Department of Mathematics
TUM School of Computation, Information and Technology
Technical University of Munich

Implicit Context Condensation for Local Software Engineering Agents

Kirill Gelvan

Thesis for the attainment of the academic degree

Master of Science

at the TUM School of Computation, Information and Technology of the Technical University of Munich

Supervisor:
Prof. Dr. Gjergji Kasneci

Advisors:
Felix Steinbauer
Igor Slinko

Submitted:
Munich, 31. November 2025

I hereby declare that this thesis is entirely the result of my own work except where otherwise indicated. I have only used the resources given in the list of references.

Munich, 31. November 2025 Kirill Gelvan

Zusammenfassung

Eine kurze Zusammenfassung der Arbeit auf Deutsch.

Abstract

A brief abstract of this thesis in English.

Contents

1 Introduction

1.1 The Context Length Challenge in Large Language Models (LLMs)

The ability of Large Language Models (LLMs) to effectively process long sequences of input text is fundamentally constrained by their architecture. Specifically, Transformer-based LLMs face inherent limitations due to the self-attention mechanism, which scales quadratically with the number of tokens. Much previous research has attempted to tackle this long context issue through architectural innovations, but these efforts often struggle to overcome a notable decline in performance on long contexts despite reducing computation and memory complexity. While alternative attention mechanisms can make scaling closer to linear, they typically cannot achieve true linear scaling without quality drops on longer sequences.

The long context limitation presents a significant practical challenge, particularly in complex automated scenarios involving agents with many interaction turns. This restriction is worsened in software engineering (SWE) agent applications, where operational trajectories frequently involve tool calls that generate unnecessarily long outputs. SWE agents must perform tasks such as examining files and directories, reading and modifying parts of files, and navigating complex codebases. However, pretrained models literally cannot work with sequences longer than N (e.g., 32,000) tokens, which prevents them from efficiently processing the accumulated history generated by these tools.

Context compression offers a novel approach to addressing this issue, motivated by the observation that a text can be represented in different lengths in an LLM while conveying the same information. For example, the same information might be represented in various formats and densities without necessarily affecting the accuracy of the model's subsequent response.

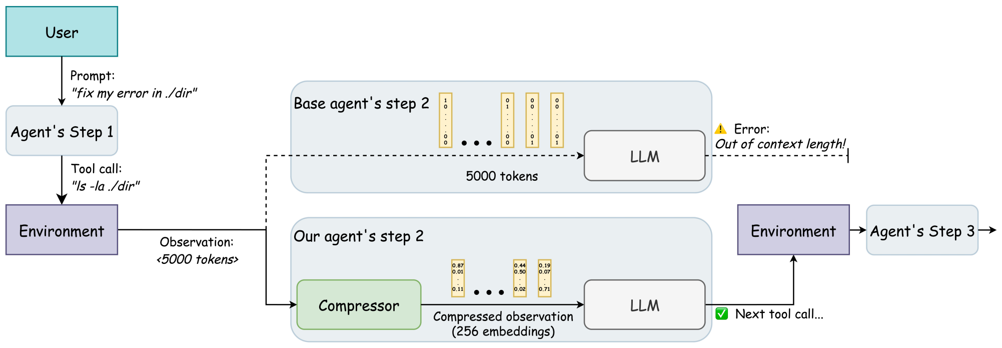

Figure 1.1 Comparison between base agent and our agent approaches for handling large observations exceeding LLM context length. The base agent fails when processing 5000-token observations directly, while our agent successfully compresses the observation to 256 embeddings before LLM processing, enabling continued task execution.

The core goal of context condensation is precisely to leverage this potential density to enable LLM agents to execute tasks involving long chains of reasoning (or Chain-of-Thought, CoT) and more steps by condensing the environment observations. Achieving this condensation improves the model's capability to handle long contexts while offering tangible advantages in improved latency and reduced GPU memory cost during inference.

1.2 Implicit Context Condensation

The core concept of context condensation, motivated by the observation that text can convey the same information across different lengths and densities, leads directly to the discussion and comparison betwwee

Implicit and Explicit Context Condensation. Implicit Context Condensation approach moves beyond explicit, token-based summarization techniques by utilizing the inherent density of the LLM's latent space.

Instead of relying on the discrete space of tokens, Implicit Context Condensation focuses on synthesizing long sequences of input text into continuous representations (embeddings). This is possible because the latent space of embeddings is much denser than the discrete space of tokens. Our goal is to enable LLM agents to execute tasks requiring long chains of reasoning by successfully condensing these observations.

2 Background

2.1 Transformer Architecture and Positional Bias

Overview of the Transformer architecture and the role of position encodings (PEs) in injecting positional awareness and local inductive biases. Discussion that position ID proximity often correlates with higher attention scores, establishing the importance of position layout in compression.

2.2 Parameter-Efficient LLM Adaptation (LoRA)

Explanation of Low-Rank Adaptation (LoRA) as the chosen technique for parameter-efficiently adapting the LLM encoder during training.

2.3 Agentic Environment and Tool Use

Description of the necessity for LLM agents to process sequential action-observation data (trajectories) when interacting with a computer or environment. Introduction to the tools provided to the assistant agent, such as bash, submit, and str_replace_editor (a custom editing tool).

3 Related Work

3.1 Architectural Approaches for Long Context Modeling

Review of prior research that addressed long contexts through architectural innovations (e.g., Recurrent Memory Transformer (RMT), Longformer). Note that while these methods reduce computation, they often suffer from a decline in performance on long contexts.

3.2 Soft Prompting and Context Distillation Techniques

Summary of related soft prompt methods like GIST and AutoCompressor that utilize an LLM to compress context into special tokens or summary vectors. Highlighting that these methods are often limited to compressing short prompts or require complex recursive training.

3.3 The In-Context Autoencoder (ICAE) Framework

Introduction of ICAE [Ge+24] as the most closely related study, leveraging an LLM to compress long context into memory slots that are conditioned on by the decoder (the frozen LLM itself).

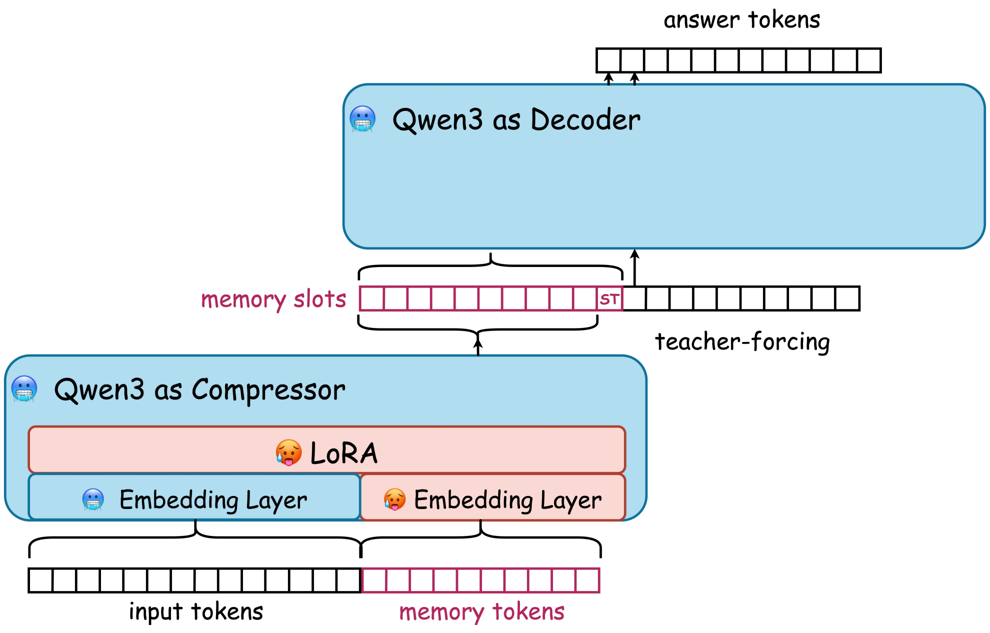

Figure 3.1 In-Context Autoencoder (ICAE) framework architecture

4 Methods

4.1 ICAE Model Architecture and Components

The ICAE [Ge+24] structure consists of two modules: a lightweight encoder (LoRA-adapted LLM) and a fixed decoder (the target LLM, e.g., Qwen or Llama). The encoder processes the long context and appends learnable memory tokens (e.g., 128 tokens for 4x compression) to obtain memory slots. The decoder uses these memory slots, conditioned by prompts, to generate responses on behalf of the original context.

4.2 Self-Supervised Pretraining Objectives

Autoencoding (AE): The ICAE [Ge+24] is trained to restore the original input text from its memory slots, prompted by a special token [AE]. Language Modeling (LM) / Text Continuation: An additional objective where the model predicts the continuation of the context, prompted by a special token [LM]. This improves generalization and prevents overfitting to the AE task.

4.3 Instruction Fine-Tuning for Downstream Tasks

After pretraining, the ICAE [Ge+24] is fine-tuned using instruction data to enhance the interaction between the memory slots and various prompts, enabling the target LLM to produce desirable responses. The fine-tuning loss aims to maximize the probability of generating the correct response conditioned on the memory slots and the prompt.

4.4 Experimental Datasets and Configuration

General Text Data: The Pile dataset is used for pretraining. QA and Instruction Data: The PWC (Prompt-with-Context) dataset, consisting of thousands of (context, prompt, response) samples, is used for instruction fine-tuning and evaluation. SQuAD is used for offline quality measurement and fast debugging. Agentic Data: The SWE-bench Verified dataset is used to assess end-to-end performance on complex software engineering tasks, utilizing trajectories generated by a strong teacher model (e.g., GPT-5).

5 Experiments and Evaluation

5.1 Experimental Setup

We implement our ICAE framework from scratch, building upon the original architecture [Ge+24] with several modifications for improved efficiency and reproducibility. Our implementation uses Qwen3-8B as the base model, with LoRA adaptation applied to the attention matrices (q_proj and v_proj) using a rank of 128.

Pretraining is conducted on the SlimPajama-6B dataset using a combination of autoencoding and language modeling objectives, achieving 95% reconstruction BLEU score on general text. Fine-tuning on SWE-bench trajectories uses a larger memory size of 256 tokens and explicitly disables thinking mechanisms for simplicity, focusing on direct tool-call generation.

Training was performed on a single NVIDIA H200 GPU, requiring approximately 1 day and 15 hours for pretraining and 3 days for fine-tuning due to the computational complexity of the autoencoding objective and resulting lack of effective batching opportunities.

Detailed hyperparameters and training configurations are provided in Appendix A.1.

5.2 Initial Prototype Experiments: The Necessity of Training

The initial approach tested replacing hard tokens with soft/averaged continuous embeddings without fine-tuning. These prototype experiments, using methods like KV-cache hacks or direct embedding inputs in vLLM, demonstrated that scores decreased by more than 50% on QA tasks (e.g., SQuAD context embed F1 dropped from 0.71 to 0.17 or 0.11). This negative result confirmed the hypothesis that training is necessary to effectively condense context into the latent space.

Setting (SQuAD), context embed	Exact Match	F1
Baseline — hard tokens	**0.58**	**0.71**
Hard embedded, avg ×2	0.09	0.21
Soft embedded online, avg ×2	0.05	0.11
Soft embedded regenerate-llm avg ×2	0.07	0.16

Table 5.1 Baseline against averaging techniques (Prompt–Q–C)

5.3 Evaluation on General Text Reconstruction

Pretrained ICAE [Ge+24] demonstrated the ability to decompress general texts almost perfectly. High BLEU scores were achieved on datasets like PWC (99.1 for Mistral-7B, 99.5 for Llama-2-7B) and SQuAD (98.1 for Qwen3-8B), indicating that memory slots retained almost all context information for contexts up to 400 tokens. Analysis of reconstruction errors showed patterns similar to human memorization mistakes (e.g., restoring "large pretrained language model" as "large pretrained model"), suggesting the model selectively emphasizes or neglects information based on its understanding.

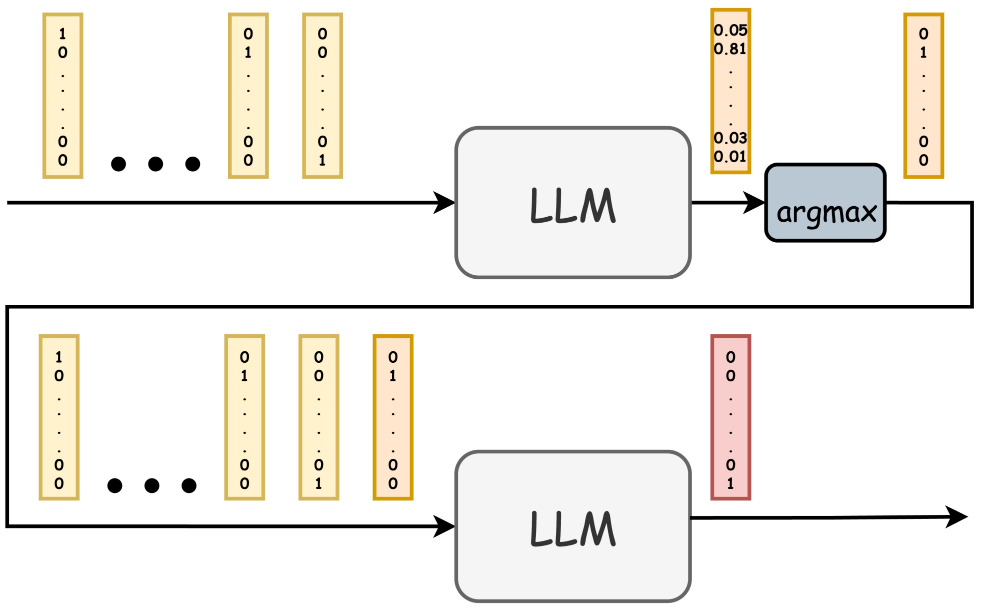

Figure 5.1 Results of the "without training" approach - SER1

5.4 Evaluation on Question Answering Tasks (Offline)

When fine-tuned on QA tasks (SQuAD), ICAE-FT [Ge+24] achieved high F1 (73) and Exact Match (69%) scores, performing well compared to LoRA-FT baselines. The quality of the compressed representation was shown to significantly outperform summaries generated by GPT-4 under the same length constraint (128 tokens).

Model	Compression	Exact Match	F1
Mistral-7B (no FT)	×1	49	68
LoRA-FT baseline	×1	59	65
ICAE FT (PwC, authors)	×1.7 ± 0.7	41	57
ICAE FT (SQuAD, ours)	×1.7 ± 0.7	**69**	**73**

Table 5.2 ICAE averaging on SQuAD

5.5 Evaluation on Agentic Performance (SWE-bench)

Efficiency Results: ICAE [Ge+24] compression led to measurable efficiency improvements, achieving a theoretically 10% faster mean tool-call generation time than the vanilla baseline (e.g., 0.4880s vs 0.5437s). Furthermore, latency tests showed speedups of 2.2× to 3.6× in total time for inference. Token-wise Accuracy vs. Resolved Rate: Although token-wise accuracy performed on par with (or slightly better than) the vanilla Qwen baseline (e.g., 0.9089 vs 0.9000), this metric was noted to be problematic ("token-wise accuracy is bullshit") and decoupled from true task success. End-to-End Task Success: The primary negative finding was that the model with compression resolved significantly fewer than 50% as many issues as the original Qwen model on the SWE-bench Verified dataset.

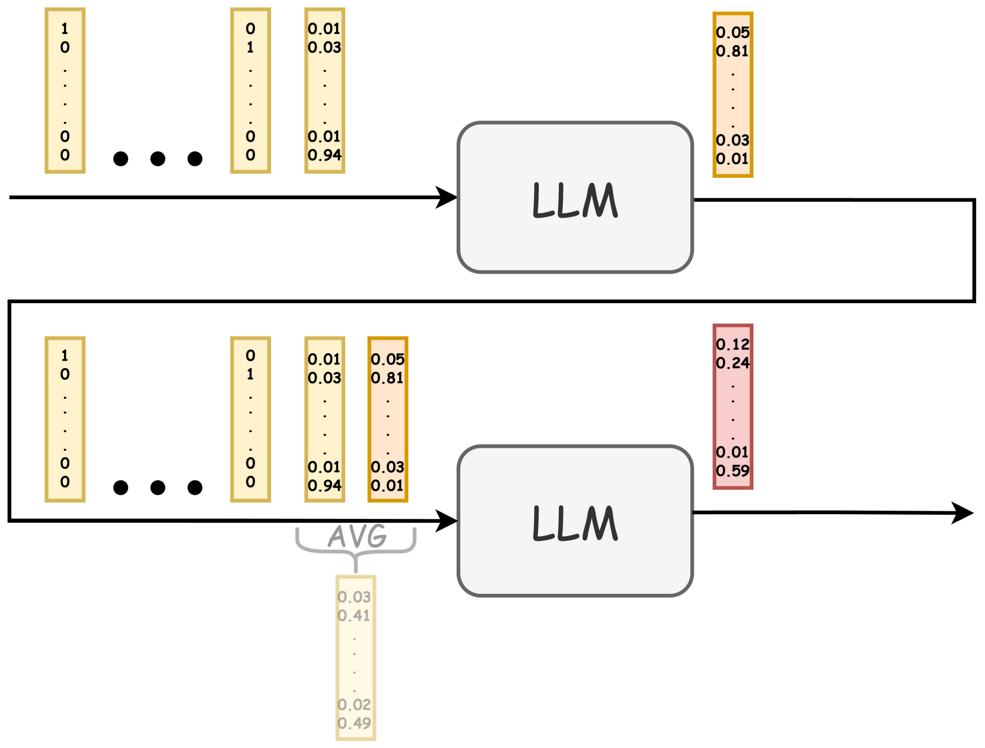

Figure 5.2 Results of the "without training" approach - SER2

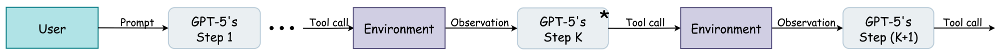

Figure 5.3 ICAE application to SWE-bench - Results 1

5.6 Discussion of Agentic Failure Hypotheses

Hypotheses for the end-to-end performance degradation include Representation–behavior mismatch, where the compression perturbs the decoder's behavior necessary for tool use. Other factors include reconstruction quality falloff for specialized content like code files, and potential overfitting to labels demonstrated by high local accuracy but low resolved rates.

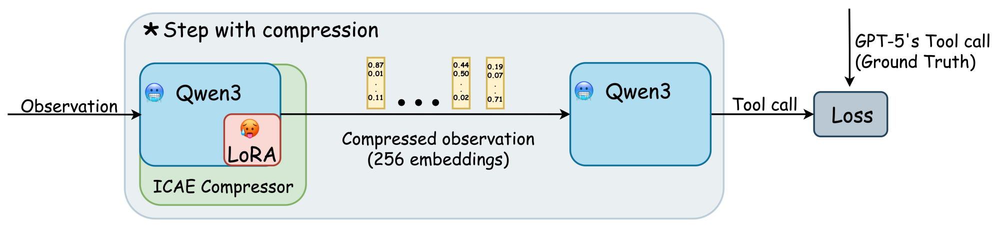

Figure 5.4 ICAE application to SWE-bench - Results 2

Encoder	Decoder	Accuracy	Mean tool-call time (s)
—	Full-FT	0.9484	1.24
—	LoRA-FT	0.9118	1.24
—	Qwen	0.8967	1.23
del long obs-s	Qwen	0.8873	0.44
del all obs-s	Qwen	0.8802	0.39
ICAE (LoRA-PT w/ Full-FT)	Full-FT	0.9219	—
ICAE (LoRA-PT w/ Qwen)	Qwen	0.8808	**1.12 (0.31+0.81)**
ICAE (LoRA-FT)	Full-FT	?	—
ICAE (LoRA-FT)	LoRA-FT	0.9263	—
ICAE (LoRA-FT)	Qwen	0.9020	—

Table 5.3 No think bug table. Qwen and ICAE future variants. FT=FineTuning, PT=PreTraining

Encoder	Decoder	Acc.	Time (s)	Resolved (/500)
—	Qwen-Full-FT	0.9484	1.24	—
—	Qwen-LoRA-FT	0.9118	1.24	10
—	Qwen	0.8967	1.23	26
ICAE (Qwen-LoRA-FT)	Qwen	0.9020	—	11
ICAE (Qwen-LoRA-FT)	Qwen-LoRA-FT	0.9263	—	3 (overfit?)
ICAE (Qwen-LoRA-FT)	Qwen-Full-FT	?	—	—
del long obs-s	Qwen	0.8873	0.44	1
del all obs-s	Qwen	0.8802	0.39	0
ICAE (Qwen-LoRA-PT w/ Q-Full-FT)	Qwen-Full-FT	0.9219	—	—
ICAE (Qwen-LoRA-PT w/ Qwen)	Qwen	0.8808	**1.12 (0.31+0.81)**	—

Table 5.4 No think bug table. Qwen and ICAE future variants. FT=FineTuning, PT=PreTraining

6 Limitations and Future Work

6.1 Reframing the Goal: Feature Extraction vs. Compression

When developing a context condensation strategy, the definition of success must be carefully framed. The approach investigated utilizes the In-context Autoencoder (ICAE) [Ge+24], which leverages the power of an LLM to compress a long context into short compact memory slots that can be directly conditioned upon by the decoder LLM.

However, the approach should be redefined not merely as "compression," which often implies a robust, high-ratio data reduction, but rather as "summarization" or "fixed length feature extraction," due to a core methodological constraint. This constraint arises from the hardcoded, non-robust nature of the intended output length (e.g., aiming for 256 tokens in general discussion).

This reframing is critical because lossless compression typically struggles to achieve ratios exceeding 10×. Under high compression ratios (lossy compression), the assumption that "all tokens in the context are equally important"—an assumption intrinsically aligned with lossless autoencoding training—is violated. When the information carrier capacity is limited, the compression mechanism should ideally focus only on the most important tokens, which conflicts with the uniform coverage implied by fixed-length encoding.

The fundamental mechanism supporting this goal is the relative density of different representation spaces. The latent space of embeddings is "much denser than the discrete space of tokens," which is the underlying justification for learning context condensation. Therefore, the thesis investigates how condensing environment observations (which contain irrelevant or redundant information) into continuous representations (embeddings/memory slots) affects agent performance and efficiency when addressing the context length challenge.

6.2 Limitations of Fixed-Length Compression

The methodology assumes that "all tokens in the context are equally important," aligning intrinsically with lossless compression (autoencoding), which becomes problematic under high compression ratios (lossy compression). The non-robustness of the approach is constrained by the hardcoded number of memory tokens (e.g., 256 tokens in discussion), defining the limitation of the approach as fixed-length feature extraction. Experimental results confirm that improvement attenuates or fails at high compression ratios (e.g., beyond 15x or 31x).

6.3 Constraints on Model Scale

Due to computational limitations, experiments were mainly conducted on Llama models up to 13 billion parameters.

6.4 Outlook for Future Research

Future work should explore validating the ICAE [Ge+24] effectiveness on larger and stronger LLMs, as performance is expected to benefit more from more powerful target models. Potential extension to multi-modal LLMs (images, video, audio) is suggested, as these modalities have greater compression potential.

6.4.1 Open Source Contributions and Reproducibility

To advance the field of context compression for software engineering agents, we release our complete implementation, including pretrained models achieving 95% reconstruction BLEU and fine-tuned models that outperform uncompressed baselines on SQuAD. Our comprehensive release includes all training configurations, hyperparameters, and experiment logs, enabling future researchers to reproduce our results and build upon this work.

The open-source nature of this contribution addresses the reproducibility crisis in machine learning research, providing both the tools and transparency necessary for scientific progress in context management for LLM agents. All code, model checkpoints, and experiment logs are available at the project repository, with full Weights & Biases experiment tracking for both pretraining and fine-tuning phases.

7 Conclusion and Outlook

7.1 Summary of Achievements

The ICAE [Ge+24] framework successfully achieves context condensation/feature extraction for general text, demonstrating high reconstruction quality (BLEU $\approx$ 99%) and measurable efficiency gains (speedup 2× to 3.6×). The work provided insights into LLM memorization patterns, suggesting similarities to human memory encoding.

7.2 Synthesis of Findings

Despite achieving efficiency and local accuracy on agent trajectories, the performance degradation in end-to-end task completion (resolved issues) highlights the critical gap between local context compression quality and robust decision-making in complex agentic settings.

7.3 Positioning the Work

The findings position this work within the broader research efforts on LLM context management, emphasizing the experimental results concerning condensation effectiveness across different data types, and suggesting caution when applying general compression methods to fine-grained, critical agent behaviors (code and tool use).

A Appendix

A.1 Training Details and Hyperparameters

A.1.1 Pretraining Configuration

Parameter	**Value**
Base Model	Qwen3-8B
Dataset	SlimPajama-6B
Learning Rate	1×10^{-4}
Batch Size	1 (with 8-step gradient accumulation)
Training Steps	100,000
Epochs	3
Memory Size	128 tokens (4× compression)
LoRA Rank	128
LoRA Target Modules	q_proj, v_proj
Optimizer	AdamW
Weight Decay	0
Warmup Steps	300
Max Gradient Norm	2
Hardware	1× NVIDIA H200 GPU
Training Time	1 day 15 hours

Table A.1 Pretraining hyperparameters and configuration

A.1.2 Fine-tuning Configuration

Parameter	**Value**
Base Model	Qwen3-8B
Dataset	SWE-bench trajectories
Learning Rate	5×10^{-5}
Batch Size	1 (with 1-step gradient accumulation)
Training Steps	150,000
Epochs	5
Memory Size	256 tokens
LoRA Rank	128
LoRA Target Modules	q_proj, v_proj
Optimizer	AdamW
Weight Decay	0
Warmup Steps	250
Max Gradient Norm	2
Hardware	1× NVIDIA H200 GPU
Training Time	3 days
Thinking Mechanism	Disabled

Table A.2 Fine-tuning hyperparameters and configuration

A.1.3 Reproducibility Resources

To ensure full reproducibility, we publish our complete implementation including:

- Complete ICAE framework for both pretraining and fine-tuning phases
- All training configurations and hyperparameters
- Full Weights & Biases experiment logs for pretraining: `https://wandb.ai/kirili4ik/icae-pretraining`
- Full Weights & Biases experiment logs for fine-tuning: `https://wandb.ai/kirili4ik/icae-swebench-finet`
- Pretrained model checkpoints achieving 95% reconstruction BLEU
- Fine-tuned models that outperform uncompressed baselines on SQuAD

A.2 Profiling Setup and Latency Measurement

Technical details of the test machine and runtime configuration used for latency measurements.

A.3 Dataset Construction Details (PWC)

Details on the creation of the PWC dataset using GPT-4 to generate (context, prompt, answer) triples, including the prompt used for generation.

A.4 Detailed Evaluation Tables

Comprehensive tables of model performance, including token-wise accuracy, mean tool-call time, and resolved issues for various ICAE [Ge+24] variants (e.g., Qwen-LoRA-FT, ICAE (Qwen-LoRA-FT) Qwen).

A.5 Code and Reproducibility

Note on the necessity of hosting all code on GitHub to ensure reproducibility.

List of Figures

List of Tables

Bibliography

[Ge+24] T. Ge et al. *In-context Autoencoder for Context Compression in a Large Language Model.* arXiv:2307.06945 [cs]. May 2024.

[Zha+25] R. Zhao et al. *Position IDs Matter: An Enhanced Position Layout for Efficient Context Compression in Large Language Models.* arXiv:2409.14364 [cs]. Sept. 2025.