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

To use the LaTeX templates provided here you will need to add the directory `tum-templates` as a local package directory to your LaTeX distribution. An easy way to do this is by setting the environment variable `TEXINPUTS` to `.//:` on Linux/Mac systems (meaning: search the current directory and its subdirectories for packages first, then use the usual search path). On a Linux or Mac you can compile this document to a PDF file in a terminal through the following commands (the first command needs to be issued only once):

```
export TEXINPUTS=.//:
pdflatex -output-directory=. master
bibtex master
pdflatex -output-directory=. master
```

# 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. 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.

The long context limitation presents a significant practical challenge, particularly in complex automated scenarios. This restriction is exacerbated in software engineering (SWE) agent applications, where operational trajectories frequently involve tool calls that generate unnecessarily long outputs. For instance, custom editing tools available to the agent, such as str_replace_editor, are explicitly designed to truncate long command outputs, which are then marked to indicate the missing context. The LLMs' limited context length 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 by a context length of 2,572 characters, 512 (sub-)words, or a compact 128 memory slots, without necessarily affecting the accuracy of the model's subsequent response.

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. For instance, empirical testing shows that compression using the In-context Autoencoder (ICAE) framework can achieve over 2× to 3.6× inference speedup in total time, especially in compute-intensive scenarios.

## 1.2 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), 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 con-

densing 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.

Wichtige Informationen finden sich in table 1.1.

| Name | Place of Birth |
|---|---|
| Gauß | Braunschweig |
| Euler | Basel |
| Edmonds | Washington, D.C. |

**Table 1.1** A most wonderful table

# 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.

**Definition 2.1.1 (Definitheit)** *Hier definieren wir definitive Definitheit.*

**Satz 2.1.2 (vom X)** *War wohl nix. Es gilt aber*

$$\sum_{i=1}^{n} f_i(x) = \int \hat{f}(x)\,\mathrm{d}x$$

## 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.

Insbesondere weisen wir auf den wunderbaren Artikel von Edmonds [Edm65] und auf [GJ79] für weitere Hintergründe.

## 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 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).

# 4 Methods

## 4.1 ICAE Model Architecture and Components

The ICAE 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 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 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 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.

## 5.2 Evaluation on General Text Reconstruction

Pretrained ICAE 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.

## 5.3 Evaluation on Question Answering Tasks (Offline)

When fine-tuned on QA tasks (SQuAD), ICAE-FT 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).

## 5.4 Evaluation on Agentic Performance (SWE-bench)

Efficiency Results: ICAE 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.

## 5.5 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.

# 6 Limitations and Future Work

## 6.1 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).

## 6.2 Constraints on Model Scale

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

## 6.3 Outlook for Future Research

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

# 7 Conclusion and Outlook

## 7.1 Summary of Achievements

The ICAE 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

Detailed configuration for training (Optimizer, learning rate, batch size, number of steps for pretrain and fine-tuning). Information on hardware used (Nvidia H200 GPUs).

## 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 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


[Als08] B. Alspach. “The wonderful Walecki construction”. In: *Bull. Inst. Combin. Appl* 52 (2008), pp. 7–20.

[Edm65] J. Edmonds. “Paths, trees, and flowers”. In: *Canadian Journal of Mathematics* 17 (1965), pp. 449–467.

[GS62] D. Gale and L. S. Shapley. “College admissions and the stability of marriage”. In: *The American Mathematical Monthly* 69.1 (1962), pp. 9–15.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability*. WH Freeman & Co, 1979.