Persistent state · screen-based agents

scopion · Discord: scopion_type21

1 · PROBLEM

what and why

Current computer-use agents fail on long-horizon tasks not because they cannot see the screen, but because they cannot maintain and update a consistent internal state. OSWorld 2.0 reports a best binary completion rate of 20.6% across 108 workflows averaging 318 tool calls, with failures concentrated in recovering hidden state, tracking multiple items, and integrating information that arrives mid-task. Yet nearly all such agents reconstruct context from screenshots and text history at every step rather than carrying a learned persistent state. This proposal measures whether a learned persistent visual-spatial state improves long-horizon state tracking, and specifically whether its advantage grows with horizon length. Compute request: 606 short jobs, ~262 GPU-hours total, 4.1 GB peak memory — parallelism, not large HBM. Every experiment runs under hash-sealed preregistration with gates that stop execution; bit-identical reproduction is already demonstrated.

2 · APPROACH

model / method, and why this one

Architecture: screen pixels → small visual frontend → persistent recurrent state → text head + action head. Four memory conditions, evaluated against horizon length H: (A) stateless single frame; (B) stack of the last N frames; (C) text/screenshot history re-fed every step (current GUI agents); (D) learned recurrent state. The claim is about a gradient: D’s relative advantage grows as H increases — measurable at small scale without frontier-scale compute. Base scale is 4M parameters by choice, to run many ablations, seeds, and shortcut audits cheaply. Frontend comparison on WikiText-2 (5 paired seeds) already shows S-Conv at 1.8027 ± 0.0034 BPC vs V (pixels→CNN) at 2.0210 ± 0.0677; whether the gap reflects capacity or convergence is unresolved.

3 · DATASET

source, size, preprocessing

(a) Frontend calibration: WikiText-2 (already run). Document-level split, context/BPTT = 128, character level. (b) Horizon experiment: synthetic document/UI generator. States of K items update at unpredictable moments; after H steps the agent must answer about or act on current state. Size generated on demand. Text rendered to bitmap at fixed font/DPI. Sweepable axes: horizon length H and perceptual difficulty. Ground truth is verifiable at every step.

4 · METRICS

how you’ll know it worked

how you’ll know it worked Frontend work: held-out BPC. Horizon experiment: task accuracy against horizon length — the quantity of interest is the slope of the gap between conditions, not absolute accuracy. Already in operation: preregistration with composite hash sealing (SHA-256), gates that stop execution unless preregistered thresholds are met, and a battery of baselines (blank, shuffled, surface-only, static median, copy-shift, no-memory, hidden-reset, oracle).

5 · COMPUTE

training time, memory, dependencies

~262 GPU-hours across 606 short jobs; peak memory ~4.1 GB. Parallelism matters more than single-accelerator HBM. Individual frontend runs take on the order of ~93 seconds at the 4M-parameter scale.

6 · DELIVERABLES

`model, paper, demo`

- · Preregistered, hash-sealed measurement platform for long-horizon state tracking.
- · Horizon-vs-memory condition curves with reported slopes and confidence intervals.
- · Open code, configs, and bit-identical reproduction scripts.
- · Write-up of whether learned persistent state advantage grows with horizon.

7 · RISK

`what breaks first, and plan B`

what breaks first, and plan B Perception confounds memory: if the visual frontend is weak, memory failures cannot be isolated. Plan B: start from the trivially-easy perceptual corner and only raise difficulty after memory effects are measurable. Small-scale results may not transfer: Plan B treats the work as a measurement platform with preregistered gates, not a capability claim. Negative or inconclusive slopes are publishable outcomes.