# Beyond the Context Window:
# Scaling Agentic RL via End-to-end Optimized Context Compression

Miao Lu[1,2] Weiwei Sun[1,3] Weihua Du[1,3] Zhan Ling[1] Xuesong Yao[1] Kang Liu[1] Jiecao Chen[1]

[1]ByteDance Seed [2]Stanford University [3]Carnegie Mellon University

## TL;DR

We train LLM agents *beyond their context window* by making **summarization a learnable part of the policy**. Our algorithm `SUPO` jointly optimizes **tool use** and **what/when to summarize**, scaling RL to long horizons with a **small, fixed working context**.

## The Context Bottleneck in Agentic RL

Long-horizon agents may issue *hundreds* of tool calls; the context (prompt + outputs + tool observations + reasoning) grows *unboundedly*, creating three barriers:

1. **Degraded instruction following** on long contexts ⇒ fewer successful rollouts.
2. **Excessive rollout cost** ⇒ rollout becomes the training bottleneck.
3. **Hard context limit** ⇒ the window caps the RL *horizon*.

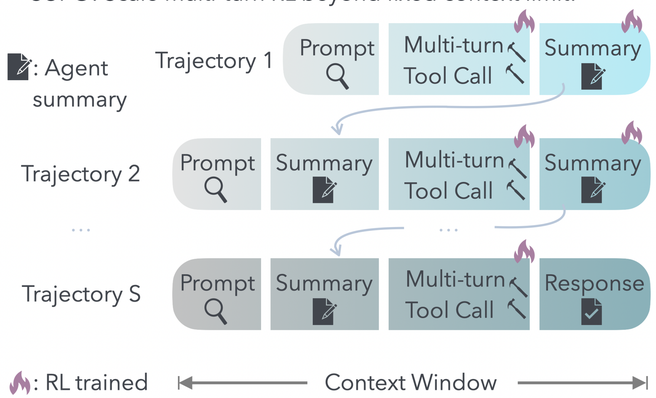
**How to train RL agents on tasks needing more information than fits in one window?**

## Our Idea: Summarization-based Context Management

Periodically **compress** the tool-use history into a concise *LLM-generated summary*, then reset the working state to `prompt + summary`. The summary is **not rule-based** — it is *learned jointly with the policy*.

**RL with Summarization-based Context Management**

SUPO: Scale multi-turn RL beyond fixed context limit!




## Key Findings

- **Beyond the window:** `SUPO` reaches a **320K** effective context using only a **32K** working window.
- **More interaction ⇒ deeper exploration:** tool calls rise sharply (27.8 → 74.9 on `SWE-Bench`).
- **Test-time scaling:** extra summary rounds (beyond training) lift `BrowseComp-Plus` to **60.0**%, best of all methods.

## Summarization-Augmented MDP $\mathcal{M}_{\mathcal{V}}^{\texttt{sum}}$

Multi-turn tool use as an MDP (ReAct-style [1]): state $s_t$ accumulates tokens; action $a_t \sim \pi_{\boldsymbol{\theta}}(\cdot|s_t)$; observation $o_t$; verifiable reward $R = 1$ iff the final answer passes; objective $\max_{\boldsymbol{\theta}} \mathbb{E}[R(s_T, a_T)]$. **Key change:** a threshold $L$ reshapes the transition:

$$s_{t+1} = \begin{cases} (s_t, a_t, o_t) & \text{append, if } L_t < L, \\ (s_t, a_t, o_t, v_{\texttt{sum}}) & \text{trigger summary, if } L_t \geq L, \\ (s_1,\ a_t) & \text{reset to prompt + summary.} \end{cases}$$

**Bounded working context:** $|s_t|+|a_t| \leq L+2L_{\mathcal{A}}+L_{\mathcal{O}}+|v_{\texttt{sum}}|$, yet the **effective horizon** $L_{\texttt{effect}} = L_{\texttt{RL}} \times (S+1)$ grows with the number of summaries $S$.

## Policy Gradient of $\mathcal{M}_{\mathcal{V}}^{\texttt{sum}}$

A rollout splits at summary indices $\{t_i\}$ into $I+1$ *summarized sub-trajectories*. Its exact policy gradient is:

$$\partial_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}\left[ R(s_T, a_T) \sum_{i=1}^{I+1} \sum_{t=t_{i-1}+1}^{t_i} \partial_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}\Big( a_t \,\Big|\, \underbrace{s_1, a_{t_{i-1}}}_{\text{prompt + prev. summary}}, a_{t_{i-1}+1}, o_{t_{i-1}+1}, \dots \Big) \right].$$

Within a block, actions $a_t$ $(t < t_i)$ optimize **tool calling / reasoning**; the terminal action $a_{t_i}$ optimizes the **summary**.

**Why it matters:** the long-horizon gradient *decomposes into a sum of ordinary single-trajectory gradients* ⇒ plugs directly into standard LLM-RL infrastructure (e.g. VeRL / GRPO).

## `SUPO`: Summarization-Augmented Policy Optimization

A GRPO-style clipped objective over *all* sub-trajectories of $G$ rollouts:

$$\mathcal{J}_{\texttt{SUPO}}(\boldsymbol{\theta}) = \widehat{\mathbb{E}}\left[ \frac{1}{Z} \sum_{j=1}^{G} \sum_{i=1}^{I^j+1} \sum_{t=t_{i-1}^j+1}^{t_i^j} \sum_{\ell=1}^{\ell_t^j} \min\Big\{ \rho_{t,\ell}^j \widehat{A}^j,\ \texttt{Clip}_{\epsilon_{\text{low}}}^{\epsilon_{\text{high}}}(\rho_{t,\ell}^j) \widehat{A}^j \Big\} \cdot \underbrace{\mathbf{1}\{T^j \leq H,\ I^j \leq S\}}_{\text{overlong mask}} \right]$$

with $Z = \sum_{j,i,t} |a_t^j|$, token importance ratio $\rho$, and a *group-relative* advantage shared across a rollout's tokens:

$$\rho_{t,\ell}^j = \frac{\pi_{\boldsymbol{\theta}}(v_{t,\ell}^j \mid s_t, v_{t,<\ell}^j)}{\pi_{\texttt{old}}(v_{t,\ell}^j \mid s_t, v_{t,<\ell}^j)}, \qquad \widehat{A}^j = \frac{R^j - \texttt{mean}(\{R^{j'}\}_{j'=1}^G)}{\texttt{std}(\{R^{j'}\}_{j'=1}^G)}.$$

## Three Design Choices that Make it Work

1. **Trajectory management.** Treat each rollout as $I^j+1$ standard trajectories (each `prompt + prev. summary` → current summary) ⇒ *reuse single-trajectory RL infra.*
2. **Group-relative advantage.** Share one $\widehat{A}^j$ (computed *inside the rollout group*) across sub-trajectories — beats per-trajectory normalization, which *dilutes* long successes.
3. **Overlong masking.** Mask rollouts that miss the answer within $H$ steps / $S$ summaries. Without it, the *summarization pattern collapses* and conditional success → 0.

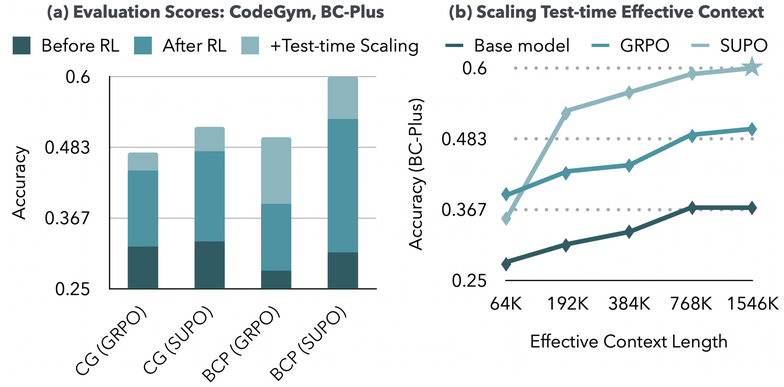
## Main Results: Three Domains

**Setup.** `CodeGym` [4] (function calling, `Qwen2.5-32B`); `BrowseComp-Plus` [5] (deep research, `Seed-OSS-36B`); `SWE-Bench-Verified` [6] (coding agent, `Seed-OSS-36B`).

| | CodeGym | | | | |
|---|---|---|---|---|---|
| **Algorithm** | **Work. len.** | **Eff. len.** | **Acc. before** | **Acc. after** | **Tool call** |
| `GRPO` | 32K | 32K (32K*1) | 32.0% | 44.5% | 52.1 |
| `SUPO` (w/o over. m.) | 4K | 32K (4K*8) | 32.8% | 45.3% (+0.8%) | 52.3 |
| `SUPO` (per-traj. adv.) | 4K | 32K (4K*8) | 32.8% | 42.1% (-2.4%) | 47.0 |
| `SUPO` | 4K | 32K (4K*8) | 32.8% | **47.7%** (+3.2%) | 54.7 |
| | **BrowseComp-Plus** | | | | |
| **Algorithm** | **Work. len.** | **Eff. len.** | **Acc. before** | **Acc. after** | **Tool call** |
| `GRPO` | 64K | 64K (64K*1) | 28.0% | 39.0% | 6.7 |
| `SUPO` (w/o over. m.) | 64K | 192K (64K*3) | 31.0% | 44.0% (+5.0%) | 10.7 |
| `SUPO` (per-traj. adv.) | 64K | 192K (64K*3) | 31.0% | 49.0% (+10.0%) | 17.5 |
| `SUPO` | 64K | 192K (64K*3) | 31.0% | **53.0%** (+14.0%) | 19.2 |
| | **SWE-Bench-Verified** | | | | |
| **Algorithm** | **Work. len.** | **Eff. len.** | **Acc. before** | **Acc. after** | **Tool call** |
| `GRPO` | 32K | 32K (32K*1) | 43.6% | 48.0% | 27.8 |
| `SUPO` | 32K | 320K (32K*10) | 46.8% | **55.0%** (+7.0%) | 74.9 |

`SUPO` beats `GRPO` on **all three** domains at **equal/smaller** working context, with up to **3**× more tool calls. Both ablations (overlong masking, advantage) are necessary.

## Evaluation & Test-time Scaling




**(a)** At equal effective context, `SUPO` > `GRPO`. **(b)** Scaling the number of summary rounds *beyond training* keeps improving accuracy; `SUPO` reaches **60.0%** on `BrowseComp-Plus`, the best of all methods.

## Selected References

[1] Yao et al. *ReAct: Synergizing Reasoning and Acting in LMs.* ICLR 2023.
[2] Guo et al. *DeepSeek-R1: Incentivizing Reasoning via RL.* arXiv 2025.
[3] Sheng et al. *HybridFlow: A Flexible RLHF Framework* (VeRL). EuroSys 2025.
[4] Du et al. *Generalizable End-to-end Tool-use RL with CodeGym.* arXiv 2025.
[5] Chen et al. *BrowseComp-Plus: A Transparent Deep-Research Benchmark.* arXiv 2025.
[6] Jimenez et al. *SWE-bench: Can LMs Resolve Real-World GitHub Issues?* ICLR 2024.