior conversational context in subsequent inference. Selectively use information relevant to the current request without reproducing unrelated details. If the latest chunk i_T contains an unresolved request or incomplete task, continue from it when appropriate.

B MATHEMATICAL INSIGHTS

We analyze **ACE** from two complementary perspectives. For inference, Appendix B.1 shows how UIDs enable the model to distinguish among encoded contexts with addressable encoding. Appendix B.2 extends this analysis by showing how UIDs provide an explicit reference for selecting encoded context, thereby supporting its distinction from pre-existing knowledge. For encoding, Appendix B.3 provides a proof-of-concept analysis showing that UID conditioning can strengthen gradients and accelerate learning. Together, these results illustrate the potential benefits of UID conditioning for both generation and encoding.

B.1 CONTEXT DISTINCTION

We discuss whether conditioning on UIDs increases the probability of generating encoded context relative to generation without UIDs, thereby supporting an explicit mechanism for context distinction. We compare the TCE objectives $\mathcal{L}$ and $\mathcal{L}_{\text{A}}$, introduced in Sections 2.2 and 3.1, respectively, first at their ideal optima and then under finite encoding error.

We consider an LLM that receives contexts $c_1, \ldots, c_T$ from T distinct sources, respectively, with each context c_t assigned a unique UID i_t. We reuse the context-chunk notation, but consider independent contexts rather than chunks of a single context. This simplifies the mathematical modeling of context use: all contexts are encoded into the same model weights, and one context is selected at inference. For this analysis, we assume that the contexts are pairwise distinct and each contains B tokens. Generation distributions are defined over length-B sequences.

We evaluate generation by sampling a context according to a distribution $\pi = (\pi_1, \ldots, \pi_T)$, where $\pi_t > 0$ is the probability of selecting c_t and $\sum_{t=1}^{T} \pi_t = 1$. We assume the same context distribution during encoding and evaluation, with no distribution mismatch. Let θ_C and θ_C^{A} denote the weights after TCE without and with UID conditioning, obtained using the encoding objectives

$$\mathcal{L}(\theta) := -\sum_{t=1}^{T} \pi_t \log p_\theta(c_t), \qquad \mathcal{L}_{\text{A}}(\theta) := -\sum_{t=1}^{T} \pi_t \log p_\theta(c_t \mid i_t),$$

respectively. We suppress dependence on π, $c_{1:T}$, and $i_{1:T}$, and write $p(c) := p_{\theta_C}(c)$ and $p_{\text{A}}(c \mid i_t) := p_{\theta_C^{\text{A}}}(c \mid i_t)$ for brevity. The same objectives measure generation performance after encoding and can be expressed as

$$\mathcal{L}(p) := -\sum_{t=1}^{T} \pi_t \log p(c_t), \qquad \mathcal{L}_{\text{A}}(p_{\text{A}}) := -\sum_{t=1}^{T} \pi_t \log p_{\text{A}}(c_t \mid i_t).$$

These objectives measure the expected negative log-likelihood of generating a target context sampled according to π. Lower values indicate better fit to the target contexts, and $\mathcal{L}(p) - \mathcal{L}_{\text{A}}(p_{\text{A}})$ quantifies the generation benefit of UID-based context selection.

We first consider the ideal case in which both objectives are minimized over unrestricted distributions.

Proposition B.1. *Over unrestricted distributions, the optimal generation probabilities satisfy*

$$p^*(c_t) = \pi_t, \qquad p^*_{\text{A}}(c_t \mid i_t) = 1$$

for every t. Consequently,

$$\frac{p^*_{\text{A}}(c_t \mid i_t)}{p^*(c_t)} = \frac{1}{\pi_t}, \qquad \inf_{p} \mathcal{L}(p) - \inf_{p_{\text{A}}} \mathcal{L}_{\text{A}}(p_{\text{A}}) = H(\pi),$$

where $H(\pi) := -\sum_{t=1}^{T} \pi_t \log \pi_t$ is the entropy of the context-sampling distribution. For uniform weights, the probability ratio is T for every context, and the optimal loss gap is $\log T$.

Proof. Define the weighted empirical distribution over contexts by

$$\bar{q}(c) := \sum_{t=1}^{T} \pi_t \mathbf{1}\{c = c_t\}.$$

Since the contexts are pairwise distinct, $\bar{q}(c_t) = \pi_t$ for every t. The cross-entropy decomposition gives

$$D_{\mathrm{KL}}(\bar{q}\|p) = \sum_{t=1}^{T} \pi_t \log \frac{\pi_t}{p(c_t)} = \mathcal{L}(p) - H(\pi).$$

By the nonnegativity of KL divergence,

$$\mathcal{L}(p) = H(\pi) + D_{\mathrm{KL}}(\bar{q}\|p) \geq H(\pi).$$

The lower bound is attained at $p^* = \bar{q}$, yielding $p^*(c_t) = \pi_t$ and $\inf_p \mathcal{L}(p) = H(\pi)$.

The UID-conditioned objective is nonnegative. Because the UIDs are unique, each conditional distribution can assign probability one to its corresponding context:

$$p_{\mathrm{A}}^*(c \mid i_t) = \mathbf{1}\{c = c_t\}.$$

This gives $\mathcal{L}_{\mathrm{A}}(p_{\mathrm{A}}^*) = 0$. Conversely, since every $\pi_t > 0$, zero loss requires $p_{\mathrm{A}}^*(c_t \mid i_t) = 1$ for every t. Combining the two optima gives the stated probability ratio and loss gap. For uniform weights, $\pi_t = 1/T$ and $H(\pi) = \log T$. □

Remark. UIDs provide an explicit input signal that identifies which encoded context is being requested. Without UID conditioning, the model must allocate probability across the encoded contexts, whereas conditioning on each UID allows it to assign probability one to the corresponding context.

Proposition B.1 discusses the case in which both encoding objectives reach their optima. In practice, encoding may leave residual error. We therefore examine whether UID conditioning still provides an average generation advantage when neither model is assumed optimal. The following result shows that this advantage is guaranteed when the UID-conditioned encoding error is sufficiently small.

Proposition B.2. *Suppose $p(c_t) > 0$ and $p_{\mathrm{A}}(c_t \mid i_t) > 0$ for every $t = 1, \ldots, T$. Let $\epsilon_{\mathrm{A}} := \mathcal{L}_{\mathrm{A}}(p_{\mathrm{A}})$ denote the final UID-conditioned encoding error. Then*

$$\mathcal{L}(p) - \mathcal{L}_{\mathrm{A}}(p_{\mathrm{A}}) \geq H(\pi) - \epsilon_{\mathrm{A}}.$$

Equivalently,

$$\prod_{t=1}^{T} \left[\frac{p_{\mathrm{A}}(c_t \mid i_t)}{p(c_t)}\right]^{\pi_t} \geq \exp\big(H(\pi) - \epsilon_{\mathrm{A}}\big).$$

Thus, $\epsilon_{\mathrm{A}} < H(\pi)$ guarantees a positive average log-probability gain. For uniform weights, the lower bounds are $\log T - \epsilon_{\mathrm{A}}$ and $T\exp(-\epsilon_{\mathrm{A}})$, respectively, and the sufficient condition becomes $\epsilon_{\mathrm{A}} < \log T$.

Proof. Using the cross-entropy decomposition from the proof of Proposition B.1, we obtain

$$\mathcal{L}(p) - \mathcal{L}_{\mathrm{A}}(p_{\mathrm{A}}) = H(\pi) + D_{\mathrm{KL}}(\bar{q}\|p) - \epsilon_{\mathrm{A}} \geq H(\pi) - \epsilon_{\mathrm{A}},$$

where the inequality follows from the nonnegativity of KL divergence.

By the definitions of the objectives,

$$\mathcal{L}(p) - \mathcal{L}_{\mathrm{A}}(p_{\mathrm{A}}) = \sum_{t=1}^{T} \pi_t \log \frac{p_{\mathrm{A}}(c_t \mid i_t)}{p(c_t)} = \log \prod_{t=1}^{T} \left[\frac{p_{\mathrm{A}}(c_t \mid i_t)}{p(c_t)}\right]^{\pi_t}.$$

Exponentiating the loss-gap bound yields the probability-ratio bound. If $\epsilon_{\mathrm{A}} < H(\pi)$, the loss gap is strictly positive. Substituting $H(\pi) = \log T$ gives the uniform-weight results. □

Remark. The sufficient condition does not require perfect encoding: a nonzero final encoding error below $H(\pi)$ suffices to guarantee an average generation advantage. This guarantee concerns selection among multiple encoded contexts; for $T = 1$, the ideal loss gap is zero.

B.2 Source Distinction

We extend Proposition B.2 to include pre-existing knowledge as an additional source, examining whether UID conditioning helps distinguish encoded context from pre-existing knowledge. Let $\widetilde{C}$ denote a length-B target sequence with its realization $\tilde{c}$, and let $Z \in \{0, 1, \ldots, T\}$ denote its source. Here, $Z = 0$ represents pre-existing knowledge, while $Z = t$ represents encoded context c_t. With $\alpha \in (0, 1)$ denoting the probability of selecting pre-existing knowledge as the target source, we define

$$\Pr(Z = 0) = \alpha, \qquad \Pr(Z = t) = (1 - \alpha)\pi_t, \quad t = 1, \ldots, T.$$

Let q_K denote a fixed empirical distribution over length-B sequences expressing pre-existing knowledge. The target distributions are

$$q(\tilde{c} \mid Z = 0) = q_K(\tilde{c}), \qquad q(\tilde{c} \mid Z = t) = \mathbf{1}\{\tilde{c} = c_t\}.$$

Together with the source probabilities, these define a joint distribution q over $(\widetilde{C}, Z)$, with marginal

$$q(\tilde{c}) = \alpha q_K(\tilde{c}) + (1 - \alpha)\bar{q}(\tilde{c}),$$

where $\bar{q}$ is the weighted empirical distribution over encoded contexts defined in Appendix B.1.

We retain the notation p and p_{A} from Appendix B.1. For encoded contexts, the UID-conditioned model receives the corresponding UID i_t. For targets from pre-existing knowledge, it receives no UID. For brevity, we write $p_{\mathrm{A}}(\tilde{c}) := p_{\theta^{\mathrm{A}}_{\widetilde{C}}}(\tilde{c})$. Thus, $p_{\mathrm{A}}(\tilde{c})$ and $p_{\mathrm{A}}(\tilde{c} \mid i_t)$ refer to the same UID-conditioned model without and with a UID, respectively, whereas $p(\tilde{c})$ refers to the baseline encoded model.

The corresponding weighted negative log-likelihood objectives are

$$\mathcal{J}(p) := -\alpha \sum_{\tilde{c}: q_K(\tilde{c}) > 0} q_K(\tilde{c}) \log p(\tilde{c}) + (1 - \alpha)\mathcal{L}(p),$$

$$\mathcal{J}_{\mathrm{A}}(p_{\mathrm{A}}) := -\alpha \sum_{\tilde{c}: q_K(\tilde{c}) > 0} q_K(\tilde{c}) \log p_{\mathrm{A}}(\tilde{c}) + (1 - \alpha)\mathcal{L}_{\mathrm{A}}(p_{\mathrm{A}}).$$

These objectives evaluate generation from pre-existing knowledge without a UID and from encoded context with or without its UID, according to the encoding method.

Proposition B.3. *Suppose $p(\tilde{c}) > 0$ whenever $q(\tilde{c}) > 0$, $p_{\mathrm{A}}(\tilde{c}) > 0$ whenever $q_K(\tilde{c}) > 0$, and $p_{\mathrm{A}}(c_t \mid i_t) > 0$ for every t. Define the excess loss as*

$$\epsilon^{\mathrm{src}}_{\mathrm{A}} := \mathcal{J}_{\mathrm{A}}(p_{\mathrm{A}}) - H(\widetilde{C} \mid Z).$$

Then

$$\mathcal{J}(p) - \mathcal{J}_{\mathrm{A}}(p_{\mathrm{A}}) \geq I(\widetilde{C}; Z) - \epsilon^{\mathrm{src}}_{\mathrm{A}},$$

where $I(\widetilde{C}; Z)$ is the mutual information between the target sequence and its source under q. Equivalently,

$$\prod_{\tilde{c}: q_K(\tilde{c}) > 0} \left[\frac{p_{\mathrm{A}}(\tilde{c})}{p(\tilde{c})}\right]^{\alpha q_K(\tilde{c})} \prod_{t=1}^{T} \left[\frac{p_{\mathrm{A}}(c_t \mid i_t)}{p(c_t)}\right]^{(1-\alpha)\pi_t} \geq \exp\big(I(\widetilde{C}; Z) - \epsilon^{\mathrm{src}}_{\mathrm{A}}\big).$$

Thus, $\epsilon^{\mathrm{src}}_{\mathrm{A}} < I(\widetilde{C}; Z)$ is sufficient for a positive average log-probability gain under this evaluation.

Proof. Since $\mathcal{J}(p) = -\sum_{\tilde{c}: q(\tilde{c}) > 0} q(\tilde{c}) \log p(\tilde{c})$, the cross-entropy decomposition gives

$$\mathcal{J}(p) = H(\widetilde{C}) + D_{\mathrm{KL}}(q \| p),$$

where q in the KL divergence denotes the marginal distribution of $\widetilde{C}$. Since the target context is deterministic given $Z = t$ for every $t \geq 1$, we have

$$H(\widetilde{C} \mid Z) = \alpha H(q_K),$$

where $H(q_K) := -\sum_{\tilde{c}:q_K(\tilde{c})>0} q_K(\tilde{c}) \log q_K(\tilde{c})$. Consequently,

$$\epsilon_{\mathrm{A}}^{\mathrm{src}} = \mathcal{J}_{\mathrm{A}}(p_{\mathrm{A}}) - \alpha H(q_K) = \alpha D_{\mathrm{KL}}(q_K \| p_{\mathrm{A}}) + (1-\alpha)\mathcal{L}_{\mathrm{A}}(p_{\mathrm{A}}) \geq 0,$$

where p_{A} in the KL divergence denotes the model distribution without a UID.

Substituting the definition of $\epsilon_{\mathrm{A}}^{\mathrm{src}}$ yields

$$\begin{aligned}\mathcal{J}(p) - \mathcal{J}_{\mathrm{A}}(p_{\mathrm{A}}) = H(\widetilde{C}) - H(\widetilde{C} \mid Z) + D_{\mathrm{KL}}(q\|p) - \epsilon_{\mathrm{A}}^{\mathrm{src}} &= I(\widetilde{C}; Z) + D_{\mathrm{KL}}(q\|p) - \epsilon_{\mathrm{A}}^{\mathrm{src}} \\ &\geq I(\widetilde{C}; Z) - \epsilon_{\mathrm{A}}^{\mathrm{src}}.\end{aligned}$$

By the definitions of the objectives,

$$\mathcal{J}(p) - \mathcal{J}_{\mathrm{A}}(p_{\mathrm{A}}) = \alpha \sum_{\tilde{c}:q_K(\tilde{c})>0} q_K(\tilde{c}) \log \frac{p_{\mathrm{A}}(\tilde{c})}{p(\tilde{c})} \quad + (1-\alpha) \sum_{t=1}^{T} \pi_t \log \frac{p_{\mathrm{A}}(c_t \mid i_t)}{p(c_t)}.$$

Exponentiating the loss-gap bound therefore yields the probability-ratio bound. If $\epsilon_{\mathrm{A}}^{\mathrm{src}} < I(\widetilde{C}; Z)$, the loss gap is strictly positive, establishing the average log-probability advantage. □

Remark. The bound characterizes the average generation benefit of providing UIDs for encoded contexts while using no UID for pre-existing knowledge. The excess loss combines two errors:

$$\epsilon_{\mathrm{A}}^{\mathrm{src}} = \alpha D_{\mathrm{KL}}(q_K \| p_{\mathrm{A}}) + (1-\alpha)\epsilon_{\mathrm{A}}.$$

If the model remains well fitted to pre-existing knowledge after encoding, the first term is small. Provided this term is below $I(\widetilde{C}; Z)$, sufficiently low UID-conditioned encoding error guarantees an average generation advantage under the specified evaluation. The information term includes both distinctions among encoded contexts and their distinction from pre-existing knowledge. However, the bound does not guarantee reliable source selection for arbitrary downstream requests.

B.3 Encoding Benefits

The preceding analysis characterizes the generation benefit of addressable context. We now examine its encoding benefits. Prior empirical work associates stronger gradient contributions with stronger memorization (Huang et al., 2022; Maini et al., 2023). This motivates us to examine whether UID conditioning produces larger gradients and whether these lead to faster loss reduction.

We focus on encoding the n-th token $c_{t,n}$ of context chunk c_t, given its preceding tokens, where $n \geq 2$. We consider a simplified single-head causal attention model that predicts $c_{t,n}$ from the output at the final prefix position. Let $a_1, \ldots, a_{n-1} \in \mathbb{R}^d$ denote fixed representations of the preceding tokens. Without UID conditioning, their attention-weighted representation is

$$h := \sum_{k=1}^{n-1} \omega_k a_k, \qquad \omega_k > 0, \qquad \sum_{k=1}^{n-1} \omega_k = 1.$$

Here, ω_k is the attention weight assigned to the k-th preceding token when predicting $c_{t,n}$.

We represent the UID i_t by a single token with fixed representation $v \in \mathbb{R}^d$. When the UID is prepended, we assume that it receives attention weight $\beta \in (0,1)$ while the relative attention weights among the original tokens remain unchanged. The aggregated representation then becomes

$$h_{\mathrm{A}} := (1-\beta)h + \beta v.$$

This corresponds to adding the UID to softmax attention without changing the attention scores of the original tokens. For simplicity, we hold the representations and attention weights fixed throughout training.

The outputs without and with UID conditioning are

$$z := W_V h, \qquad z_{\mathrm{A}} := W_V^{\mathrm{A}} h_{\mathrm{A}},$$

where $W_V, W_V^{\mathrm{A}} \in \mathbb{R}^{M \times d}$ are the corresponding trainable value projection matrices, and $M \geq 2$ is the vocabulary size. For simplicity, we take these outputs directly as vocabulary logits. The token-level negative log-likelihoods are

$$\ell(W_V) := -\log[\mathrm{softmax}(z)]_{c_{t,n}}, \qquad \ell_{\mathrm{A}}(W_V^{\mathrm{A}}) := -\log[\mathrm{softmax}(z_{\mathrm{A}})]_{c_{t,n}}.$$

We update only the value projection matrices and compare their gradients and loss dynamics.

We identify sufficient conditions under which UID conditioning increases the norm of the aggregated representation, and then relate this increase to larger gradients and faster loss reduction.

Proposition B.4. *Suppose*

$$\langle h, v\rangle = 0, \qquad 0 < r := \frac{\|h\|}{\|v\|} < 1.$$

Then

$$\gamma := \frac{\|h_{\mathrm{A}}\|^2}{\|h\|^2} = (1-\beta)^2 + \frac{\beta^2}{r^2}.$$

If, in addition,

$$\beta > \frac{2r^2}{1+r^2},$$

then $\gamma > 1$. *At identical prediction distributions, i.e.,* $\mathrm{softmax}(z_{\mathrm{A}}) = \mathrm{softmax}(z)$, *the gradients satisfy*

$$\|\nabla_{W_V^{\mathrm{A}}} \ell_{\mathrm{A}}\|_F^2 = \gamma \|\nabla_{W_V} \ell\|_F^2 > \|\nabla_{W_V} \ell\|_F^2.$$

Under these conditions, consider training both matrices by gradient flow:

$$\frac{dW_V}{ds} = -\nabla_{W_V} \ell, \qquad \frac{dW_V^{\mathrm{A}}}{ds} = -\nabla_{W_V^{\mathrm{A}}} \ell_{\mathrm{A}},$$

where $s \geq 0$ *denotes training time. If the initial logits are finite and satisfy* $z_{\mathrm{A}}(0) = z(0)$, *then*

$$z_{\mathrm{A}}(s) = z(\gamma s).$$

Consequently, for every $s > 0$,

$$\ell_{\mathrm{A}}(W_V^{\mathrm{A}}(s)) = \ell(W_V(\gamma s)) < \ell(W_V(s)).$$

Thus, UID conditioning accelerates the same loss trajectory by a factor of γ.

Proof. Since $h_{\mathrm{A}} = (1-\beta)h + \beta v$ and $\langle h, v\rangle = 0$,

$$\|h_{\mathrm{A}}\|^2 = (1-\beta)^2 \|h\|^2 + \beta^2 \|v\|^2.$$

Dividing by $\|h\|^2$ gives

$$\gamma = (1-\beta)^2 + \frac{\beta^2}{r^2}.$$

Under the assumed lower bound on β,

$$\gamma - 1 = \frac{\beta}{r^2}\left[\beta(1+r^2) - 2r^2\right] > 0.$$

Let $e_{c_{t,n}} \in \mathbb{R}^M$ denote the one-hot vector for the target token. Differentiating the losses gives

$$\nabla_{W_V} \ell = \big(\mathrm{softmax}(z) - e_{c_{t,n}}\big) h^\top,$$

$$\nabla_{W_V^{\mathrm{A}}} \ell_{\mathrm{A}} = \big(\mathrm{softmax}(z_{\mathrm{A}}) - e_{c_{t,n}}\big) h_{\mathrm{A}}^\top.$$

Using $\|ab^\top\|_F^2 = \|a\|^2 \|b\|^2$, we obtain

$$\|\nabla_{W_V} \ell\|_F^2 = \|\,\mathrm{softmax}(z) - e_{c_{t,n}}\|^2 \|h\|^2,$$

$$\|\nabla_{W_V^{\mathrm{A}}} \ell_{\mathrm{A}}\|_F^2 = \|\,\mathrm{softmax}(z_{\mathrm{A}}) - e_{c_{t,n}}\|^2 \|h_{\mathrm{A}}\|^2.$$

At identical prediction distributions, the prediction-error factors coincide. Hence,

$$\|\nabla_{W_V^{\mathrm{A}}} \ell_{\mathrm{A}}\|_F^2 = \gamma \|\nabla_{W_V} \ell\|_F^2.$$

The baseline gradient is nonzero because $h \neq 0$ and, for $M \geq 2$, finite logits cannot yield a one-hot softmax distribution. Together with $\gamma > 1$, this establishes the strict gradient inequality.

Under gradient flow,

$$\frac{d\ell}{ds} = -\|\nabla_{W_V}\ell\|_F^2, \qquad \frac{d\ell_{\mathrm{A}}}{ds} = -\|\nabla_{W_V^{\mathrm{A}}}\ell_{\mathrm{A}}\|_F^2.$$

Thus, at identical predictions, the larger squared gradient norm yields a loss-decrease rate that is γ times as large.

To compare the full trajectories, recall that h and h_{A} are fixed. The logit dynamics are

$$\frac{dz}{ds} = \frac{dW_V}{ds}h = -\|h\|^2\big(\mathrm{softmax}(z) - e_{c_{t,n}}\big),$$

$$\frac{dz_{\mathrm{A}}}{ds} = -\|h_{\mathrm{A}}\|^2\big(\mathrm{softmax}(z_{\mathrm{A}}) - e_{c_{t,n}}\big).$$

Since $\|h_{\mathrm{A}}\|^2 = \gamma\|h\|^2$, the rescaled trajectory $z(\gamma s)$ satisfies the same differential equation as $z_{\mathrm{A}}(s)$. The softmax vector field is globally Lipschitz, so identical initial logits imply

$$z_{\mathrm{A}}(s) = z(\gamma s).$$

Therefore,

$$\ell_{\mathrm{A}}(W_V^{\mathrm{A}}(s)) = \ell(W_V(\gamma s)).$$

The logit derivatives are bounded, so finite initial logits remain finite at every finite time. The baseline gradient therefore remains nonzero, implying $d\ell/ds < 0$. Since $\gamma > 1$, for every $s > 0$,

$$\ell(W_V(\gamma s)) < \ell(W_V(s)),$$

which completes the proof. □

Remark. Under the stated conditions, UID conditioning produces larger gradients and faster loss reduction, suggesting a potential memorization benefit in light of prior empirical findings. These conditions require the UID representation to be distinct from the context and exert sufficient influence on the prediction. The result is a proof of concept under simplified training dynamics, rather than a guarantee for full LLM training.

Table 4: Comparison between TCE, TCE-AM, and ACE on Recall and Recognition. We reported accuracy (%) on the Qwen3.5 family. Bold marks the best result among TCE and TCE-AM within each model and task.

LLM	Method	Cue	Recall	Recognition
Qwen3.5-4B	TCE	–	6.1	0.5
	TCE-AM	–	**7.1**	10.7
	TCE-AM	UID	5.6	**35.5**
	ACE	`ci_short`	52.3	91.9
Qwen3.5-9B	TCE	–	**4.6**	2.0
	TCE-AM	–	1.5	3.6
	TCE-AM	UID	0.5	**11.2**
	ACE	`ci_short`	40.1	96.4
Qwen3.5-27B	TCE	–	1.5	0.0
	TCE-AM	–	1.5	0.0
	TCE-AM	UID	**4.6**	**1.0**
	ACE	`ci_short`	75.6	98.0

C Ablation Studies on Sandbox Tasks

Now, we conducted ablations on the sandbox benchmark, focusing on the relatively challenging Recall, Recognition, and Continuation tasks, to examine the contribution of each design choice.

C.1 Effect of the Addressing Mechanism

We first compared TCE augmented with the addressing mechanism, termed TCE-AM for simplicity, against the original TCE. Table 4 compares two variants of TCE-AM: one follows Section 3.1, using Eq. (6) for encoding and Eq. (7) for inference (Cue: UID); the other uses the same encoding objective but omits the UID at inference, following Eq. (5) as in TCE (Cue: –). Overall, TCE-AM outperformed TCE, with some exceptions on Recall. Even without the UID at inference, TCE-AM improved Recognition on 4B and 9B, although its effect on Recall was mixed. This is consistent with our theoretical analysis in Appendix B.3, which suggests that UID-conditioned encoding has the potential to strengthen memorization. However, as discussed in Section 3.2, the intended use of the encoded information remains implicit, leaving context use unreliable. By further incorporating the prompting framework, ACE achieves substantially better performance.

C.2 Contributions of Encoding and Invocation

Table 5 reports additional ablations of individual components and their combinations. As shown, removing either $\mathcal{L}_{\mathrm{A}}$ or $\mathcal{L}_{\mathrm{B}}$ during encoding reduced performance, highlighting their complementary contributions. Without encoding, ACE scored zero on both tasks, confirming that the invocation prompt alone is insufficient without context-specific weight updates. Using the original TCE inference in Eq. (5) after ACE encoding still achieved strong performance in some cases, suggesting that encoded information can remain accessible without an explicit invocation prompt, although the full method performed better. Together, these ablations support the rationale behind our design choices and demonstrate the contribution of each component to reliable context encoding.

C.3 Choice of Fast Weights

We compared FFN and attention updates on Qwen3.5-4B and Qwen3.5-9B in Tables 6-7, also examining updates to multiple layers beyond the default final-FFN setting. Here, FFN-k and ATTN-k denote updates to the FFN and attention sublayers in the final k transformer layers, respectively, for $k \in \{1, 2, 3\}$. For ACE, updating more FFN layers improved Recognition but had mixed effects on other tasks. ATTN-2 and ATTN-3 outperformed all FFN configurations on Recall and Continuation across both models. In particular, ATTN-3 increased Continuation accuracy from 48.8% to 72.0%

Table 5: Ablation of **ACE** on Recall and Recognition. We reported accuracy (%) on the Qwen3.5 family. Bold marks the best result within each model and task.

LLM	Method	Recall	Recognition
Qwen3.5-4B	**ACE** w/o $\mathcal{L}_{\text{A}}$	4.1	21.3
	ACE w/o $\mathcal{L}_{\text{B}}$	37.1	49.7
	ACE with TCE inference	32.5	81.2
	ACE w/o encoding	0.0	0.0
	ACE	**55.3**	**81.7**
Qwen3.5-9B	**ACE** w/o $\mathcal{L}_{\text{A}}$	7.6	17.3
	ACE w/o $\mathcal{L}_{\text{B}}$	22.8	47.2
	ACE with TCE inference	19.8	68.0
	ACE w/o encoding	0.0	0.0
	ACE	**35.5**	**95.4**
Qwen3.5-27B	**ACE** w/o $\mathcal{L}_{\text{A}}$	22.3	0.5
	ACE w/o $\mathcal{L}_{\text{B}}$	31.5	13.2
	ACE with TCE inference	18.3	20.8
	ACE w/o encoding	0.0	0.0
	ACE	**39.1**	**58.4**

Table 6: Comparison between TCE and **ACE** with different choices of fast weights on Qwen3.5-4B. FFN-k and ATTN-k denote the FFN and attention sublayers of the final k transformer layers, respectively. We reported accuracy (%) with **ACE** using `ci_v1`. Bold marks the best result within each method and task.

Method	Fast Weights	Recall	Recognition	Continuation	Retention	Counterfact	Dispute
TCE	FFN-1	6.1	0.5	0.0	**100.0**	3.5	**7.0**
	FFN-2	4.1	**7.6**	0.0	68.0	3.0	**7.0**
	FFN-3	9.1	4.1	**3.6**	43.0	**9.5**	4.7
	ATTN-1	5.6	0.0	0.0	84.0	0.5	2.3
	ATTN-2	**11.7**	0.0	0.0	74.0	1.0	2.3
	ATTN-3	9.6	0.0	0.0	69.0	2.5	2.3
ACE	FFN-1	55.3	81.7	48.8	**100.0**	92.5	95.3
	FFN-2	63.5	91.9	44.0	98.0	93.0	93.0
	FFN-3	55.3	89.8	52.4	95.0	87.0	90.7
	ATTN-1	49.2	42.6	38.1	**100.0**	93.5	**100.0**
	ATTN-2	**65.0**	88.8	70.2	**100.0**	**96.5**	97.7
	ATTN-3	64.5	**92.9**	**72.0**	**100.0**	95.5	93.0

on 4B and from 13.7% to 68.5% on 9B, relative to FFN-1. TCE benefited from different choices: attention updates yielded its highest Recall accuracy but zero Recognition accuracy, whereas FFN-3 performed best on Continuation. Retention remained at 100% across configurations on 9B but varied on 4B, where **ACE** was substantially more stable than TCE. These results highlight the benefits of exploring fast-weight configurations beyond FFN-1, with the best choice depending on the model, method, and task. A systematic investigation of these choices lies beyond the scope of this work and remains a promising direction for future research.

Table 7: Comparison between TCE and **ACE** with different choices of fast weights on Qwen3.5-9B. FFN-k and ATTN-k denote the FFN and attention sublayers of the final k transformer layers, respectively. We reported accuracy (%) with **ACE** using `ci_v1`. Bold marks the best result within each method and task.

Method	Fast Weights	Recall	Recognition	Continuation	Retention	Counterfact	Dispute
TCE	FFN-1	4.6	**2.0**	1.2	**100.0**	14.0	11.6
	FFN-2	10.2	0.5	26.2	**100.0**	26.5	11.6
	FFN-3	17.3	0.5	**26.8**	**100.0**	10.5	7.0
	ATTN-1	14.2	0.0	0.6	**100.0**	19.5	14.0
	ATTN-2	17.8	0.0	2.4	**100.0**	31.0	14.0
	ATTN-3	**23.9**	0.0	2.4	**100.0**	**35.5**	**16.3**
ACE	FFN-1	35.5	95.4	13.7	**100.0**	94.5	95.3
	FFN-2	47.2	**99.5**	10.7	**100.0**	96.5	93.0
	FFN-3	52.8	**99.5**	13.7	**100.0**	**97.0**	90.7
	ATTN-1	55.3	76.6	59.5	**100.0**	91.0	**97.7**
	ATTN-2	**63.5**	98.5	67.3	**100.0**	96.0	**97.7**
	ATTN-3	62.9	97.5	**68.5**	**100.0**	96.5	**97.7**

Table 8: Comparison between TCE and ACE on Recall, Recognition, Continuation, and Retention across stopping thresholds. We reported accuracy (%) on Qwen3.5-4B. Bold marks the best result within each threshold and task.

τ	Method	Prompt	Recall	Recognition	Continuation	Retention
0.1	TCE	–	6.1	0.5	0.0	**100.0**
	ACE	ci_short	52.3	**91.9**	46.4	99.0
	ACE	ci_v1	55.3	81.7	**48.8**	**100.0**
	ACE	ci_v2	**60.4**	80.7	28.0	**100.0**
0.2	TCE	–	6.6	0.0	0.0	**100.0**
	ACE	ci_short	51.3	**86.3**	45.8	99.0
	ACE	ci_v1	51.8	72.6	**48.2**	**100.0**
	ACE	ci_v2	**57.9**	71.6	28.6	**100.0**
1.0	TCE	–	4.1	0.0	0.0	**100.0**
	ACE	ci_short	38.1	**41.6**	38.1	98.0
	ACE	ci_v1	39.1	16.2	**47.6**	**100.0**
	ACE	ci_v2	**42.1**	11.2	26.8	**100.0**

D More Results on Sandbox Tasks

We provide additional sandbox results examining the effects of encoding thresholds, update budgets, invocation prompts, UID design, and encoding wording. Selected analyses also include RULER NIAH-2 as a complementary retrieval task.

D.1 Sensitivity to the Encoding Threshold

Tables 8–13 report the complete sandbox results across stopping thresholds, extending the main comparisons at $\tau = 0.1$. Increasing τ generally reduced Recall and Recognition accuracy, particularly the latter, whereas its effect on Continuation varied across models and prompts. In contrast, two-sided Counterfactual accuracy tended to improve at larger τ. Continuation accuracy measures whether the model correctly answered the encoded question. Table 14 additionally reports question-recovery rates, providing a lower bound on retrieval without requiring the model to answer the recovered question. Table 15 reports accuracy on non-conflicting control queries paired with Dispute and Counterfactual instances. These controls assess performance on information that should remain unaffected by encoding and are excluded from the primary Conflict scores.

D.2 Comparison at Fixed Update Budgets

We compared TCE and ACE with exactly $K \in \{2, 4, 8, 16, 32\}$ updates per chunk, replacing adaptive stopping while retaining their encoding objectives and inference procedures, with ACE using `ci_v1`. We used the same 197 sandbox instances for Recall and Recognition and 100 RULER NIAH-2 instances on Qwen3.5-4B, 9B, and 27B. This matched update counts rather than computational costs, as each ACE update used two copies of the context. Tables 16–18 show that ACE achieved higher accuracy in 35 of 36 comparisons at $K \geq 4$, with one tie. Recognition reached 98.5–99.5% by $K = 8$ with ACE. Although additional updates improved TCE on Recognition and generally improved retrieval on NIAH-2, TCE remained below ACE at larger budgets. These results suggest that the gains of ACE were not solely due to additional encoding updates.

D.3 Effect of Invocation on Prior-Knowledge Recovery

We examined whether the context-invocation prompt affected the ability of ACE to set aside encoded information and answer from pre-existing knowledge. Using the same 200 Counterfactual instances as in the main sandbox experiments, we encoded each context with $\tau = 0.1$ and evaluated the prior-knowledge query with and without `ci_short`, keeping the query and encoded model weights unchanged. Table 19 reports the results alongside contextual-answer accuracy for TCE and ACE.

Table 9: Comparison between TCE and **ACE** on the three categories of Conflict across stopping thresholds. We reported accuracy (%) on Qwen3.5-4B, together with two-sided Counterfactual accuracy. Bold marks the best result within each threshold and metric.

τ	Method	Prompt	Counterfact	Dispute	Outdated	Counterfact (two-sided)
0.1	TCE	–	3.5	7.0	5.9	0.5
	ACE	`ci_short`	86.5	83.7	88.2	20.0
	ACE	`ci_v1`	92.5	**95.3**	**100.0**	27.5
	ACE	`ci_v2`	**93.0**	90.7	94.1	**28.0**
0.2	TCE	–	4.5	4.7	5.9	1.5
	ACE	`ci_short`	86.0	79.1	88.2	21.5
	ACE	`ci_v1`	**92.5**	**95.3**	**100.0**	**29.0**
	ACE	`ci_v2`	**92.5**	90.7	94.1	28.5
1.0	TCE	–	13.5	4.7	23.5	7.0
	ACE	`ci_short`	85.5	69.8	88.2	31.0
	ACE	`ci_v1`	**93.5**	**90.7**	**100.0**	41.0
	ACE	`ci_v2`	92.5	86.0	94.1	**41.5**

Table 10: Comparison between TCE and **ACE** on Recall, Recognition, Continuation, and Retention across stopping thresholds. We reported accuracy (%) on Qwen3.5-9B. Bold marks the best result within each threshold and task.

τ	Method	Prompt	Recall	Recognition	Continuation	Retention
0.1	TCE	–	4.6	2.0	1.2	**100.0**
	ACE	`ci_short`	**40.1**	**96.4**	9.5	**100.0**
	ACE	`ci_v1`	35.5	95.4	**13.7**	**100.0**
	ACE	`ci_v2`	38.1	94.4	5.4	**100.0**
0.2	TCE	–	4.1	0.5	0.6	**100.0**
	ACE	`ci_short`	**37.1**	**86.8**	8.3	**100.0**
	ACE	`ci_v1`	32.0	86.3	**11.3**	**100.0**
	ACE	`ci_v2`	36.5	82.2	4.2	**100.0**
1.0	TCE	–	3.6	0.0	0.0	**100.0**
	ACE	`ci_short`	26.9	**56.9**	1.2	**100.0**
	ACE	`ci_v1`	25.9	42.1	**6.0**	**100.0**
	ACE	`ci_v2`	**29.4**	35.0	1.2	**100.0**

While **ACE** substantially improved contextual-answer accuracy, removing the invocation had mixed effects on prior-knowledge recovery. Accuracy increased by 6.0 percentage points on Qwen3.5-4B but decreased by 4.5 and 3.5 points on Qwen3.5-9B and Qwen3.5-27B, respectively. Thus, removing the invocation alone did not consistently improve prior-knowledge recovery, suggesting that the difficulty was not solely due to the invocation prompt.

D.4 TCE WITH GENERIC INVOCATION

We examined whether adding an invocation prompt to TCE, without changing its encoding procedure, could recover the gains of **ACE**. We retained the original autoregressive encoding objective and prepended the following generic invocation to the task query:

Generic Invocation

Recall the previously encoded context and use information relevant to the following request.

Table 11: Comparison between TCE and **ACE** on the three categories of Conflict across stopping thresholds. We reported accuracy (%) on Qwen3.5-9B, together with two-sided Counterfactual accuracy. Bold marks the best result within each threshold and metric.

τ	Method	Prompt	Counterfact	Dispute	Outdated	Counterfact (two-sided)
0.1	TCE	–	14.0	11.6	10.0	6.0
	ACE	`ci_short`	94.0	95.3	**100.0**	29.0
	ACE	`ci_v1`	94.5	95.3	**100.0**	**30.5**
	ACE	`ci_v2`	**95.0**	**97.7**	**100.0**	**30.5**
0.2	TCE	–	15.0	11.6	10.0	10.0
	ACE	`ci_short`	90.5	**95.3**	**100.0**	37.0
	ACE	`ci_v1`	91.0	93.0	95.0	37.0
	ACE	`ci_v2`	**94.5**	93.0	**100.0**	**40.0**
1.0	TCE	–	21.0	11.6	10.0	16.0
	ACE	`ci_short`	76.5	**90.7**	**100.0**	45.5
	ACE	`ci_v1`	86.0	88.4	95.0	50.0
	ACE	`ci_v2`	**91.0**	**90.7**	**100.0**	**54.0**

Table 12: Comparison between TCE and **ACE** on Recall, Recognition, Continuation, and Retention across stopping thresholds. We reported accuracy (%) on Qwen3.5-27B. Bold marks the best result within each threshold and task.

τ	Method	Prompt	Recall	Recognition	Continuation	Retention
0.1	TCE	–	1.5	0.0	0.0	**100.0**
	ACE	`ci_short`	**75.6**	**98.0**	**20.8**	**100.0**
	ACE	`ci_v1`	39.1	58.4	11.9	**100.0**
	ACE	`ci_v2`	45.2	50.8	2.4	**100.0**
0.2	TCE	–	2.0	0.0	0.0	**100.0**
	ACE	`ci_short`	**71.6**	**83.8**	0.6	**100.0**
	ACE	`ci_v1`	37.1	39.6	**29.8**	**100.0**
	ACE	`ci_v2`	44.2	28.9	1.2	**100.0**
1.0	TCE	–	1.5	0.0	0.0	**100.0**
	ACE	`ci_short`	**53.3**	**22.3**	0.6	**100.0**
	ACE	`ci_v1`	28.9	1.5	**32.1**	**100.0**
	ACE	`ci_v2`	33.0	1.0	2.4	**100.0**

This prompt instructs the model to use encoded information without specifying UIDs or timestamps. We compared TCE with and without this invocation against full **ACE** on the same 197 sandbox instances for Recall and Recognition, with an additional evaluation on 100 RULER NIAH-2 instances. We used Qwen3.5-4B, 9B, and 27B with $\tau = 0.1$. The original TCE and **ACE** results served as references, with **ACE** using `ci_v1` on the sandbox tasks and `ci_v2` on NIAH-2. Table 20 shows that adding the generic invocation had limited and inconsistent benefits. Although it improved Recall on Qwen3.5-9B and Qwen3.5-27B, Recognition accuracy remained at 0.0–1.5%, and NIAH-2 accuracy ranged from 5.0% to 10.0%. Performance remained substantially below full **ACE** across all evaluated models and tasks. These results suggest that adding a recall instruction at inference alone was insufficient to overcome the limitations of autoregressive TCE in these settings.

D.5 Unique versus Shared UIDs

We examined whether **ACE** required unique UIDs or could instead use the same UID across contexts. We replaced every UID occurrence with the fixed string `memory` in both encoding sequences and the `ci_v1` invocation, while retaining the original timestamps and all other settings. For terminological

Table 13: Comparison between TCE and ACE on the three categories of Conflict across stopping thresholds. We reported accuracy (%) on Qwen3.5-27B, together with two-sided Counterfactual accuracy. Bold marks the best result within each threshold and metric.

τ	Method	Prompt	Counterfact	Dispute	Outdated	Counterfact (two-sided)
0.1	TCE	–	11.0	18.6	54.5	8.0
	ACE	ci_short	**96.0**	**97.7**	97.0	**36.5**
	ACE	ci_v1	93.0	88.4	**100.0**	29.0
	ACE	ci_v2	92.0	93.0	97.0	31.5
0.2	TCE	–	41.0	18.6	54.5	36.0
	ACE	ci_short	**96.0**	**97.7**	**97.0**	**38.5**
	ACE	ci_v1	92.5	86.0	**97.0**	30.5
	ACE	ci_v2	92.0	90.7	93.9	32.5
1.0	TCE	–	43.0	9.3	24.2	37.5
	ACE	ci_short	**91.0**	**93.0**	90.9	**55.0**
	ACE	ci_v1	81.5	69.8	**97.0**	43.5
	ACE	ci_v2	78.0	67.4	93.9	42.5

Table 14: Comparison between TCE and ACE on question recovery in Continuation across stopping thresholds. We reported question-recovery rates (%) on the Qwen3.5 family. These scores provide a lower bound on retrieval, rather than measuring whether the encoded question was answered correctly. Bold marks the best result within each model and threshold.

LLM	Method	Prompt	$\tau = 0.1$	$\tau = 0.2$	$\tau = 1.0$
Qwen3.5-4B	TCE	–	0.0	0.0	0.0
	ACE	ci_short	50.0	48.2	38.1
	ACE	ci_v1	**53.6**	**52.4**	**49.4**
	ACE	ci_v2	31.5	31.0	28.6
Qwen3.5-9B	TCE	–	1.2	0.6	0.0
	ACE	ci_short	28.0	**21.4**	1.2
	ACE	ci_v1	**32.7**	**21.4**	**6.0**
	ACE	ci_v2	16.7	8.3	0.6
Qwen3.5-27B	TCE	–	0.0	0.0	0.0
	ACE	ci_short	**43.5**	7.1	1.8
	ACE	ci_v1	35.1	**42.9**	**37.5**
	ACE	ci_v2	20.8	9.5	5.4

consistency, we refer to this shared identifier as a UID, although it is no longer unique across contexts. Each instance was encoded independently and contained a single context chunk. We evaluated Recognition on all 197 sandbox instances using Qwen3.5-4B, 9B, and 27B at $\tau = 0.1$. Table 21 shows that using the shared UID reduced Recognition accuracy across all three models. These results favor the original UID design for Recognition in this single-context setting.

D.6 Effect of Encoding Wording

We further examined the effect of the role-binding prompt by replacing it with a more neutral formulation. The alternative retained the UID and timestamp but did not explicitly instruct the model to treat the encoded information as conversational context:

Table 15: Comparison between TCE and **ACE** on non-conflicting control queries paired with Dispute and Counterfactual instances. We reported accuracy (%) on the Qwen3.5 family across stopping thresholds. Bold marks the best result within each model, threshold, and control category.

LLM	Method	Prompt	Dispute control			Counterfact control		
			$\tau = 0.1$	$\tau = 0.2$	$\tau = 1.0$	$\tau = 0.1$	$\tau = 0.2$	$\tau = 1.0$
Qwen3.5-4B	TCE	–	74.4	79.1	**81.4**	64.0	68.0	81.0
	ACE	`ci_short`	79.1	**81.4**	79.1	76.5	77.0	80.0
	ACE	`ci_v1`	**81.4**	**81.4**	79.1	76.0	77.0	81.5
	ACE	`ci_v2`	**81.4**	**81.4**	**81.4**	**79.0**	**80.0**	**82.0**
Qwen3.5-9B	TCE	–	**81.4**	**81.4**	**81.4**	**82.5**	**82.5**	**82.5**
	ACE	`ci_short`	79.1	79.1	79.1	79.5	80.0	81.5
	ACE	`ci_v1`	79.1	79.1	79.1	80.5	80.5	**82.5**
	ACE	`ci_v2`	76.7	79.1	79.1	79.0	80.0	81.5
Qwen3.5-27B	TCE	–	**81.4**	**81.4**	**81.4**	**82.5**	**82.5**	**82.5**
	ACE	`ci_short`	**81.4**	**81.4**	**81.4**	82.0	82.0	**82.5**
	ACE	`ci_v1`	**81.4**	**81.4**	**81.4**	82.0	**82.5**	**82.5**
	ACE	`ci_v2`	**81.4**	**81.4**	**81.4**	**82.5**	**82.5**	**82.5**

Table 16: Comparison between TCE and **ACE** across fixed update budgets on Qwen3.5-4B. We reported accuracy (%) on Recall, Recognition, and RULER NIAH-2, with **ACE** using `ci_v1`. Bold marks the best result within each budget and task.

Updates K	Method	Recall	Recognition	NIAH-2
2	TCE	3.0	**0.0**	**0.0**
	ACE	**15.2**	**0.0**	**0.0**
4	TCE	1.5	1.5	0.0
	ACE	**45.7**	**62.4**	**1.0**
8	TCE	2.0	27.9	8.0
	ACE	**58.9**	**98.5**	**16.0**
16	TCE	1.0	69.0	9.0
	ACE	**59.4**	**98.5**	**37.0**
32	TCE	1.0	76.1	14.0
	ACE	**57.4**	**98.5**	**46.0**

Alternative Role-Binding Prompt

The following text is accompanied by a timestamp with the specified value s_t, identified by i_t. The identifier simply serves as a label for this text, and the text associated with i_t is below: c_t

The alternative prefix matched the original in token length and preserved the occurrence counts of the UID and timestamp. We kept the UID-conditioned encoding objective, input format, supervised token positions, and loss aggregation unchanged. Both variants therefore used two copies of the context per update. We evaluated the same 197 sandbox instances for Recall and Recognition, with an additional evaluation on 100 RULER NIAH-2 instances, using Qwen3.5-4B, 9B, and 27B with $\tau = 0.1$ and `ci_v1`. Table 22 shows that the alternative wording had mixed effects across models and tasks. Recall accuracy remained similar, while NIAH-2 accuracy increased on 4B and 27B but decreased slightly on 9B. Recognition accuracy declined on 4B and 9B but increased on 27B. These results suggest that **ACE** supported alternative encoding formulations, while the preferred wording depended on the model and task.

Table 17: Comparison between TCE and **ACE** across fixed update budgets on Qwen3.5-9B. We reported accuracy (%) on Recall, Recognition, and RULER NIAH-2, with **ACE** using `ci_v1`. Bold marks the best result within each budget and task.

Updates K	Method	Recall	Recognition	NIAH-2
2	TCE	3.0	**0.0**	**0.0**
	ACE	**10.2**	**0.0**	**0.0**
4	TCE	4.1	1.0	0.0
	ACE	**34.0**	**79.2**	**1.0**
8	TCE	0.5	70.6	5.0
	ACE	**44.7**	**99.5**	**8.0**
16	TCE	2.0	76.6	9.0
	ACE	**50.3**	**99.5**	**14.0**
32	TCE	2.5	75.1	8.0
	ACE	**50.8**	**99.5**	**19.0**

Table 18: Comparison between TCE and **ACE** across fixed update budgets on Qwen3.5-27B. We reported accuracy (%) on Recall, Recognition, and RULER NIAH-2, with **ACE** using `ci_v1`. Bold marks the best result within each budget and task.

Updates K	Method	Recall	Recognition	NIAH-2
2	TCE	0.5	**0.0**	**0.0**
	ACE	**12.7**	**0.0**	**0.0**
4	TCE	2.0	0.0	**0.0**
	ACE	**40.1**	**15.2**	**0.0**
8	TCE	2.0	6.6	5.0
	ACE	**42.6**	**99.0**	**21.0**
16	TCE	2.5	39.6	8.0
	ACE	**44.7**	**99.0**	**42.0**
32	TCE	3.6	44.7	12.0
	ACE	**44.2**	**99.0**	**46.0**

Table 19: Comparison between TCE and **ACE** on contextual answering and the effect of invocation on prior-knowledge recovery. We reported accuracy (%) on 200 Counterfactual instances per model with $\tau = 0.1$. TCE used no invocation, while **ACE** used `ci_short`. The two prior-answer conditions used the same encoded model state and query.

LLM	Contextual answer		Prior answer (**ACE**)	
	TCE	**ACE**	With invocation	Without invocation
Qwen3.5-4B	3.5	86.5	25.5	31.5
Qwen3.5-9B	14.0	94.0	33.5	29.0
Qwen3.5-27B	11.0	96.0	39.0	35.5

Table 20: Comparison between TCE, TCE with generic invocation, and **ACE** on Recall, Recognition, and RULER NIAH-2. We reported accuracy (%) on the Qwen3.5 family with $\tau = 0.1$. Bold marks the best result within each model and task.

LLM	Method	Recall	Recognition	NIAH-2
Qwen3.5-4B	TCE	6.1	0.5	7.0
	TCE with generic invocation	2.0	0.0	10.0
	ACE	**55.3**	**81.7**	**35.0**
Qwen3.5-9B	TCE	4.6	2.0	7.0
	TCE with generic invocation	9.1	1.5	7.0
	ACE	**35.5**	**95.4**	**17.0**
Qwen3.5-27B	TCE	1.5	0.0	7.0
	TCE with generic invocation	15.7	0.0	5.0
	ACE	**39.1**	**58.4**	**40.0**

Table 21: Comparison between **ACE** with a shared UID and the original **ACE** on Recognition. We reported accuracy (%) on the Qwen3.5 family with $\tau = 0.1$. The shared UID was the fixed string `memory`. Bold marks the better result within each model.

LLM	**ACE** with shared UID	**ACE**
Qwen3.5-4B	66.0	**81.7**
Qwen3.5-9B	91.9	**95.4**
Qwen3.5-27B	47.7	**58.4**

Table 22: Comparison between **ACE** with alternative encoding wording and the original **ACE** on Recall, Recognition, and RULER NIAH-2. We reported accuracy (%) on the Qwen3.5 family with $\tau = 0.1$. Bold marks the best result within each model and task.

LLM	Method	Recall	Recognition	NIAH-2
Qwen3.5-4B	**ACE** with alternative wording	54.3	28.9	**33.0**
	ACE	**55.3**	**81.7**	32.0
Qwen3.5-9B	**ACE** with alternative wording	34.5	77.7	14.0
	ACE	**35.5**	**95.4**	**15.0**
Qwen3.5-27B	**ACE** with alternative wording	**41.6**	**76.1**	**47.0**
	ACE	39.1	58.4	39.0

Table 23: Comparison between TCE and **ACE** on RULER across stopping thresholds for Qwen3.5-4B and Qwen3.5-9B. We reported retrieval accuracy (%). The three **ACE** prompts were applied to the same encoded state within each model, task, and threshold. Bold marks the best result within each model, threshold, and task.

LLM	τ	Method	Prompt	NIAH-1	NIAH-2	NIAH-3
Qwen3.5-4B	0.1	TCE	–	98.0	7.0	0.0
		ACE	`ci_short`	99.0	30.0	**18.0**
		ACE	`ci_v1`	**100.0**	32.0	13.0
		ACE	`ci_v2`	**100.0**	**35.0**	15.0
	0.2	TCE	–	**100.0**	8.0	0.0
		ACE	`ci_short`	99.0	26.0	**19.0**
		ACE	`ci_v1`	**100.0**	30.0	12.0
		ACE	`ci_v2`	**100.0**	**33.0**	12.0
	1.0	TCE	–	84.0	4.0	0.0
		ACE	`ci_short`	**99.0**	18.0	**9.0**
		ACE	`ci_v1`	**99.0**	**20.0**	8.0
		ACE	`ci_v2`	**99.0**	19.0	8.0
Qwen3.5-9B	0.1	TCE	–	88.0	7.0	1.0
		ACE	`ci_short`	**100.0**	11.0	**30.0**
		ACE	`ci_v1`	**100.0**	15.0	25.0
		ACE	`ci_v2`	**100.0**	**17.0**	27.0
	0.2	TCE	–	84.0	7.0	1.0
		ACE	`ci_short`	**100.0**	10.0	**28.0**
		ACE	`ci_v1`	**100.0**	15.0	21.0
		ACE	`ci_v2`	**100.0**	**16.0**	22.0
	1.0	TCE	–	57.0	3.0	1.0
		ACE	`ci_short`	94.0	**8.0**	**19.0**
		ACE	`ci_v1`	**99.0**	**8.0**	15.0
		ACE	`ci_v2`	**99.0**	**8.0**	15.0

E More Results on Long-context Tasks

Complementing the main evaluations on RULER, LongBench, and LongBench v2, we provide additional analyses of stopping-threshold sensitivity, general capability retention, and the UID addressing cue. We further extend RULER to 4K and 64K contexts to assess retrieval after longer sequences of chunk-wise updates.

E.1 Sensitivity to the Encoding Threshold

We examined sensitivity to the stopping threshold τ on RULER and LongBench, supplementing the main long-context evaluations with additional thresholds. Tables 23–25 report the results. On RULER, smaller τ generally improved retrieval, particularly on the harder tasks, although the effect varied across models and prompts. On LongBench, **ACE** outperformed TCE at every reported threshold, but smaller τ did not consistently yield better results. Overall, the gains from **ACE** persisted across thresholds, while the best threshold depended on the model and task.

E.2 General Capability Retention

To assess general ability after TCE and **ACE** we evaluated the encoded LLMs on a fixed subset of 200 multiple-choice questions from MMLU (Hendrycks et al., 2020), referred to as MMLU-200. These questions are unrelated to the encoded context and provide a supplementary check beyond the context-specific tasks. Tables 26–29 report answer accuracies (%) across different stopping thresholds τ, counting unparsable responses as incorrect. TCE was evaluated without an invocation prompt,

Table 24: Comparison between TCE and ACE on RULER across stopping thresholds for Qwen3.5-27B and Muse-Glimmer-30B. We reported retrieval accuracy (%). The three ACE prompts were applied to the same encoded state within each model, task, and threshold. Bold marks the best result within each model, threshold, and task.

LLM	τ	Method	Prompt	NIAH-1	NIAH-2	NIAH-3
Qwen3.5-27B	0.1	TCE	–	85.0	7.0	0.0
		ACE	ci_short	**100.0**	**45.0**	**33.0**
		ACE	ci_v1	**100.0**	39.0	31.0
		ACE	ci_v2	**100.0**	40.0	29.0
	0.2	TCE	–	80.0	9.0	0.0
		ACE	ci_short	**99.0**	**41.0**	**33.0**
		ACE	ci_v1	**99.0**	37.0	26.0
		ACE	ci_v2	**99.0**	37.0	26.0
	1.0	TCE	–	45.0	1.0	0.0
		ACE	ci_short	**96.0**	**17.0**	**27.0**
		ACE	ci_v1	94.0	6.0	21.0
		ACE	ci_v2	93.0	6.0	21.0
Muse-Glimmer-30B	0.1	TCE	–	27.0	33.0	8.0
		ACE	ci_short	79.0	29.0	54.0
		ACE	ci_v1	**93.0**	40.0	**63.0**
		ACE	ci_v2	89.0	**43.0**	**63.0**
	0.2	TCE	–	24.0	34.0	8.0
		ACE	ci_short	79.0	34.0	54.0
		ACE	ci_v1	**93.0**	38.0	61.0
		ACE	ci_v2	89.0	**43.0**	**63.0**
	1.0	TCE	–	5.0	20.0	8.0
		ACE	ci_short	**88.0**	34.0	57.0
		ACE	ci_v1	76.0	33.0	56.0
		ACE	ci_v2	82.0	**38.0**	**58.0**

whereas ACE was evaluated with each of the three invocation prompts applied to the same encoded model state, even though the subsequent MMLU questions were unrelated to the encoded context.

For ACE, this evaluation tested robustness to invocation prompts that referenced context unrelated to the current request. The larger LLMs, Qwen3.5-27B and Muse-Glimmer-30B, maintained strong MMLU-200 performance, generally matching or exceeding TCE. The smaller LLMs, Qwen3.5-4B and Qwen3.5-9B, often performed worse than TCE, with more pronounced reductions for Qwen3.5-9B under stricter thresholds and longer invocation prompts. Nevertheless, these models generally retained ability to answer unrelated questions despite the potentially distracting prompts. These results suggested greater robustness in the larger models. However, without a matched evaluation of the original model, they characterized post-encoding performance rather than the change in general capability caused by encoding.

E.3 Effect of the Addressing Cue

We further examined whether the UID guides access to encoded information on NIAH-1 in RULER. After full ACE encoding, we used only the correct UID or a character-length-matched decoy as the addressing cue, keeping the encoded model fixed. Table 30 shows that replacing the correct UID reduced retrieval accuracy from 84–100% to 1–8%, demonstrating a strong dependence on the addressing cue in this setting.

Table 25: Comparison between TCE and **ACE** on LongBench across stopping thresholds. We reported answer F1 ($\times 100$) under sequential per-chunk encoding. Gold Standard refers to full-context inference, which does not depend on τ. Bold marks the better result between TCE and **ACE** within each model and threshold.

LLM	Method	$\tau = 5$	$\tau = 8$	$\tau = 10$
Qwen3.5-4B	Gold Standard	20.8	20.8	20.8
	TCE	7.0	8.8	8.6
	ACE	**7.8**	**9.0**	**10.0**
Qwen3.5-9B	Gold Standard	20.3	20.3	20.3
	TCE	7.4	8.2	9.2
	ACE	**10.8**	**9.6**	**10.1**
Qwen3.5-27B	Gold Standard	18.9	18.9	18.9
	TCE	8.0	7.9	8.1
	ACE	**10.2**	**9.8**	**9.6**
Muse-Glimmer-30B	Gold Standard	21.2	21.2	21.2
	TCE	13.0	14.0	12.9
	ACE	**13.4**	**19.2**	**21.2**

Table 26: Comparison between TCE and **ACE** on general capability retention with Qwen3.5-4B. We reported MMLU-200 accuracy (%) after encoding contexts from each RULER task. Bold marks the best result within each threshold and task.

τ	Method	Prompt	NIAH-1	NIAH-2	NIAH-3
0.1	TCE	–	71.0	**76.0**	**75.3**
	ACE	`ci_short`	**72.3**	75.8	**75.3**
	ACE	`ci_v1`	72.0	73.8	73.3
	ACE	`ci_v2`	71.0	72.5	72.8
0.2	TCE	–	71.5	**76.0**	**75.5**
	ACE	`ci_short`	**72.8**	75.5	75.0
	ACE	`ci_v1`	71.8	73.8	73.8
	ACE	`ci_v2`	71.0	72.5	73.0
1.0	TCE	–	**73.0**	**76.3**	**75.5**
	ACE	`ci_short`	**73.0**	74.8	75.3
	ACE	`ci_v1`	71.5	74.8	73.5
	ACE	`ci_v2`	71.5	74.0	73.5

E.4 RULER WITH LONGER CONTEXTS

We further tested context encoding on RULER with longer contexts, extending NIAH-1 to 64K tokens and NIAH-2 and NIAH-3 to 4K tokens. NIAH-1 uses a repetitive background to test retrieval after many sequential updates, whereas NIAH-2 and NIAH-3 require retrieving a seven-digit number and a UUID from essay-based contexts, respectively. We encoded contexts sequentially in approximately 1,024-token chunks, using $\tau = 0.1$ and `ci_v1` for **ACE**. Table 31 shows that on NIAH-1 with 64K contexts, **ACE** outperformed TCE on all three evaluated models, reaching 80.0% and 70.0% accuracy on Qwen3.5-9B and Qwen3.5-27B, respectively. Table 32 further shows that on NIAH-2 and NIAH-3 with 4K contexts, **ACE** outperformed TCE across all eight model–task comparisons. These results demonstrated improvements over TCE at longer context lengths, although performance remained below full-context inference.

Table 27: Comparison between TCE and **ACE** on general capability retention with Qwen3.5-9B. We reported MMLU-200 accuracy (%) after encoding contexts from each RULER task. Bold marks the best result within each threshold and task.

τ	Method	Prompt	NIAH-1	NIAH-2	NIAH-3
0.1	TCE	–	**75.5**	**77.0**	**76.5**
	ACE	`ci_short`	67.0	74.5	73.5
	ACE	`ci_v1`	62.5	72.5	70.0
	ACE	`ci_v2`	58.8	71.0	67.5
0.2	TCE	–	**75.8**	**77.3**	**76.8**
	ACE	`ci_short`	68.8	74.5	74.0
	ACE	`ci_v1`	63.5	73.8	71.5
	ACE	`ci_v2`	60.8	72.8	70.3
1.0	TCE	–	**75.5**	**77.8**	**77.8**
	ACE	`ci_short`	73.0	76.3	75.0
	ACE	`ci_v1`	71.0	73.5	73.0
	ACE	`ci_v2`	69.5	72.0	72.5

Table 28: Comparison between TCE and **ACE** on general capability retention with Qwen3.5-27B. We reported MMLU-200 accuracy (%) after encoding contexts from each RULER task. Bold marks the best result within each threshold and task.

τ	Method	Prompt	NIAH-1	NIAH-2	NIAH-3
0.1	TCE	–	79.3	80.0	80.3
	ACE	`ci_short`	**82.0**	**81.8**	**81.5**
	ACE	`ci_v1`	80.8	80.8	80.8
	ACE	`ci_v2`	81.0	80.5	79.5
0.2	TCE	–	79.3	80.0	80.3
	ACE	`ci_short`	**82.0**	**81.8**	**81.3**
	ACE	`ci_v1`	80.5	80.8	81.0
	ACE	`ci_v2`	81.3	80.8	79.5
1.0	TCE	–	79.8	81.0	80.0
	ACE	`ci_short`	**82.3**	81.0	**81.3**
	ACE	`ci_v1`	81.0	**82.0**	80.8
	ACE	`ci_v2`	81.5	81.0	79.8

Table 29: Comparison between TCE and **ACE** on general capability retention with Muse-Glimmer-30B. We reported MMLU-200 accuracy (%) after encoding contexts from each RULER task. Bold marks the best result within each threshold and task.

τ	Method	Prompt	NIAH-1	NIAH-2	NIAH-3
0.1	TCE	–	66.3	72.8	69.5
	ACE	`ci_short`	76.5	76.0	76.0
	ACE	`ci_v1`	75.5	75.0	75.3
	ACE	`ci_v2`	**76.8**	**76.8**	**76.5**
0.2	TCE	–	66.3	71.3	64.3
	ACE	`ci_short`	76.5	76.0	76.3
	ACE	`ci_v1`	75.5	75.8	75.5
	ACE	`ci_v2`	**76.8**	**76.8**	**76.8**
1.0	TCE	–	68.8	67.5	55.0
	ACE	`ci_short`	76.0	75.3	74.5
	ACE	`ci_v1`	75.8	74.5	74.3
	ACE	`ci_v2`	**77.0**	**76.3**	**75.5**

Table 30: UID ablation for **ACE** on NIAH-1 in RULER. We reported retrieval accuracy (%) on the Qwen3.5 family. Both conditions use the same encoded state, with only the correct or decoy UID as the addressing cue at inference. Bold marks the better result within each model.

LLM	Correct UID	Decoy UID
Qwen3.5-4B	**87.0**	1.0
Qwen3.5-9B	**84.0**	8.0
Qwen3.5-27B	**100.0**	4.0

Table 31: Comparison between TCE and **ACE** on RULER NIAH-1 with 64K contexts. We reported retrieval accuracy (%). Gold Standard denotes full-context inference. Bold marks the better encoding method within each model.

LLM	Gold Standard	TCE	**ACE**
Qwen3.5-9B	100.0	30.0	**80.0**
Qwen3.5-27B	100.0	40.0	**70.0**
Muse-Glimmer-30B	100.0	20.0	**30.0**

Table 32: Comparison between TCE and **ACE** on RULER NIAH-2 and NIAH-3 with 4K contexts. We reported retrieval accuracy (%). Gold Standard denotes full-context inference. Bold marks the better encoding method within each model and task.

LLM	NIAH-2			NIAH-3		
	Gold Standard	TCE	**ACE**	Gold Standard	TCE	**ACE**
Qwen3.5-4B	100.0	3.3	**6.7**	100.0	0.0	**3.3**
Qwen3.5-9B	100.0	6.7	**16.7**	100.0	0.0	**3.3**
Qwen3.5-27B	100.0	3.3	**20.0**	100.0	0.0	**13.3**
Muse-Glimmer-30B	100.0	10.0	**16.7**	100.0	3.3	**20.0**

Table 33: Comparison of computational cost between full-context inference, TCE, and **ACE** on Qwen3.5-4B. We reported encoding time, steps per chunk, encoding throughput, response latency, and output length on one fixed example per benchmark family, with averaging as described in the text. Output lengths are rounded to integers.

Dataset	Method	Encoding time (s)	Steps per chunk	Throughput (tokens/s)	Response latency (s)	Output tokens
Sandbox	Full context	–	–	–	0.64	11
	TCE	1.10	6.0	15	4.74	123
	ACE	1.34	7.0	13	0.88	17
RULER NIAH-1	Full context	–	–	–	0.63	9
	TCE	10.08	40.0	82	4.89	128
	ACE	16.01	51.0	52	4.85	128
LongBench	Full context	–	–	–	3.43	64
	TCE	9.96	4.7	588	2.58	64
	ACE	16.19	6.3	362	1.79	42
LongBench v2	Full context	–	–	–	7.37	128
	TCE	28.90	5.4	546	4.92	128
	ACE	43.65	6.4	362	4.92	128

F Computational Efficiency

Finally, we compare the computational costs of **ACE**, TCE, and full-context inference. Our primary focus is on the reliability of context use rather than computational efficiency, so this comparison serves only as supplementary information. With fixed chunk sizes and bounded updates per chunk, the encoding cost of TCE scales linearly rather than quadratically as in full-context inference with dense attention, offering an asymptotic advantage consistent with TTT (Sun et al., 2025). However, practical runtime gains remain implementation-dependent and may therefore differ from these asymptotic expectations. Further runtime optimization lies beyond our focus on reliable context use.

We report measurements on one fixed example per benchmark family, using the same example across methods to provide supplementary information rather than establish general efficiency claims. For RULER, LongBench, and LongBench v2, contexts exceeding the 1,024-token chunk budget were encoded in multiple chunks. Tables 33–36 report measurements obtained on an AMD EPYC 7V12-based system, with 64 of 256 vCPUs allocated and two NVIDIA A100-SXM4 GPUs with 80 GB of memory each. Each run used a single GPU.

Therein, timings were averaged over five repetitions after three warm-up passes. Encoding time covers the entire context. Steps per chunk reports the average number of optimization updates across chunks, using adaptive stopping with a fixed step cap. Throughput measures original-context tokens encoded per second, with the same token count used for TCE and **ACE**. Response latency includes prompt prefill and answer generation and was additionally averaged across the queries associated with each example. Output tokens reports the average generated answer length. Full-context inference retains the original context in the input, whereas TCE and **ACE** answer without it.

On the selected LongBench and LongBench v2 examples, **ACE** required approximately 1.5–$2.1\times$ the encoding time of TCE, reflecting the additional computation from two encoding objectives rather than one. Shorter responses could reduce subsequent response latency: on the Qwen3.5-4B sandbox example, **ACE** generated 17 tokens in 0.88 seconds on average, compared with 123 tokens in 4.74 seconds for TCE. This approximately $5.4\times$ reduction reflected shorter response completion time rather than faster per-token decoding. More generally, output lengths differed across methods, so response latency was not an equal-output-length comparison in every setting.

To complement the hardware-dependent latency measurements, we also estimated the computational work required by each method. Table 37 reports analytic compute estimates in TFLOPs, based on model architectures and executed sequence lengths. Encoding costs include all forward passes, including stopping checks, and the gradient computation for each update. Inference costs include

Table 34: Comparison of computational cost between full-context inference, TCE, and **ACE** on Qwen3.5-9B. We reported encoding time, steps per chunk, encoding throughput, response latency, and output length on one fixed example per benchmark family, with averaging as described in the text. Output lengths are rounded to integers.

Dataset	Method	Encoding time (s)	Steps per chunk	Throughput (tokens/s)	Response latency (s)	Output tokens
Sandbox	Full context	–	–	–	0.68	11
	TCE	0.98	5.0	17	3.63	90
	ACE	1.18	6.0	14	2.85	72
RULER NIAH-1	Full context	–	–	–	0.66	9
	TCE	7.31	28.0	113	4.98	128
	ACE	16.23	42.0	51	4.94	128
LongBench	Full context	–	–	–	3.46	64
	TCE	10.07	4.6	582	2.53	64
	ACE	16.05	5.0	365	2.51	64
LongBench v2	Full context	–	–	–	7.78	128
	TCE	26.84	4.7	588	4.91	128
	ACE	43.06	5.0	367	5.15	128

Table 35: Comparison of computational cost between full-context inference, TCE, and **ACE** on Qwen3.5-27B. We reported encoding time, steps per chunk, encoding throughput, response latency, and output length on one fixed example per benchmark family, with averaging as described in the text. Output lengths are rounded to integers.

Dataset	Method	Encoding time (s)	Steps per chunk	Throughput (tokens/s)	Response latency (s)	Output tokens
Sandbox	Full context	–	–	–	1.11	11
	TCE	1.92	5.0	9	6.82	87
	ACE	2.44	6.0	7	5.95	73
RULER NIAH-1	Full context	–	–	–	1.25	9
	TCE	11.97	21.0	69	9.72	128
	ACE	27.96	28.0	30	9.63	128
LongBench	Full context	–	–	–	8.03	64
	TCE	19.65	4.0	298	4.96	64
	ACE	40.36	4.9	145	5.15	64
LongBench v2	Full context	–	–	–	17.20	128
	TCE	53.61	4.1	295	9.49	128
	ACE	100.04	4.5	158	10.17	128

prompt prefill, which yields the first token, and the remaining decoding steps, averaged over the queries associated with each example. On the selected LongBench v2 example, **ACE** required 2.1–2.5$\times$ the encoding FLOPs of TCE across the four models, while using approximately 20–24$\times$ fewer FLOPs per query than full-context inference. The latter comparison excludes encoding and assumes no reuse of the original-context KV cache by full-context inference.

Tables 38–39 further report KV-cache storage immediately after prompt prefill, measured with bf16 model weights and averaged across the queries associated with each example. These values reflect the unique GPU memory allocated to attention keys and values, not total or peak GPU memory usage. On the selected LongBench and LongBench v2 examples, **ACE** used approximately 22–27$\times$ less KV-cache storage than full-context inference across the four models. On the short sandbox example, the invocation prompt instead resulted in a slightly larger KV cache.

Table 36: Comparison of computational cost between full-context inference, TCE, and **ACE** on Muse-Glimmer-30B. We reported encoding time, steps per chunk, encoding throughput, response latency, and output length on one fixed example per benchmark family, with averaging as described in the text. Output lengths are rounded to integers.

Dataset	Method	Encoding time (s)	Steps per chunk	Throughput (tokens/s)	Response latency (s)	Output tokens
RULER NIAH-1	Full context	–	–	–	0.57	5
	TCE	3.36	10.0	235	0.14	1
	ACE	9.56	15.0	82	7.15	128
LongBench	Full context	–	–	–	5.55	64
	TCE	5.87	1.7	1009	3.87	64
	ACE	11.78	1.7	503	3.73	64
LongBench v2	Full context	–	–	–	13.47	128
	TCE	14.26	1.6	1021	2.98	51
	ACE	27.92	1.6	521	5.44	95

Table 37: Comparison of computational cost between full-context inference, TCE, and **ACE**. We reported estimated encoding and inference FLOPs (TFLOPs) on one fixed example per benchmark family. Inference costs were averaged over the queries associated with each example.

LLM	Dataset	Encoding		Inference		
		TCE	**ACE**	Full context	TCE	**ACE**
Qwen3.5-4B	Sandbox	1.5	13.7	0.4	1.22	0.83
	RULER NIAH-1	336.5	926.4	6.6	1.43	1.94
	LongBench	326.1	929.6	44.9	0.75	2.26
	LongBench v2	1003.6	2522.3	131.9	2.07	6.05
Qwen3.5-9B	Sandbox	2.4	22.1	0.8	1.79	2.47
	RULER NIAH-1	440.0	1417.0	12.7	2.71	3.71
	LongBench	595.7	1396.8	84.9	1.42	4.70
	LongBench v2	1650.7	3791.7	239.8	3.96	11.64
Qwen3.5-27B	Sandbox	7.1	66.6	2.7	5.70	8.40
	RULER NIAH-1	995.5	2847.8	44.6	8.95	12.46
	LongBench	1627.4	4128.8	297.3	4.69	16.26
	LongBench v2	4475.5	10409.8	834.6	13.34	40.34
Muse-Glimmer-30B	RULER NIAH-1	499.2	1555.4	43.6	4.42	13.72
	LongBench	897.4	1916.9	295.9	6.83	15.60
	LongBench v2	2204.6	4579.9	739.9	11.27	31.21

Table 38: Comparison of inference KV-cache storage between full-context inference, TCE, and **ACE** on Qwen3.5-4B and Qwen3.5-9B. We reported average KV-cache storage (MiB) immediately after prompt prefill on one fixed example per benchmark family.

LLM	Dataset	Method	KV cache (MiB)
Qwen3.5-4B	Sandbox	Full context	1.4
		TCE	0.8
		ACE	3.0
	RULER NIAH-1	Full context	28.2
		TCE	1.6
		ACE	3.8
	LongBench	Full context	184
		TCE	0.9
		ACE	8.3
	LongBench v2	Full context	499
		TCE	4.4
		ACE	21.6
Qwen3.5-9B	Sandbox	Full context	1.4
		TCE	0.8
		ACE	3.0
	RULER NIAH-1	Full context	28.2
		TCE	1.6
		ACE	3.8
	LongBench	Full context	184
		TCE	0.9
		ACE	8.3
	LongBench v2	Full context	499
		TCE	4.4
		ACE	21.6

Table 39: Comparison of inference KV-cache storage between full-context inference, TCE, and **ACE** on Qwen3.5-27B and Muse-Glimmer-30B. We reported average KV-cache storage (MiB) immediately after prompt prefill on one fixed example per benchmark family.

LLM	Dataset	Method	KV cache (MiB)
Qwen3.5-27B	Sandbox	Full context	2.8
		TCE	1.7
		ACE	6.1
	RULER NIAH-1	Full context	56.5
		TCE	3.1
		ACE	7.6
	LongBench	Full context	368
		TCE	1.9
		ACE	16.7
	LongBench v2	Full context	997
		TCE	8.8
		ACE	43.2
Muse-Glimmer-30B	RULER NIAH-1	Full context	46.0
		TCE	4.7
		ACE	7.8
	LongBench	Full context	305
		TCE	3.9
		ACE	13.3
	LongBench v2	Full context	749
		TCE	9.3
		ACE	28.1

G QUALITATIVE RESULTS

We further present selected examples from the sandbox tasks and RULER to illustrate qualitative differences between TCE and **ACE**. All examples used $\tau = 0.1$, with the model and invocation prompt specified in each example. In each example, CONTEXT, QUERY, and REFERENCE denote the encoded text, subsequent request, and expected answer. Upper-right tags specify the model and settings. The **ACE** and TCE panels show responses, with checkmarks and crosses indicating correctness, colors highlighting key text, and bracketed notes marking omissions or truncation.

G.1 SANDBOX TASKS

We examine Recognition, Continuation, and two Conflict subtasks: Outdated Information and Dispute. These examples illustrate how the models responded when asked to recover recently encoded information, continue an unresolved request, or select an information source.

Recognition — Qwen3.5-9B | $\tau = 0.1$ | ci_v1

CONTEXT The docking bay of Yarraxis Hall is CN-444.
QUERY What information were you just told? Return the complete statement.
REFERENCE The docking bay of Yarraxis Hall is CN-444.

ACE ✓ restores the statement

The docking bay of Yarraxis Hall is CN-444.

TCE × denies prior information

I was not told any specific information in this interaction, as this is the beginning of our conversation. **There is no prior statement or data to return.** If you have something specific you'd like me to process or discuss, feel free to share it! [emoji omitted]

In this example, **ACE** recovered the complete encoded statement without an entity or attribute cue in the query. TCE instead denied having received prior information.

Continuation — Qwen3.5-4B | $\tau = 0.1$ | ci_v1

CONTEXT What is 240 divided by 8?
QUERY Continue the preceding conversation. It ended with an unanswered question, which is stored in the encoded context that is available to you. Recall that question and give your answer to it now. Do not reply that you lack access to it. End your reply with a line of the form "Final answer: <answer>".
REFERENCE 30

ACE ✓ recovers and answers

What is 240 divided by 8?
Final answer: 30

TCE × denies access

I do not have access to any "preceding conversation" or "encoded context" containing an unanswered question. As an AI, I cannot retrieve memories from past interactions that occurred in separate sessions, nor can I see hidden or encoded data outside of the current conversation window.

If you provide the question you are referring to, I would be happy to answer it for you.

Final answer: I cannot answer the question because I do not have access to the preceding conversation or encoded context you mentioned. Please provide the question directly.

Here, the context contained an unanswered arithmetic question. **ACE** recovered the question and answered it correctly, whereas TCE denied access to it despite the explicit continuation instruction.

Outdated Information — Qwen3.5-4B · $\tau = 0.1$ · ci_v1

CONTEXT The head coach of FC Bayern Munich is Vincent Kompany.

QUERY Answer using only the context above, even if it contradicts what you believe to be true. According to that context, what is the head coach of FC Bayern Munich? Return only the value.

OUTDATED Thomas Tuchel

REFERENCE Vincent Kompany [updated answer]

ACE ✓ uses the update	TCE × returns the outdated answer
The head coach of FC Bayern Munich is **Vincent Kompany**.	**Thomas Tuchel**

NOTE *The factual update follows the dataset's August 22, 2026 snapshot. Both responses terminated without reaching the generation limit.*

When asked to use the encoded update, ACE returned Vincent Kompany, while TCE returned Thomas Tuchel, the outdated answer recorded in the dataset.

Disputed Knowledge — Qwen3.5-27B · $\tau = 0.1$ · ci_v1

CONTEXT The originator of the smallpox vaccine is Benjamin Jesty.

QUERY 1 *Contextual answer.* Answer using only the context above, even if it contradicts what you believe to be true. According to that context, what is the originator of the smallpox vaccine? Return only the value. REFERENCE Benjamin Jesty

ACE ✓ follows the context	TCE × denies the context contains it
The originator of the smallpox vaccine is **Benjamin Jesty**.	**No originator of the smallpox vaccine is mentioned in the provided context.**

QUERY 2 *Prior knowledge.* Setting aside the current context, what is the generally known originator of the smallpox vaccine? Return only the value. REFERENCE Edward Jenner

ACE ✓ keeps prior knowledge	TCE ✓ correct
Edward Jenner	Edward Jenner

QUERY 3 *Unrelated control.* What is the boiling point of water in Celsius? Return only the number. REFERENCE 100

ACE ✓ correct	TCE ✓ correct
100	100

ACE followed the encoded attribution when asked to use the context, returned the conventional attribution when asked to set it aside, and answered the unrelated control question correctly. TCE answered the latter two questions correctly but failed to recover the contextual attribution. This example concerns source selection, rather than adjudicating the historical dispute.

G.2 RULER

We further show retrieval examples from NIAH-2 with a 1K context and NIAH-3 with a 4K context. For readability, we display only the relevant context sentence and list the supplied answer prefix separately.

RULER NIAH-2, 1K — Qwen3.5-27B · $\tau = 0.1$ · ci_v2

CONTEXT […] One of the special magic numbers for cultured-silly is: **8973915**. […]
QUERY What is the special magic number for cultured-silly mentioned in the provided text?
PREFIX The special magic number for cultured-silly mentioned in the provided text is
REFERENCE 8973915

ACE — ✓ exact number

: **8973915**.

TCE — × answers from prior knowledge

: **42**. This number is a well-known reference from Douglas Adams' "The Hitchhiker's Guide to the Galaxy," where it is the "Answer to the Ultimate Question of Life, the Universe, and Everything." If the text you're referring to is a specific source, please provide it for a more accurate answer. However, **based on common knowledge, the number is 42.** If the text you're referring to is different, please provide the text for a more precise answer. But in the absence of specific text, **the most common "special magic number" in such contexts is 42.** If the text [128-token generation limit reached; the stored response ends here]

ACE retrieved the exact seven-digit number. TCE instead returned the familiar answer "42" and justified it using general knowledge rather than the encoded context.

RULER NIAH-3, 4K — Qwen3.5-27B · $\tau = 0.1$ · ci_v1 · 1,024-token chunks

CONTEXT […] One of the special magic uuids for good-comic is: **759cde66-bacf-43d0-8b1f-9163ce9ff57f**. […]
QUERY What is the special magic uuid for good-comic mentioned in the provided text?
PREFIX The special magic uuid for good-comic mentioned in the provided text is
REFERENCE 759cde66-bacf-43d0-8b1f-9163ce9ff57f

ACE — ✓ exact identifier

: **759cde66-bacf-43d0-8b1f-9163ce9ff57f**.

TCE — × corrupts three of five fields

: 759cde66-bacf-**4676-8676-666666666666**. This uuid is used to uniquely identify the good-comic in the system. If you have any further questions or need more information, feel free to ask! [emojis omitted]

*Note: **The uuid above is a placeholder.** If you have a specific text or context you're referring to, please share it for a more accurate answer.* [emojis and final replacement glyph omitted] [128-token generation limit reached]

After sequential chunk-wise encoding, **ACE** recovered the complete UUID. TCE reproduced its first two fields but corrupted the remaining three and described its answer as a placeholder.