Implicit Context Condensation for Local SWE Agents

AI Agents & Planning

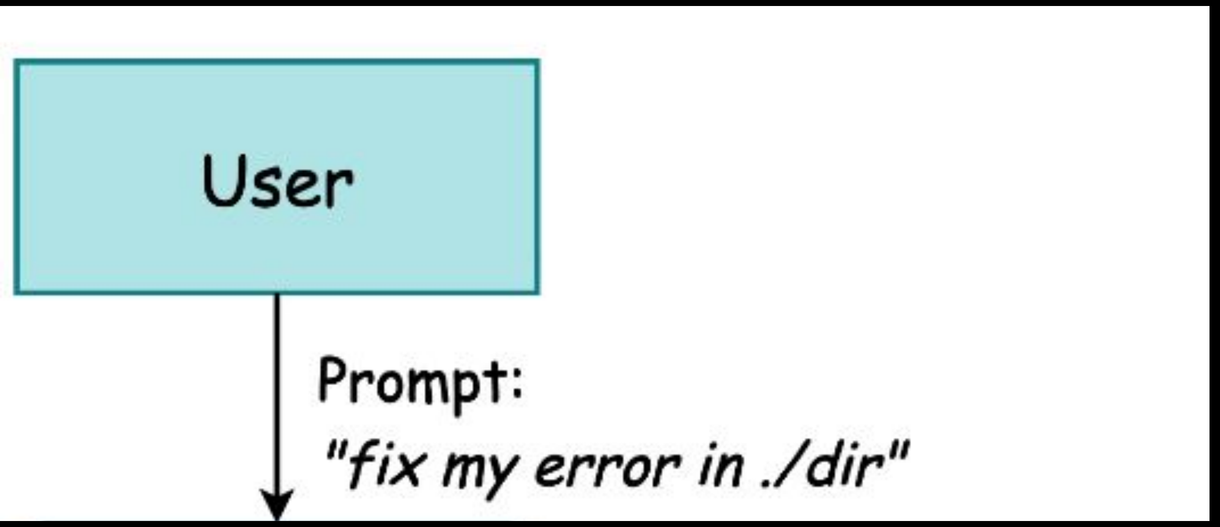
Kirill Gelvan w\ Igor Slinko, Felix Steinbauer, Yaroslav Zharov, Gjergji Kasneci

10.12.2025

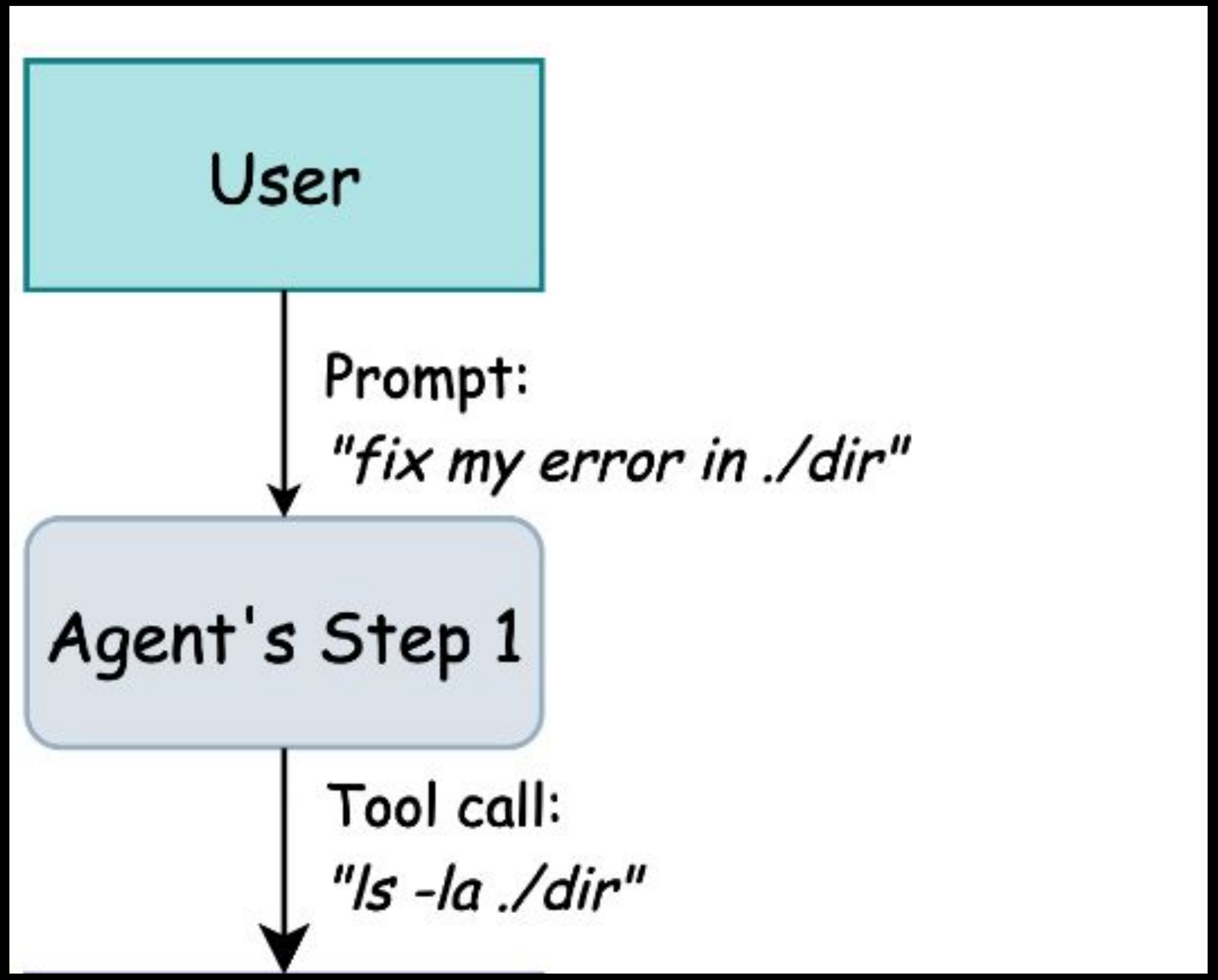
1. Motivation
2. Alternative Approaches
3. Methodology
4. Experiments
5. Discussion & Future Work
6. Conclusion

Motivation

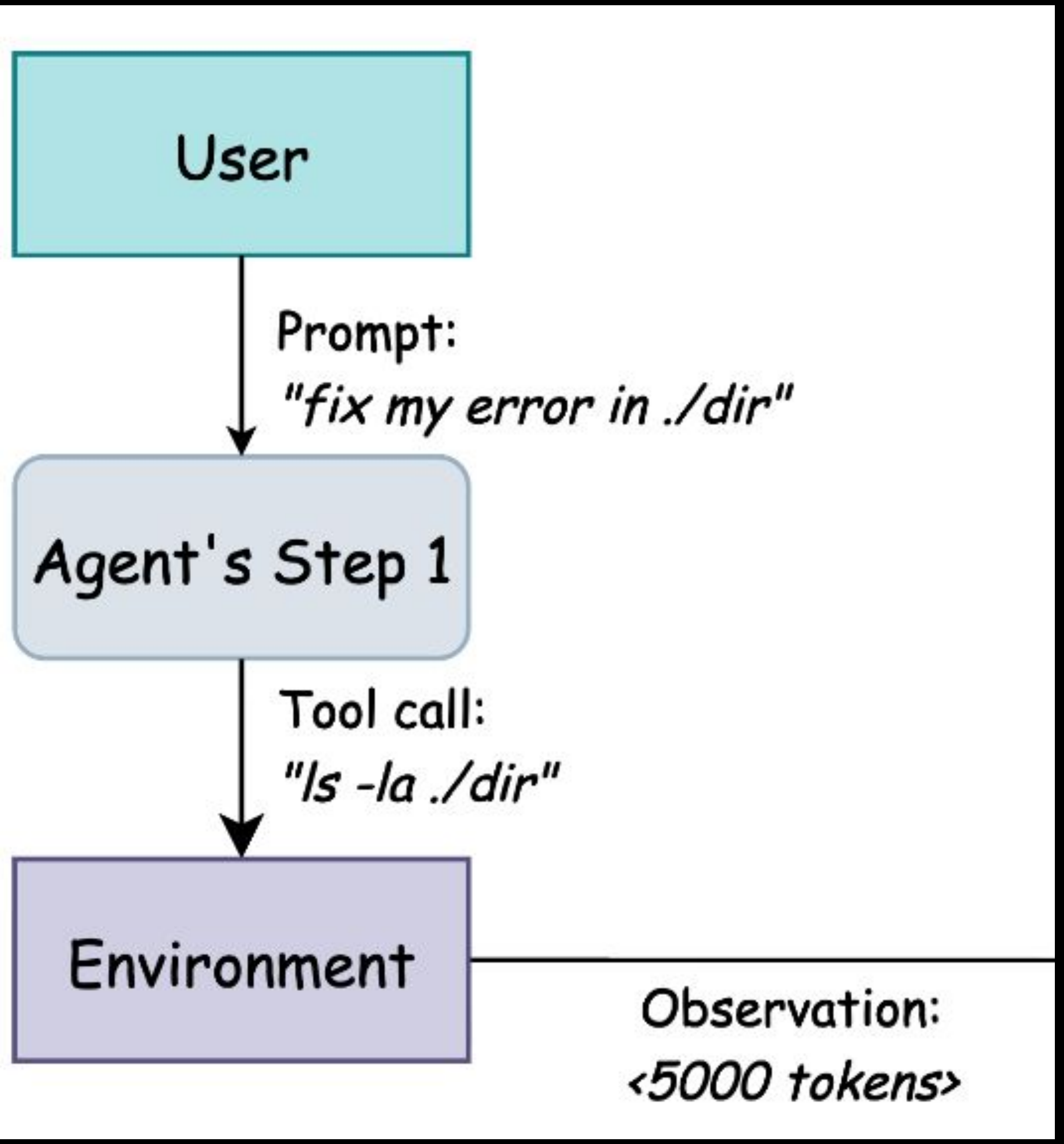
Example



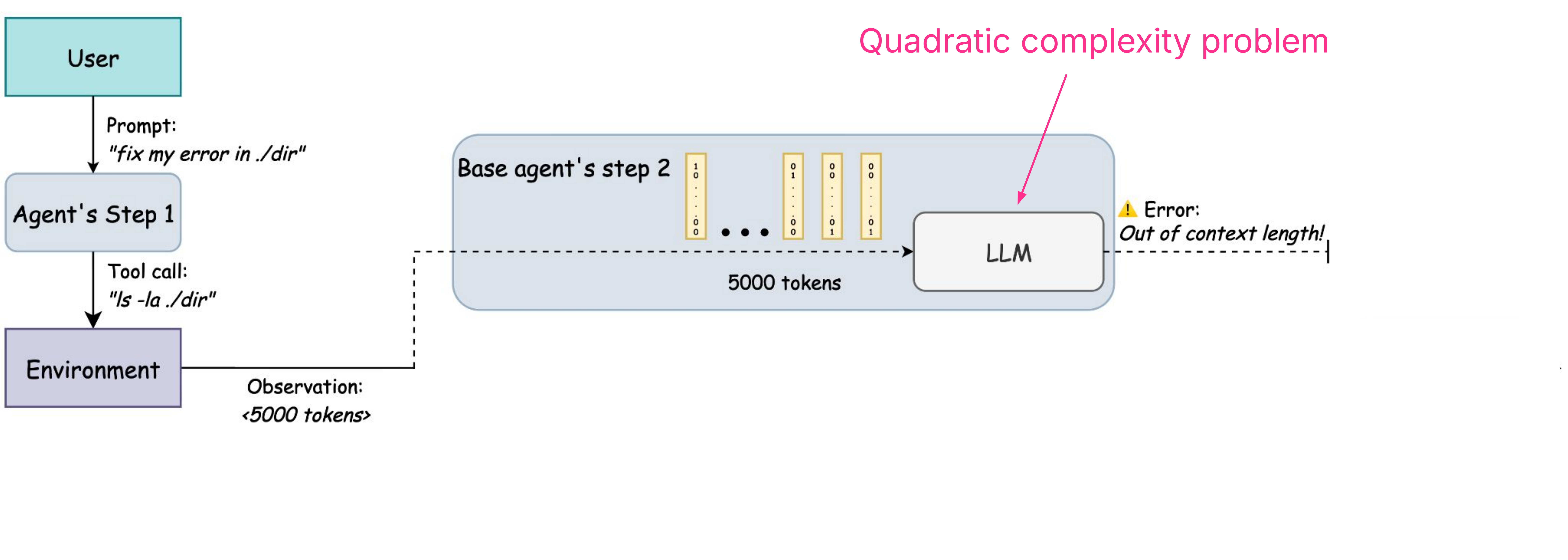
Example


User
Prompt:
"fix my error in ./dir"
Agent's Step 1
Tool call:
"ls -la ./dir"

Example

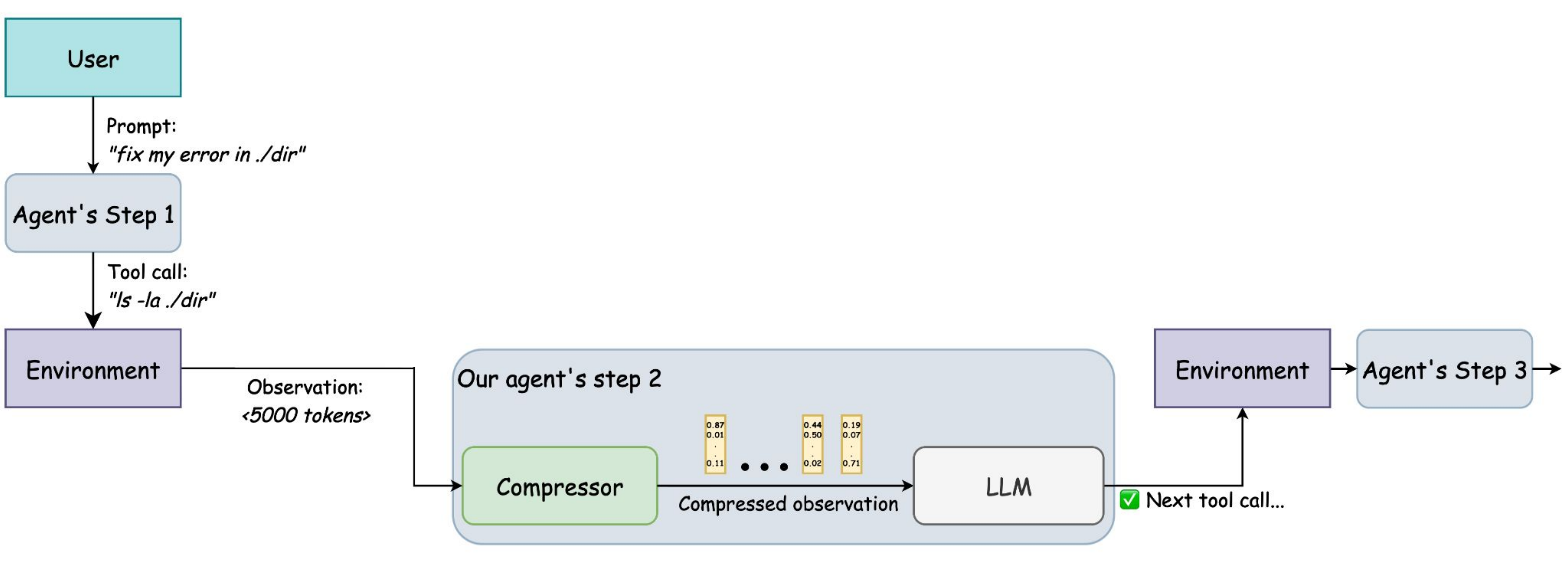

User
Prompt:
"fix my error in ./dir"
Agent's Step 1
Tool call:
"ls -la ./dir"
Environment
Observation:
<5000 tokens>

Example



Quadratic complexity problem
User
Prompt:
"fix my error in ./dir"
Agent's Step 1
Tool call:
"ls -la ./dir"
Environment
Observation:
<5000 tokens>
Base agent's step 2
5000 tokens
LLM
Error:
Out of context length!

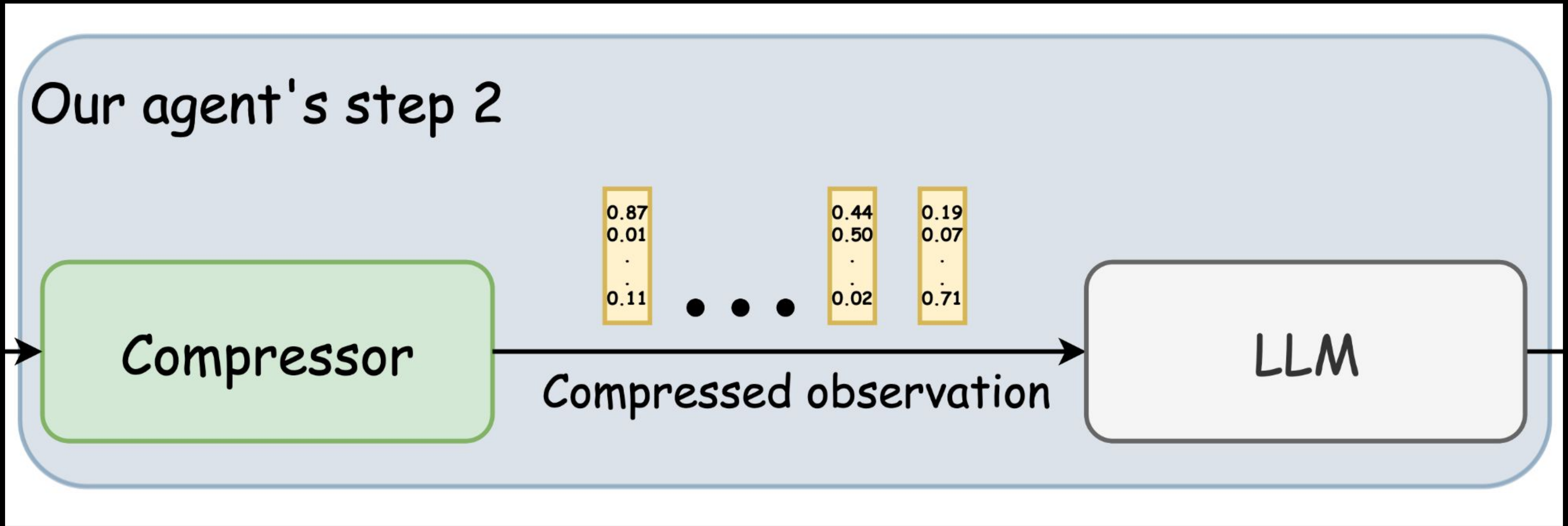
Example



User
Prompt:
"fix my error in ./dir"
Agent's Step 1
Tool call:
"ls -la ./dir"
Environment
Observation:
<5000 tokens>
Our agent's step 2
Compressor
0.87
0.01
0.11
0.44
0.50
0.02
0.19
0.07
0.71
Compressed observation
LLM
✅ Next tool call...
Environment
Agent's Step 3

Why bother with implicit condensation?

- Embeddings can hold lots of information
- Theoretically can be faster
- Avoids external retrieval systems



Research Questions

RQ1: How does implicit context condensation influence the efficiency of LLM-based agents when applied to software engineering tasks?

RQ2: How does the performance of implicit context condensation on standard NLP benchmarks transfer to SWE tasks, single-shot and agentic?

Alternative Approaches

- Extended context windows
    - Infini-attention Transformer [1]
        - ✅ 100K–1M token contexts
        - ❌ New architecture + re-training
- Explicit compression
    - LLMLingua-2 (prompt pruning) [2], SlimCode (code pruning) [3]
        - ✅ Model-agnostic, cheaper & faster
        - ❌ Lossy by design
- Soft-tokens
    - CoCoNut (soft-tokens for reasoning) [4]
        - ✅ Faster reasoning via continuous vectors + inspiring
        - ❌ Applied to reasoning tokens, uses RL

The focus – Implicit compression

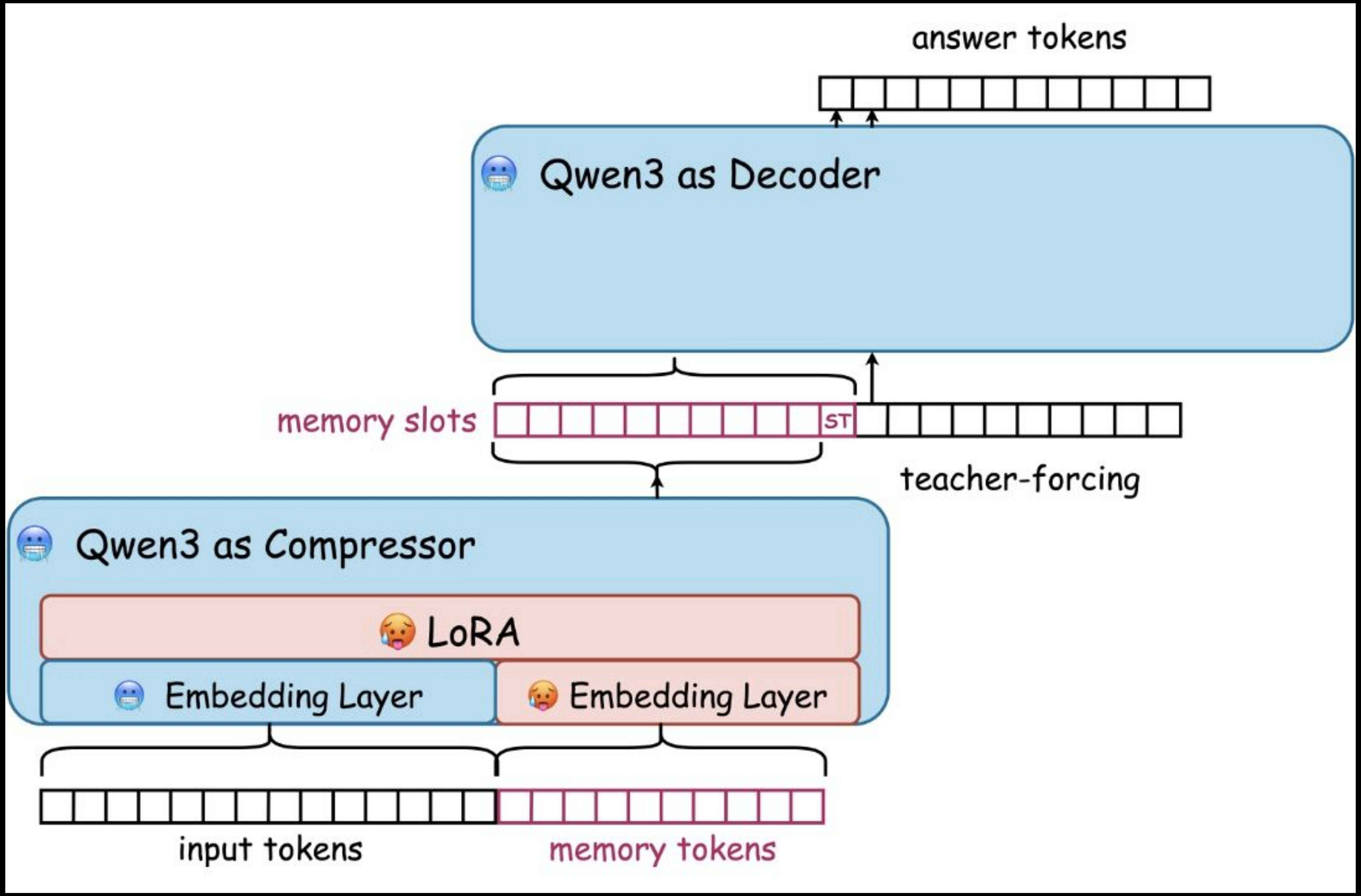
✅ Continuous, theoretically near-lossless compression

✅ Plug-in to existing LLMs, can be fine-tuned on coding

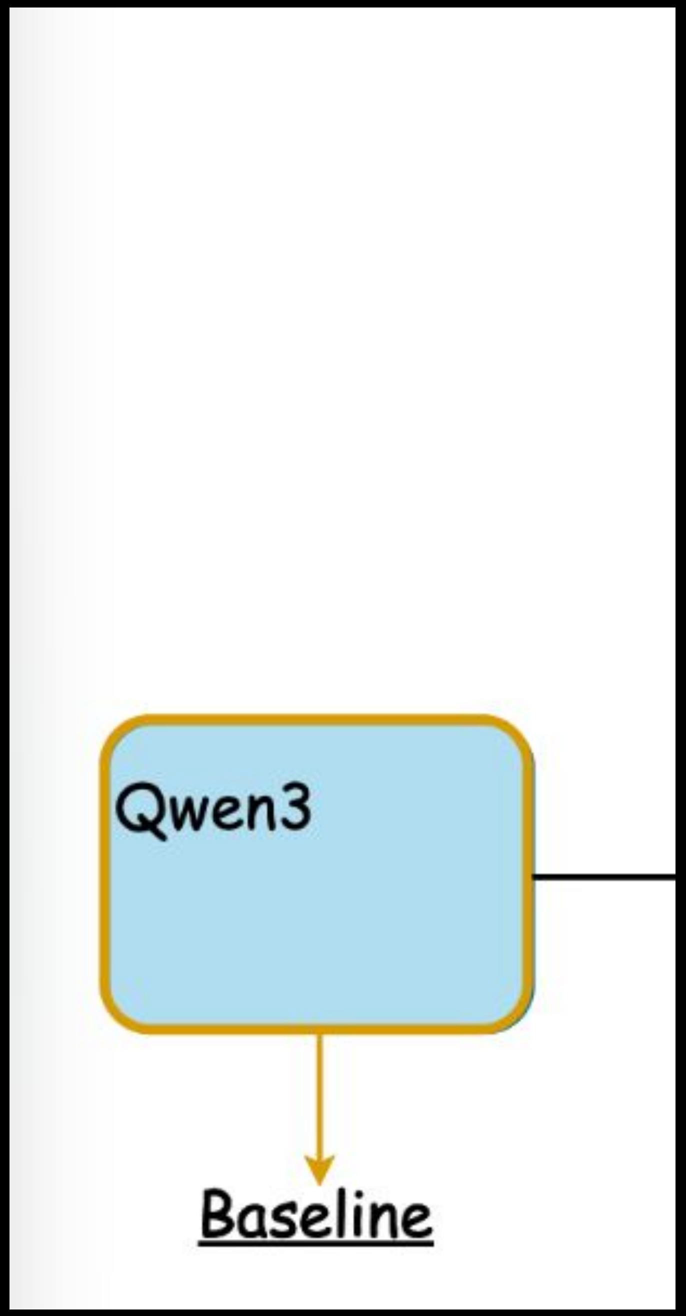
Methodology

In-Context AutoEncoder (ICAE) [5]

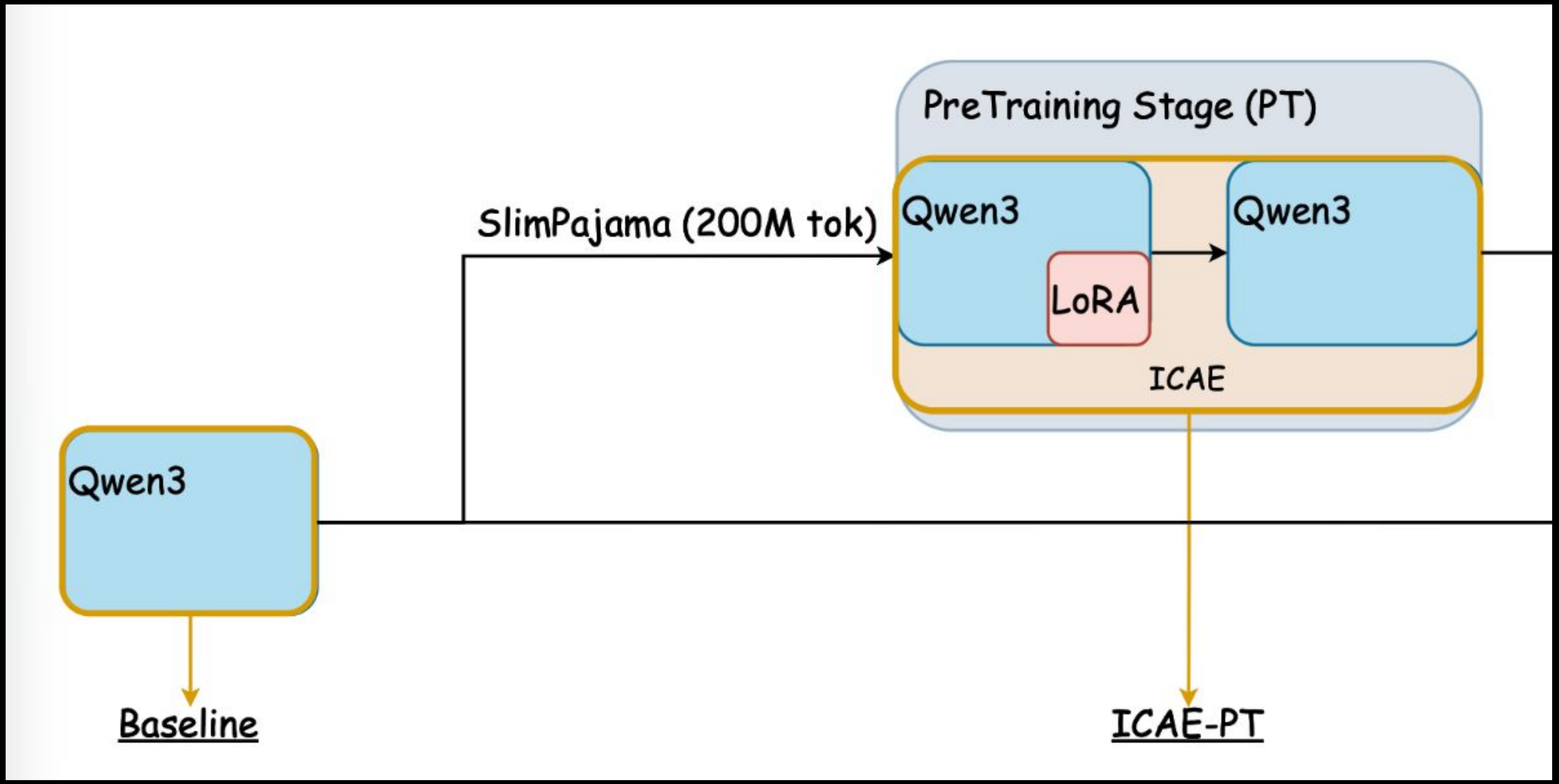
- Same LLM as both compressor and decoder
- Encoder (LoRA) writes full context into a few learned memory slots
- Decoder reads only memory slots + Special Token (ST = AE or LM)
- Trained first for reconstruction, then FT for tasks



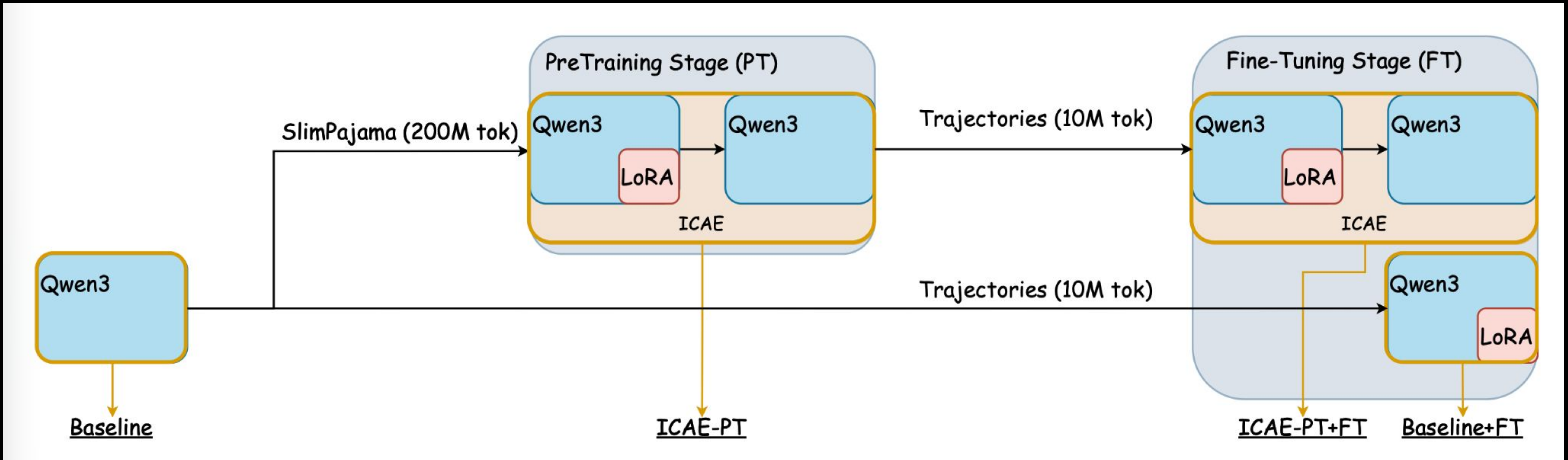
Training process & model names



Training process & model names



Training process & model names



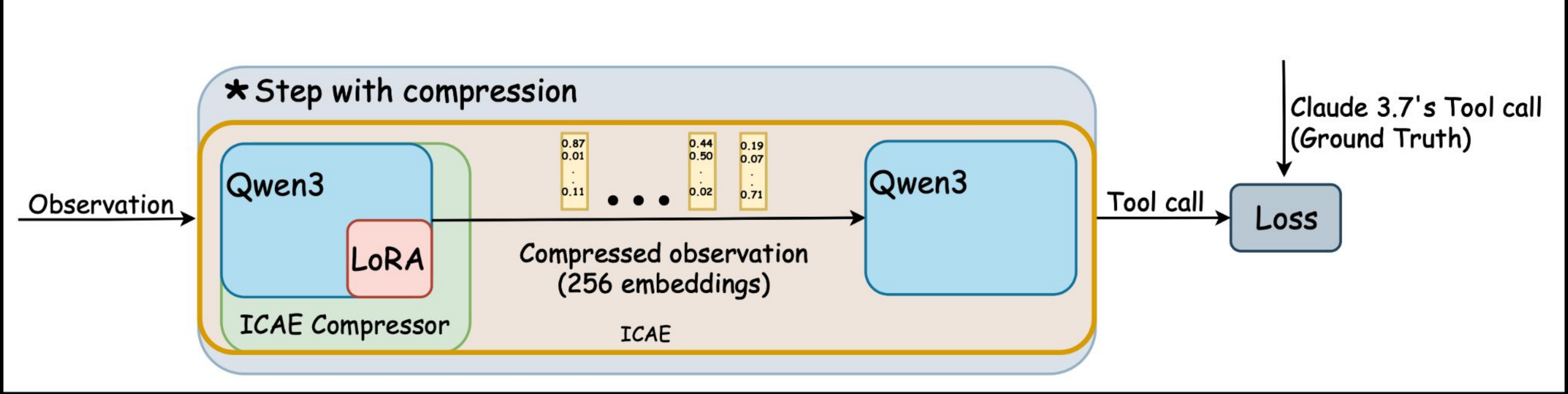
Pre-Training on general texts

- General texts dataset (SlimPajama6B[6])

- 50% AE and 50% LM task

- Train only the encoder's LoRA weights so its memory slots let the frozen LLM reconstruct or continue the text

Fine-Tuning (incl. on trajectories)



Experiments

Metrics

- BLEU – reconstruction / solution quality based on n-gram precision [8]
- Resolve rate – fraction of tasks successfully solved
- Time – mean time per generated tool-call
- # Turns – interaction steps needed to reach a solution

Pretraining

- 4x compression

Autoencoding Reconstruction Failure Example

```
Original:
<p align="center">
<a href="https://swe-agent.com/latest/">
<strong>Documentation</strong></a> 
...

Reconstructed:
<p align="center">
<a href="https://swe-agent.com/agent/latest/">
<strong>Documentation</strong></a> 
...
```

Pretraining (AE scores)

Run	Checkpoint (# steps)	Compression	BLEU (mean, n=100)
Qwen3-8B (no ICAE)	18k	×1	0.867
10M tokens subset			
`ICAE-PT`	9k	×4	0.909
`ICAE-PT`	12k	×4	<u>0.942</u>
`ICAE-PT`	18k	×4	0.902
1B tokens subset			
`ICAE-PT`	9k	×4	0.936
`ICAE-PT`	12k (main)	×4	**0.964**
`ICAE-PT`	18k	×4	0.928

SQuAD [9]

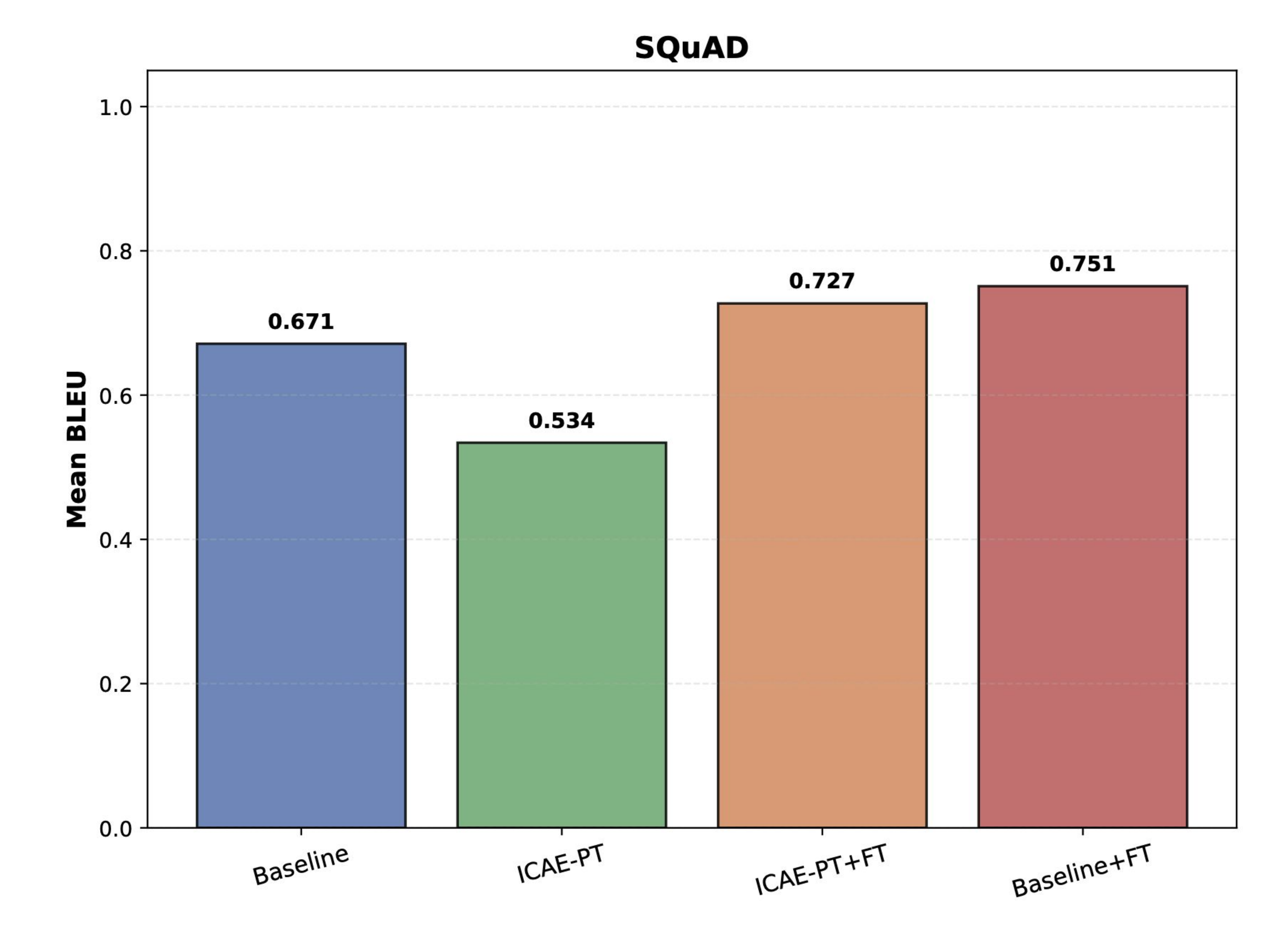
Single-step QA
(dates, names, etc)

In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under **gravity**. The main forms of precipitation include drizzle, rain, sleet, snow, **graupel** and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals **within a cloud**. Short, intense periods of rain in scattered locations are called "showers".

What causes precipitation to fall?
gravity

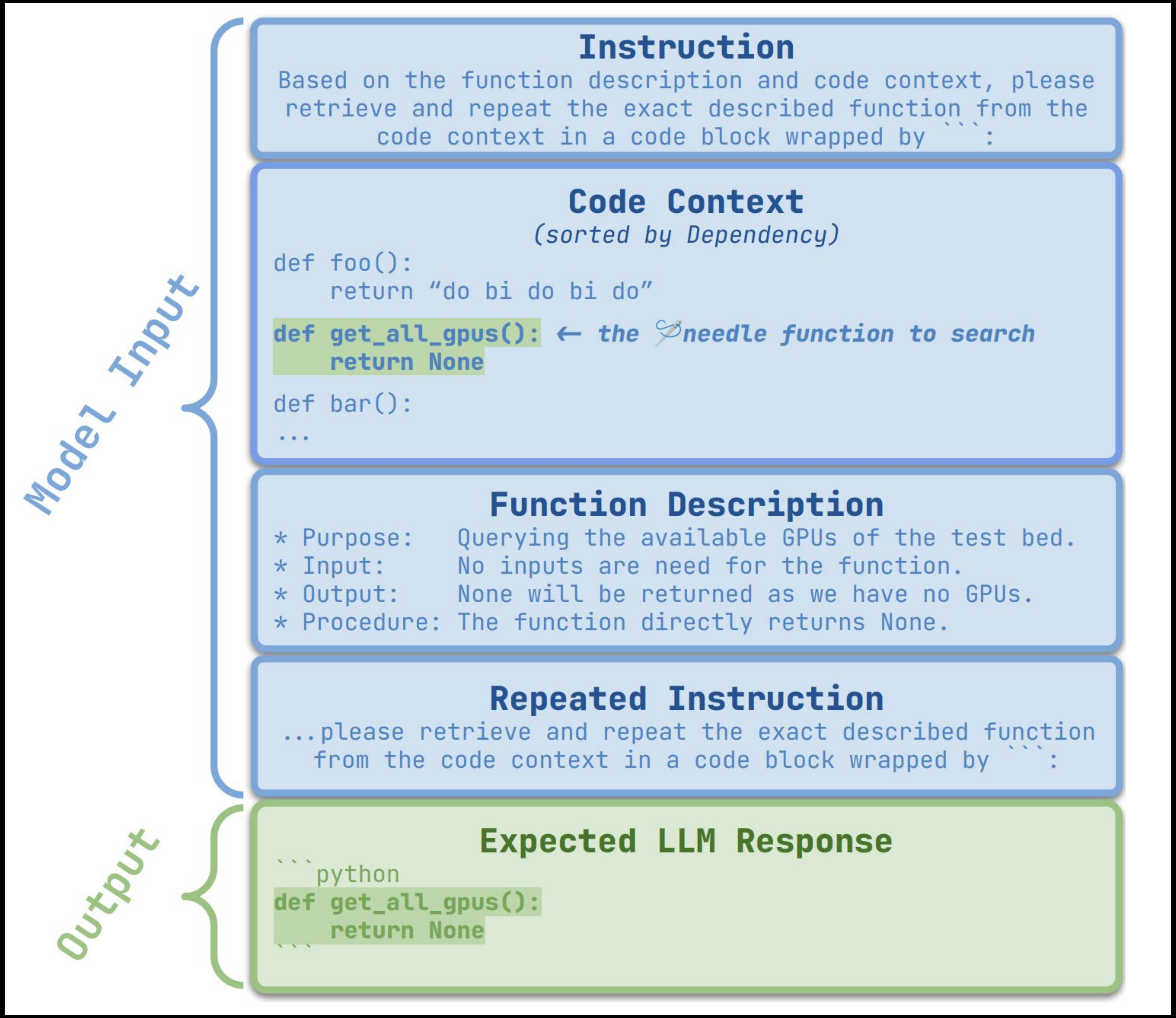
What is another main form of precipitation besides drizzle, rain, snow, sleet and hail?
graupel

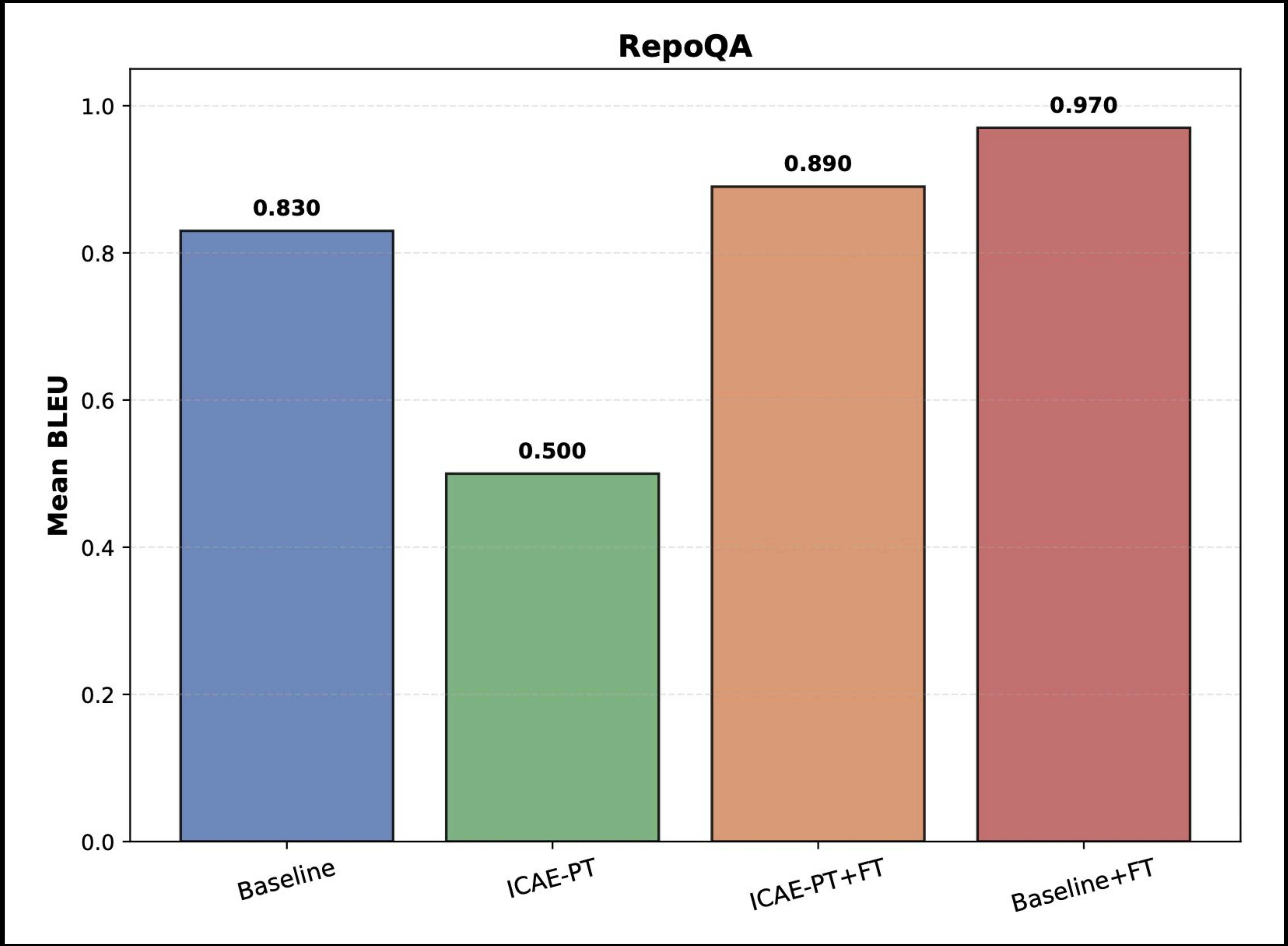
Where do water droplets collide with ice crystals to form precipitation?
within a cloud


SQuAD
Mean BLEU
1.0
0.8
0.6
0.4
0.2
0.0
0.671
0.534
0.727
0.751
Baseline
ICAE-PT
ICAE-PT+FT
Baseline+FT

RepoQA [10]

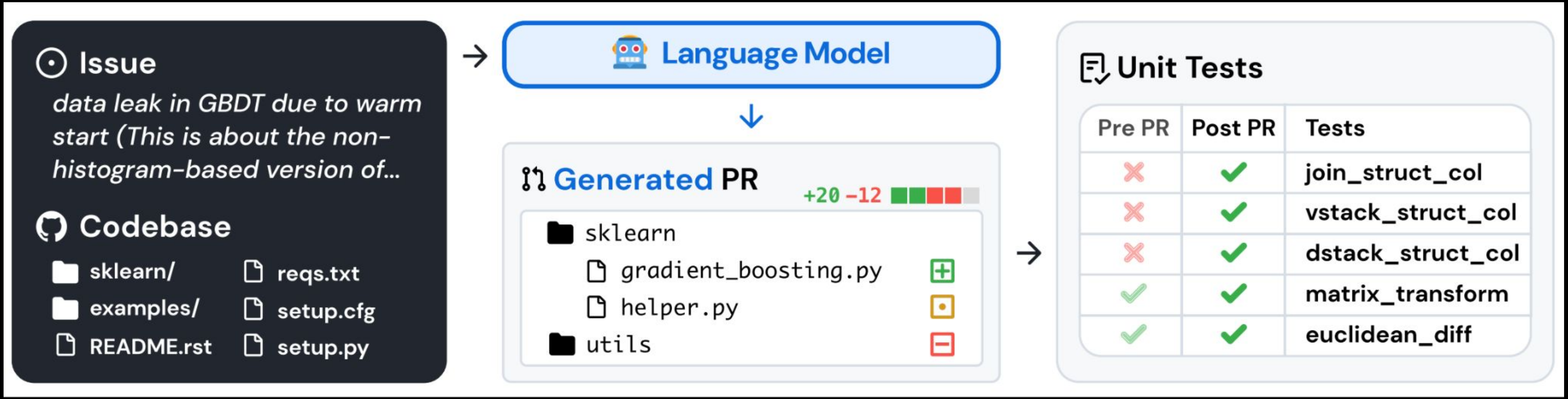
Single-step
code reconstruction
(needle in a haystack)

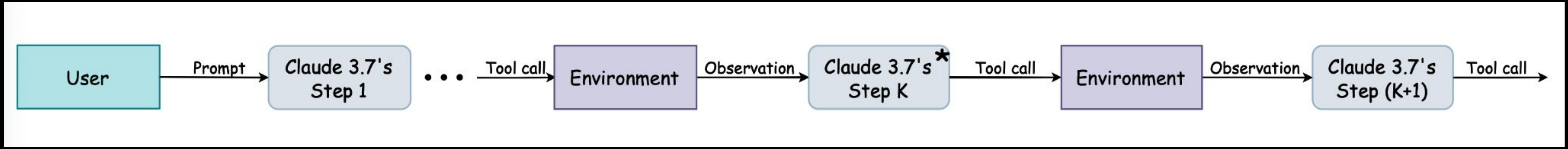



RepoQA
Mean BLEU
1.0
0.8
0.6
0.4
0.2
0.0
0.830
0.500
0.890
0.970
Baseline
ICAE-PT
ICAE-PT+FT
Baseline+FT

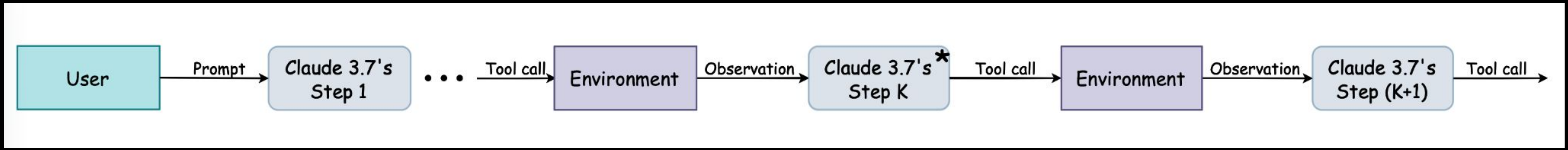
SWE-Bench Verified [11]

Agentic SWE





- Observations of length >256 tokens are compressed



Research Questions

RQ1: How does implicit context condensation influence the efficiency of LLM-based agents when applied to software engineering tasks?

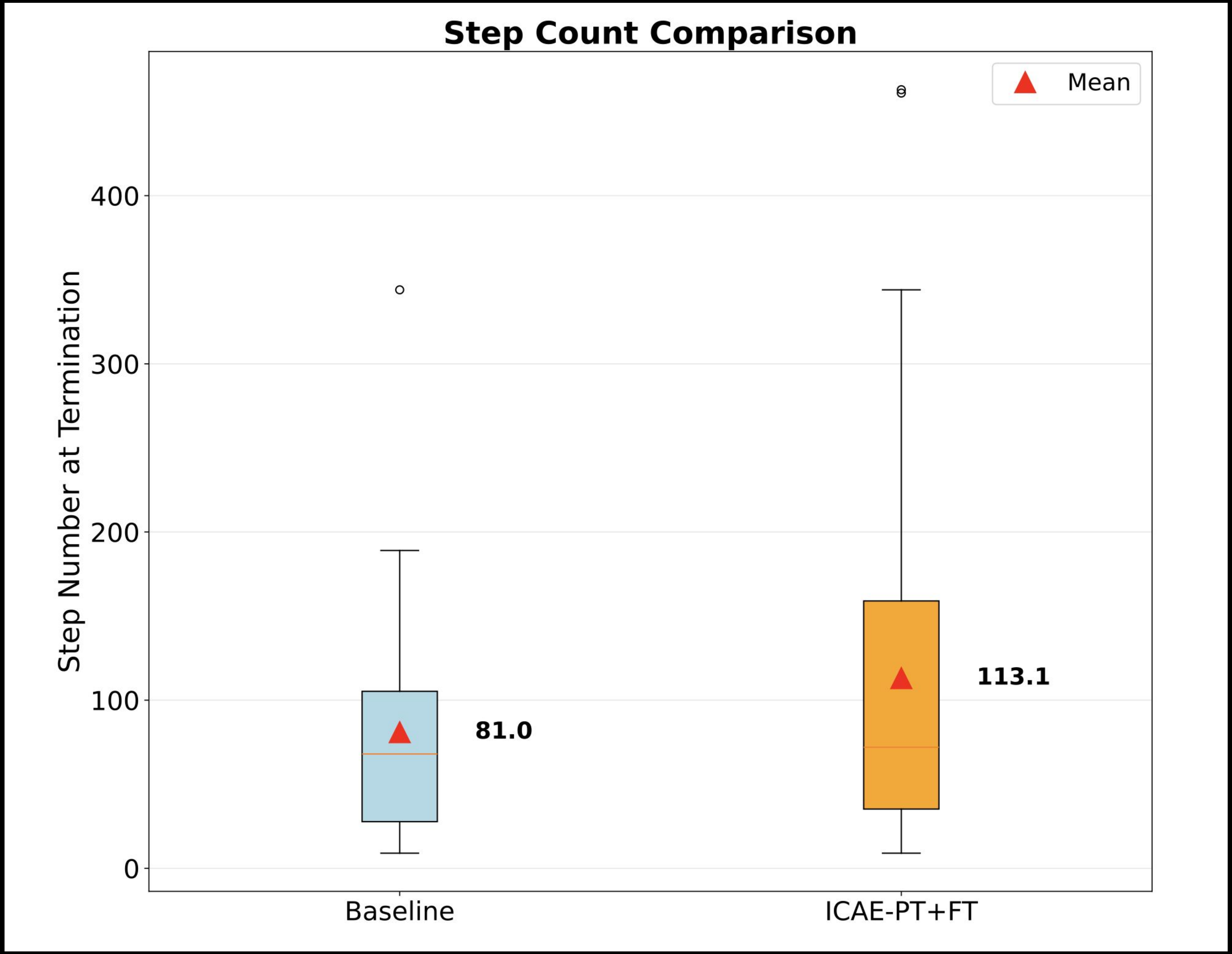
Time

Name	Time (s) ↓
Naive (del long obs-s)	0.44
Baseline+FT	1.24
Baseline	1.23
ICAE-PT	1.23
ICAE-PT+FT	**1.12 (0.31+0.81)**

Max # steps: 75
Max context len: 32,768 tokens

Steps

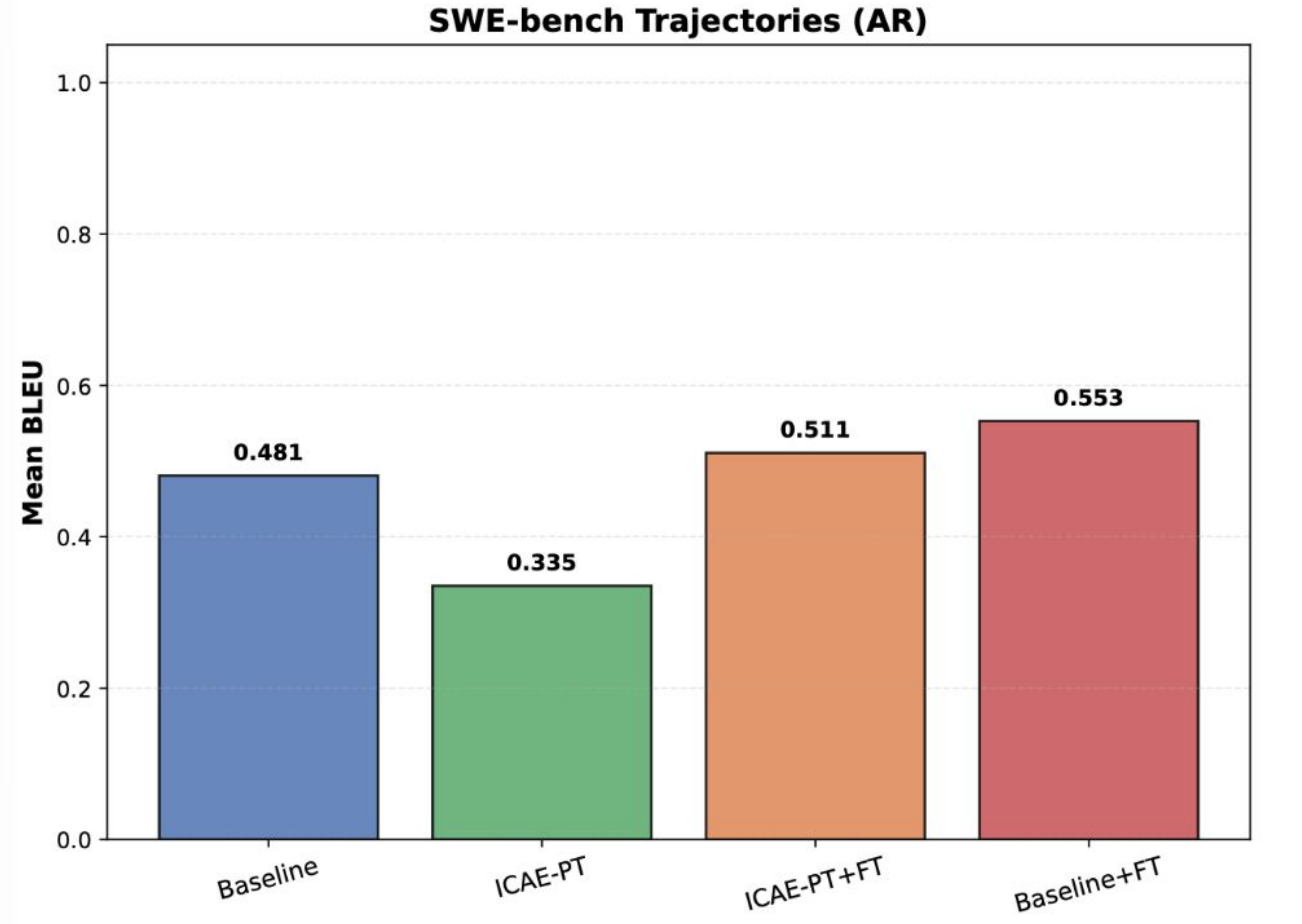
Max # steps: –
Max context len: 32,768 tokens



Research Questions

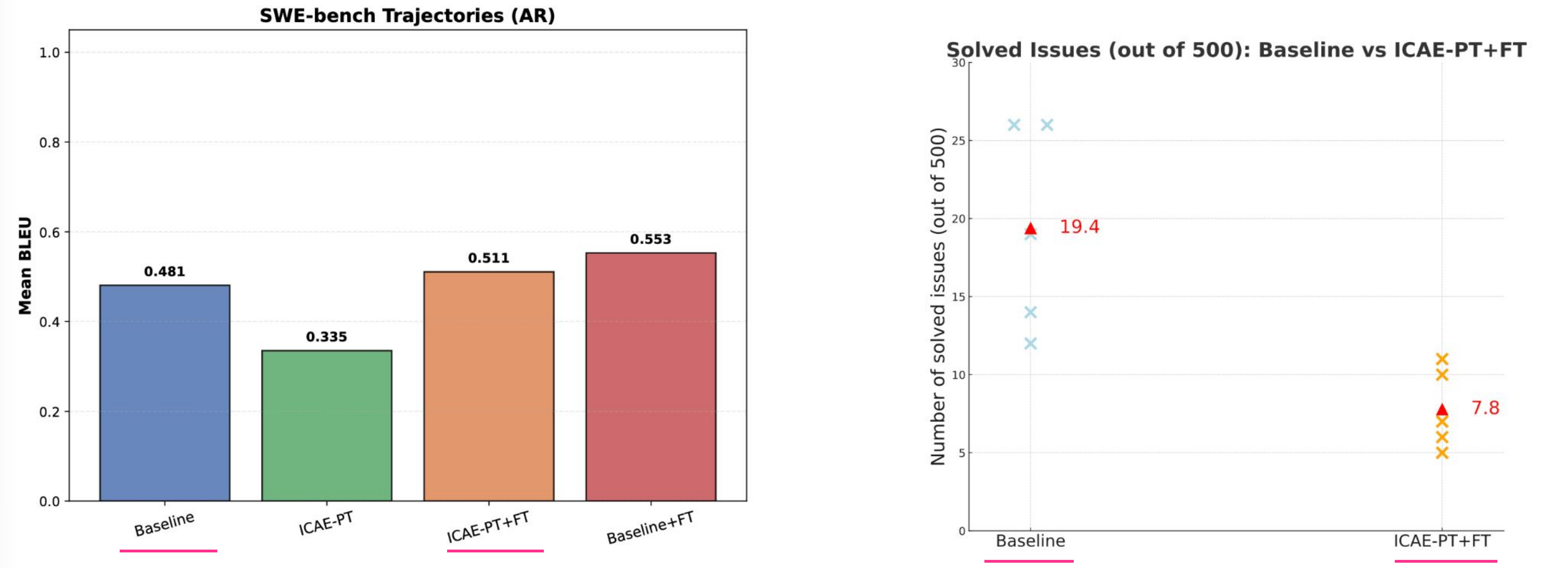
RQ2: How does the performance of implicit context condensation on standard NLP benchmarks transfer to SWE tasks, single-shot and agentic?

BLEU



(a) BLEU scores on SWE-bench

BLEU vs Resolve Rate



(a) BLEU scores on SWE-bench

(b) Resolve rate comparison

Discussion & Future Work

Reconstruction fidelity & cascading errors

- Slightly imperfect BLEU → small bugs can accumulate in long trajectories

Future steps:

- Improve pretraining BLEU & tasks
- Explore other compression ratios

Autoencoding Reconstruction Failure Example

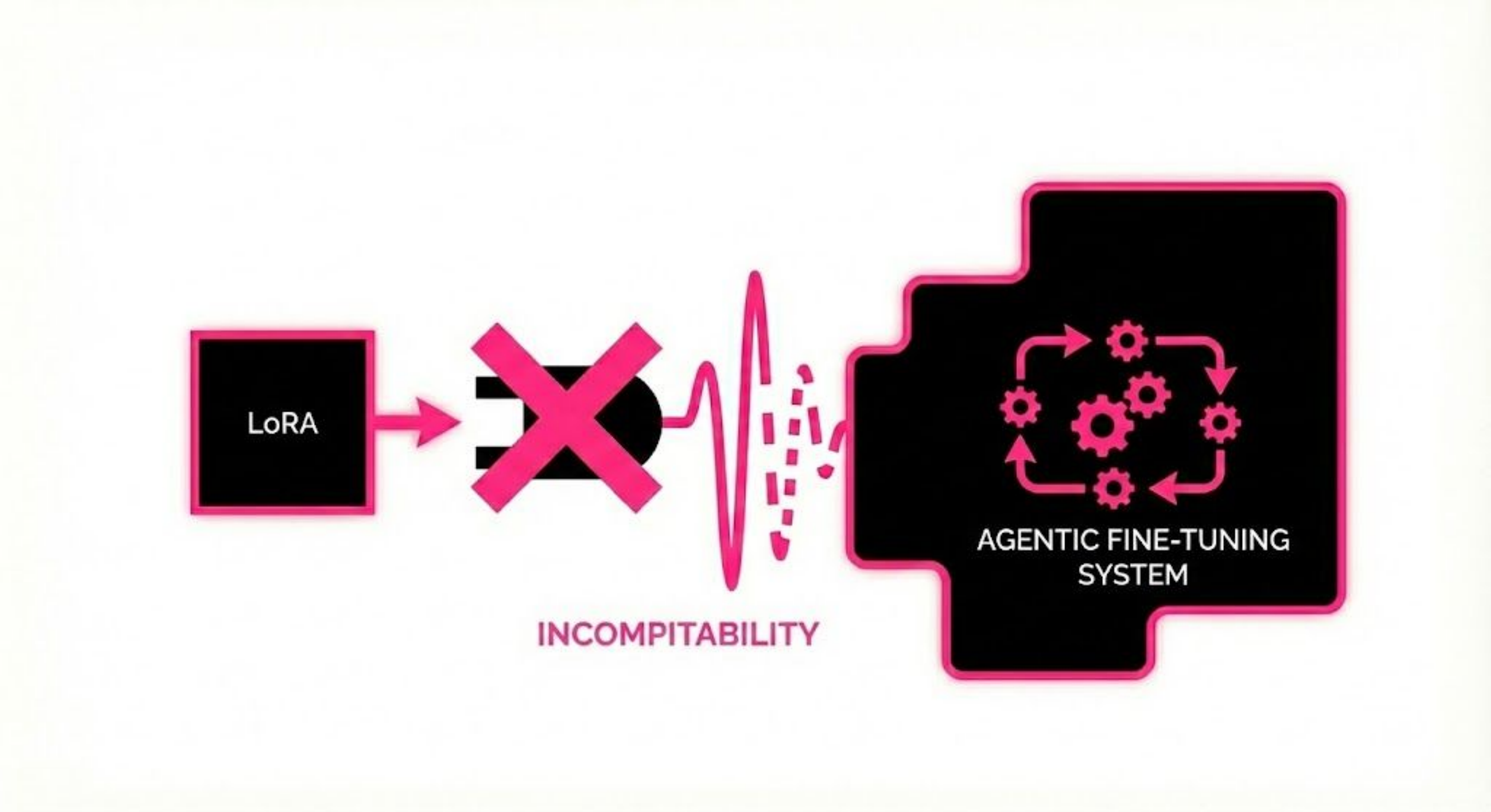
```
Original:
<p align="center">
<a href="https://swe-agent.com/latest/">
<strong>Documentation</strong></a> 
...

Reconstructed:
<p align="center">
<a href="https://swe-agent.com/agent/latest/">
<strong>Documentation</strong></a> 
...
```

LoRA does not work with trajectories-FT

- Low-rank approximations may learn teacher style, not robust policies

Future steps:

- Use full-model training
- Explore RL-style optimization with direct rewards from SWE-bench

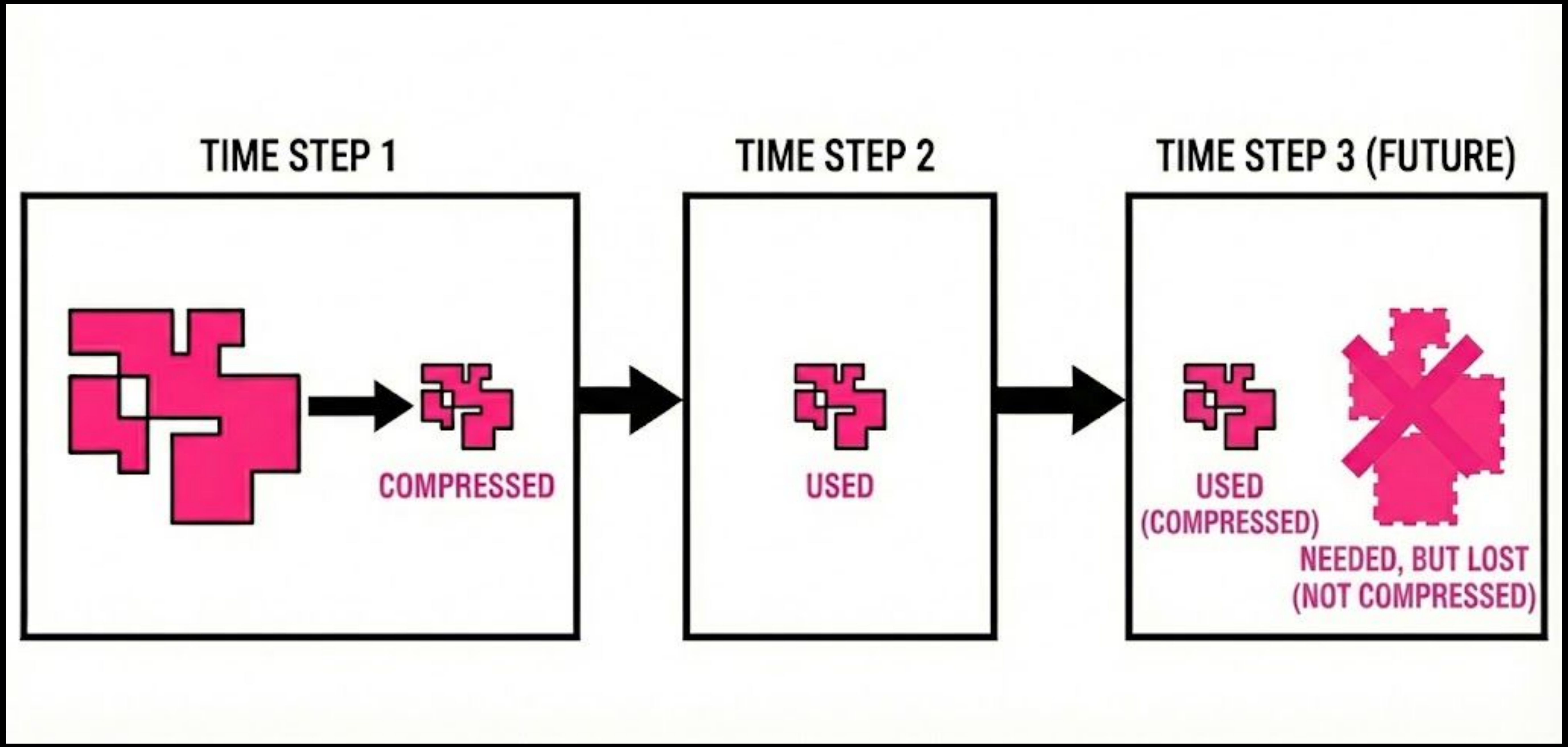


Information preserving for multi-turn interactions

- Per-step compression + single-step loss → throw away details that matter many steps later

Future steps:

- Train over multiple steps
- Use trajectory-level objectives (e.g., RL) to retain information critical for later steps

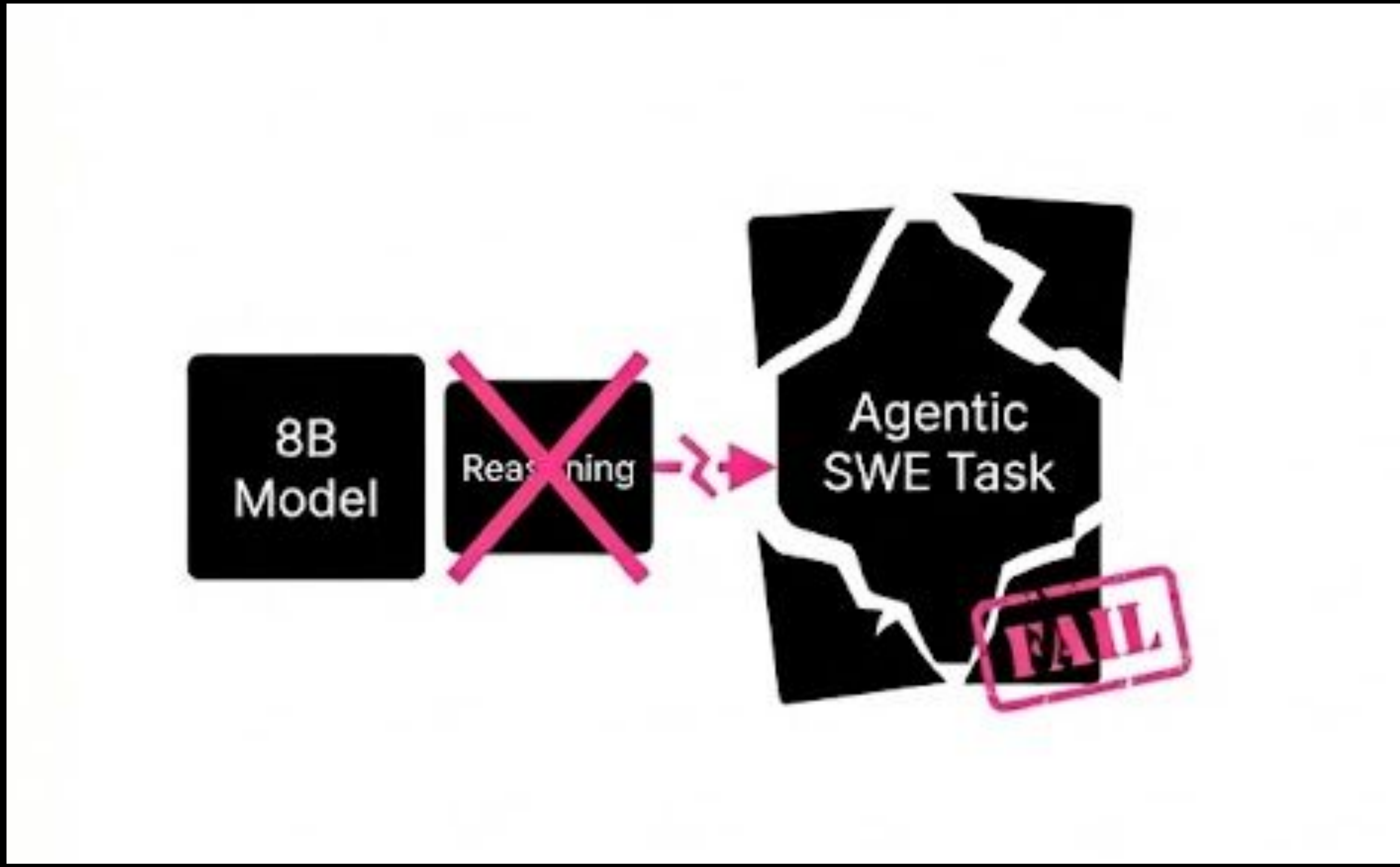


Model size and reasoning

- Only 8B-sized model without reasoning might be insufficient for multi-turn agentic SWE

Future steps:

- Train model of sizes 32B+
- Try FT on teacher's model reasoning as well



Conclusion

- RQ1 – Efficiency: ICAE-based compression enables longer trajectories (by ≈40%) and tool-call speedups (by ≈10%) under a fixed context window.

- RQ2 – Transferability: Compression transfers positively to QA and single-shot SWE tasks, but degrades agentic SWE performance.

Let's discuss! 🧐

References

[1] T. Munkhdalai, M. Faruqui, and S. Gopal, “Leave no context behind: Efficient infinite context transformers with infini-attention,” arXiv:2404.07143, 2024.

[2] Z. Pan et al., “LLMLingua-2: Data distillation for efficient and faithful task-agnostic prompt compression,” arXiv:2403.12968, 2024.

[3] Y. Wang et al., “Natural is the best: Model-agnostic code simplification for pre-trained large language models,” Proc. ACM on Software Engineering 1(FSE), pp. 586–608, 2024.

[4] CoCoNut – not listed in the thesis bibliography; please insert your preferred CoCoNut citation here.

[5] T. Ge et al., “In-context autoencoder for context compression in a large language model,” arXiv:2307.06945, 2023.

[6] M. Weber et al., “RedPajama: An open dataset for training large language models” (used as the SlimPajama-6B pretraining corpus), Advances in Neural Information Processing Systems 37, 2024.

[7] J. Yang et al., “SWE-Smith: Scaling data for software engineering agents,” arXiv:2504.21798, 2025.

[8] K. Papineni et al., “BLEU: A method for automatic evaluation of machine translation,” in Proc. 40th Annual Meeting of the ACL, pp. 311–318, 2002.

[9] P. Rajpurkar et al., “SQuAD: 100,000+ questions for machine comprehension of text,” arXiv:1606.05250, 2016.

[10] J. Liu et al., “RepoQA: Evaluating long context code understanding,” arXiv:2406.06025, 2024.

[11] C. E. Jimenez et al., “SWE-bench: Can language models resolve real-world GitHub issues?” arXiv:2310.06770, 2023.

Backup Slides

Name in our paper	**Encoder**	**Decoder**	**Resolved (/500)** ↑	**Time (s)** ↓
Naive Baselines				
—	del long obs-s	Qwen	1	0.44
—	del all obs-s	Qwen	0	0.39
Baselines				
—	—	Qwen (Full-FT)	**86**	1.24
Baseline+FT	—	Qwen (LoRA-FT)	10	1.24
Baseline	—	Qwen	<u>19.4 ± 6.5</u>	1.23
ICAE Compression (ours)				
ICAE-PT	ICAE (LoRA-PT)	Qwen	2	1.23
ICAE-PT+FT	ICAE (LoRA-PT & LoRA-FT)	Qwen	7.8 ± 2.59	**1.12 (0.31+0.81)**
—	ICAE (LoRA-PT & LoRA-FT)	Qwen (LoRA-FT)	10	<u>1.13</u>

```
---- BEGIN FUNCTION #1: bash ----
Description: Execute a bash command in the terminal.

Parameters:
  (1) command (string, required): The bash command to execute. Can be empty to view additional logs when previous exit code is
  `-1`. Can be `ctrl+c` to interrupt the currently running process.
---- END FUNCTION #1 ----

---- BEGIN FUNCTION #2: submit ----
Description: Finish the interaction when the task is complete OR if the assistant cannot proceed further with the task.
No parameters are required for this function.
---- END FUNCTION #2 ----

---- BEGIN FUNCTION #3: str_replace_editor ----
Description: Custom editing tool for viewing, creating and editing files
* State is persistent across command calls and discussions with the user
* If `path` is a file, `view` displays the result of applying `cat -n`. If `path` is a directory, `view` lists non-hidden files
and directories up to 2 levels deep
* The `create` command cannot be used if the specified `path` already exists as a file
* If a `command` generates a long output, it will be truncated and marked with `<response clipped>`
* The `undo_edit` command will revert the last edit made to the file at `path`

Notes for using the `str_replace` command:
* The `old_str` parameter should match EXACTLY one or more consecutive lines from the original file. Be mindful of whitespaces!
* If the `old_str` parameter is not unique in the file, the replacement will not be performed. Make sure to include enough
context in `old_str` to make it unique
* The `new_str` parameter should contain the edited lines that should replace the `old_str`

Parameters:
  (1) command (string, required): The commands to run. Allowed options are: `view`, `create`, `str_replace`, `insert`,
  `undo_edit`.
Allowed values: [`view`, `create`, `str_replace`, `insert`, `undo_edit`]
  (2) path (string, required): Absolute path to file or directory, e.g., `/repo/file.py` or `/repo`.
  (3) file_text (string, optional): Required parameter of `create` command, with the content of the file to be created.
  (4) old_str (string, optional): Required parameter of `str_replace` command containing the string in `path` to replace.
  (5) new_str (string, optional): Optional parameter of `str_replace` command containing the new string (if not given, no string
  will be added). Required parameter of `insert` command containing the string to insert.
  (6) insert_line (integer, optional): Required parameter of `insert` command. The `new_str` will be inserted AFTER the line
  `insert_line` of `path`.
  (7) view_range (array, optional): Optional parameter of `view` command when `path` points to a file. If none is given, the full
  file is shown. If provided, the file will be shown in the indicated line number range, e.g., [11, 12] will show lines 11 and
  12. Indexing at 1 to start. Setting [start_line, -1] shows all lines from start_line to the end of the file.
---- END FUNCTION #3 ----
```