Known By Their Actions: Fingerprinting LLM Browser Agents via UI Traces

William Lugoloobi[1]* **Samuele Marro**[2,3] **Jabez Magomere**[1] **Joss Wright**[1] **Chris Russell**[1]
[1]Oxford Internet Institute, University of Oxford
[2]Department of Engineering Science, University of Oxford
[3]Institute for Decentralized AI

Abstract

As LLM-based agents increasingly browse the web on users' behalf, a natural question arises: can websites passively identify which underlying model powers an agent? Doing so would represent a significant security risk, enabling targeted attacks tailored to known model vulnerabilities. Across 14 frontier LLMs and four web environments spanning information retrieval and shopping tasks, we show that an agent's actions and interaction timings, captured via a passive JavaScript tracker, are sufficient to identify the underlying model with up to 96% F1 within a fixed harness. Although fingerprints are harness-specific, with cross-harness transfer falling to near chance, pooling traces across harnesses recovers model identification at 0.74 to 0.81 Macro-F1. We further show that strong classifiers can be trained from roughly one third of the available traces, and that identification improves steadily as more events are observed. Injecting randomised timing delays between actions substantially degrades classifier performance, but does not provide robust protection: a classifier retrained on delayed traces largely recovers performance. We release our code and a labelled corpus of agent traces at `https://github.com/KabakaWilliam/known_actions`.

1 Introduction

LLM-based agents that browse the web and operate computer interfaces on behalf of users are moving rapidly from research prototypes to production [1–3]. As these systems are deployed at scale across live websites, every page they visit becomes a potential observation point, and we show that observation alone is enough to identify the model. This exposes users to potential security risks.

Every agent visit to a website leaves a trace of clicks, scrolls, keypresses, and other *actions* observable to any party controlling the page. Prior work has established that behavioural traces like this distinguish human users from automated clients [4, 5], and that passively collected browser attributes re-identify human users across sessions [6, 7]. These works ask a binary question: human or bot. As LLM-based agents displace scripted automation, we probe deeper: *given that the client is an agent, which model is pulling the strings?* In classical security frameworks, target identification is the first step toward exploitation [8]. Thus for LLM-based agents, knowing the underlying model enables more targeted attacks: an adversary can select from a known set of model-specific jailbreaks or reduce the search space for a white-box adversarial attack [9].

To our knowledge, we are the first to show that the underlying foundation model of a browser agent can be inferred from passive in-page UI traces alone. Concurrent work [10] shows that AI browsing agents can be distinguished from human visitors and from each other using behavioural and network features. Its classes, however, are agent products (e.g., ChatGPT Atlas, Comet, Browser Use), each

*Kindly correspond to `william.lugoloobi@oii.ox.ac.uk`

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

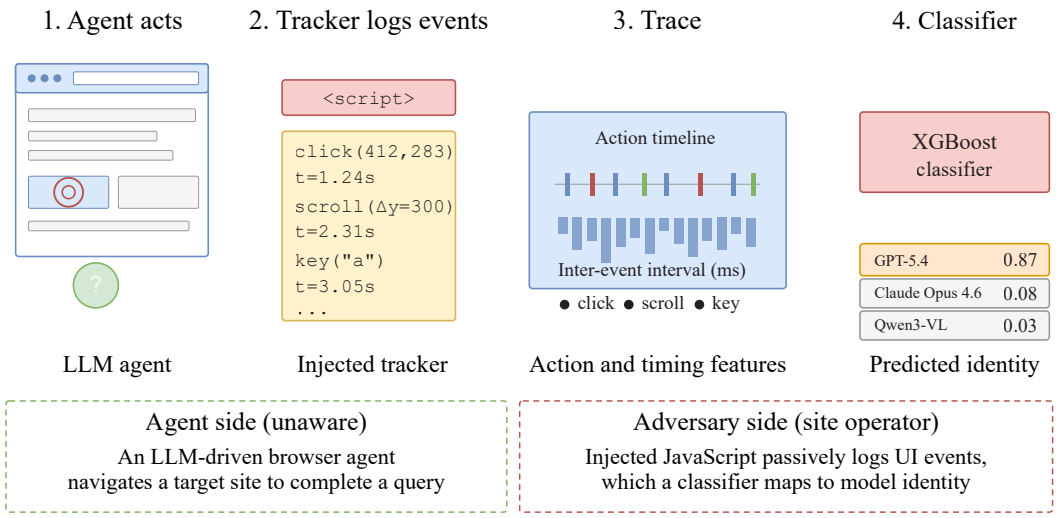


Figure 1: **Overview of our trace collection and threat model.** Using a JavaScript tracker, we collect actions performed by a browsing agent and train a classifier to predict model identity.

running its own harness, and we find that changing the harness alone is enough to break a model classifier (see Section B.6). We address the complementary, subsequent problem of attributing a session already believed to be an agent to a specific model, rather than its harness. Using lightweight classifiers trained on UI action traces collected via injected JavaScript, we achieve agent identification F1 scores of up to 96% across 14 frontier LLMs. The identification does not rely on browser attributes and headers (which can be spoofed), but on the temporal and structural dynamics of how different models navigate, click, and interact with page elements. In other words, the behaviour of a model is sufficient to accurately identify it. For adversaries, this means that agents can be identified and exploited.

The main contributions of this paper are as follows:

- **Agent actions are a fingerprint of model identity.** We demonstrate for the first time that the on-page actions of LLM browser agents encodes the identity of the underlying model, achieving up to 96% classification Macro F1 across 14 frontier models using only behavioural traces collected via passive JavaScript injection.
- **A formalised threat model with defence analysis.** We characterise agent fingerprinting under a passive, co-located adversary and show that the fingerprinter is fast and cheap to update once labelled traces of a new model–harness pair are available. We discuss this more in Appendix B.2. We also test two natural defences, randomised timing delays and prompt normalisation. Both weaken a classifier trained on unperturbed traces, but neither holds up once the adversary retrains on matching traces.
- **Resources for agent fingerprinting.** We release a labelled corpus of agent interaction traces across four web environments and a browser harness compatible with both closed and open-source LLMs, enabling reproducible research into behavioural attribution of LLM agents.

2 Related Work

LLM-based web agents. Autonomous agents that combine language models with browser automation have rapidly moved from research prototypes to production systems [1–3]. Benchmarks such as WebArena [11] and Mind2Web [12] have established standard evaluation environments for these systems. As these agents are deployed at scale, the question of whether a site operator can determine which model is visiting becomes practically consequential for access control, content delivery, and adversarial exploitation.

Bot detection and browser fingerprinting. A large body of work studies how to distinguish automated from human web traffic. Early approaches relied on request-pattern heuristics and crawl detection [13–18]. Subsequent work has shown that fine-grained behavioural signals, such as mouse movements and interaction timing encode neuromotor structure that can reliably separate humans from bots [19, 20, 5, 21]. More recent systems combine behavioural features with server-side logs to detect increasingly sophisticated bots that mimic realistic browser fingerprints [4].

In parallel, browser fingerprinting research has demonstrated that passively collected client-side attributes (e.g., canvas, fonts, WebGL) can be combined into persistent identifiers at scale [6, 7]. Across these lines of work, the problem is typically framed as binary classification: human versus bot. We instead consider a finer-grained setting: given that the user is an LLM-based agent, can we identify *which model* produced the interaction trace solely from its actions?

Side-channel attacks and traffic fingerprinting. Side-channel attacks show that systems leak sensitive information through correlated observables, even when their outputs do not [22]. Applied to web traffic, deep learning classifiers trained on encrypted packet sequences identify visited websites with over 98% accuracy [23, 24]. Cook et al. [25] draw a useful distinction between on-path attackers, who observe network traffic from a separate machine, and co-located attackers, whose code runs on the same machine as the victim. Our attack is co-located: we inject JavaScript trackers into the page and collect action traces directly, without any network-level visibility.

LLM and agent fingerprinting. Pasquini et al. [9] fingerprint LLM-integrated applications by sending crafted queries and analysing responses, achieving over 95% accuracy across 42 model versions. Beyond the result, they demonstrate that knowing the underlying model enables targeted attacks: an adversary who can identify the model can craft inputs that exploit model-specific behaviours, biases, or known failure modes, turning fingerprinting from a reconnaissance step [8, 26, 27] into an attack primitive. Closest to our setting, Zhang et al. [28] show that task-specific LLM agent applications leave distinct network traffic fingerprints. Their attack observes packet-level metadata generated by agent tool use, and uses it to infer behaviours, application identity, and downstream user attributes. Concurrently, Wang et al. [10] classify seven commercial and open-source browsing agents from in-page behavioural and browser-fingerprint features.

In contrast, we study model attribution from ordinary UI events generated while an agent browses a website. Because all agents in a given experiment share the same browser harness and action space, our classifier separates models within that setting rather than distinguishing applications or interaction patterns. The resulting fingerprint is tied to the model–harness pair (Appendix B.6), but our results show that the attribution surface extends beyond network observers and active probers to the visited page itself.

3 Problem Formulation

3.1 Agent Identification as a Classification Problem

We study whether interaction traces produced by web-browsing agents contain sufficient signal to identify the underlying language model, and formalise this as a supervised classification problem over behavioural traces.

Agent and environment. An agent $a \in \mathcal{A}$ is instantiated by a language model $m \in \mathcal{M}$ interacting with a web environment $\mathcal{E}$ through a fixed browser harness h. At each timestep t, the agent conditions on the current observation s_t (a rendered screenshot), updates an internal plan, and produces an action u_t (e.g., click, scroll, keypress). The environment executes u_t, yielding a new observation s_{t+1}, and the process repeats for N planning steps.

We assume all agents share the same interface and action space. Within an experiment this removes the harness as a source of variation, but it does not make the fingerprint harness-independent: the trace reflects the model–harness pairing. We test this with a second harness in Appendix B.6. Throughout, identifying a model means identifying it as driven through a given harness.

Interaction trace. A session generates a trace

$$\tau = \{(s_t, u_t, \Delta t_t)\}_{t=1}^{T},$$

where Δt_t denotes the time elapsed between consecutive actions. We restrict our analysis to client-side interaction signals, that is, the sequence and timing of actions produced by the agent, and do not use server-side metadata such as headers, IP addresses, or TLS fingerprints. Traces may span multiple pages within the same host and are treated as belonging to a single session tied to a query Q. We further restrict traces to a single domain to control for environmental variability.

Identification task. Given a trace τ generated by an unknown agent, the goal is to predict the originating model

$$\hat{m} = f(\tau), \quad \hat{m} \in \mathcal{M},$$

where f is a classifier trained on labelled traces. Unlike prior work that frames this as human versus bot discrimination, we assume the client is automated and ask which model produced the behaviour.

Feature creation. We consider feature mappings $\phi(\tau)$ derived from Inter-Event Intervals (IEIs, the time between two consecutive actions), navigation structure (e.g., click frequency and page transitions), and interaction patterns (e.g., action type distributions). These features are designed to capture behavioural regularities induced by the underlying model. Full descriptions of extracted features are provided in Appendix A.5.

3.2 Threat Model

We consider a passive, co-located adversary: a site operator who injects lightweight JavaScript into pages to collect actions performed by an agent while visiting. Additionally, the adversary is assumed to have already established that the session originates from an automated agent rather than a human user, using classical bot-detection methods [4] or detectors designed for AI browsing agents [10]. Consequently, the identification problem becomes one of model attribution, not human-versus-bot detection; detecting such sessions is a separate problem.

The adversary has access to the sequence and timing of on-page actions, but no access to model internals, generated text, or network-layer traffic. We assume a realistic setting in which standard browser fingerprint signals are present alongside behavioural traces, and a fresh browser is instantiated for each session. The adversary is passive by assumption: they cannot modify page content or craft adversarial inputs to probe the agent directly. The adversary's objective is identification: once the underlying model is known, they can consult a corpus of model-specific jailbreaks [29, 30] or initialise a targeted optimisation procedure [9, 31, 32] with a substantially reduced search space, bypassing the cost of generic black-box probing entirely.

Depending on the adversary's knowledge of the agent population, this identification problem takes two forms.

Closed-set fingerprinting. Let $\mathcal{A} = \{a_1, \ldots, a_K\}$ be the full set of K agents. The adversary assumes that any observed trace originates from some $a_i \in \mathcal{A}$ and learns a classifier $f : \mathcal{T} \to \mathcal{A}$ over all K classes, where $\mathcal{T}$ denotes the space of interaction traces τ. At evaluation time, test traces are drawn from the same set $\mathcal{A}$, so the problem reduces to standard multi-class classification.

Notably, a closed-set classifier can be **cheaply updated as new models are released**: using the same collection pipeline, an adversary can gather traces from a new model–harness pairing and retrain the classifier in seconds (Table 13).

Open-set fingerprinting. In a more realistic setting, the adversary cannot know every agent they may encounter. Let $\mathcal{A}_{\text{train}} \subset \mathcal{A}$ be the agents known at training time, and let $\mathcal{A}_{\text{unk}} = \mathcal{A} \setminus \mathcal{A}_{\text{train}}$ denote the unknown agents. The classifier must either assign a trace to a known agent class or flag it as *unknown*.

We instantiate this setting via a **leave-one-agent-out (LOO)** protocol. Let $K = |\mathcal{A}|$. For each held-out agent $a_i \in \mathcal{A}$, we train a classifier on traces from $\mathcal{A} \setminus \{a_i\}$ (all agents except a_i) and evaluate on the test-split traces of the $K - 1$ known agents together with all traces from a_i as the unknown class. We measure the ability to separate known from unknown traces using AUROC over the binary known/unknown discrimination, reported separately for each held-out agent a_i, yielding K values across the full agent set.

4 Experimental Setup

Data We construct a dataset spanning two broad task domains where agents have been widely applied: information seeking tasks and online shopping. For question answering, we repurpose 2WikiMultiHop [33] and FRAMES [34] as live web tasks, which requires models to navigate and retrieve information across multiple pages. Similarly, for shopping, we adapt the e-commerce benchmarks Webshop [35] and Deepshop [36]. Standard train, validation, and test splits for all datasets are summarised in Table 1. Together, these environments provide a broad basis for eliciting and comparing behavioural fingerprints across diverse interaction regimes. This structure also lets us distinguish in-domain attribution, cross-task transfer within a website, pooled site-level training, and cross-site transfer; we report these generalisation experiments in Appendix B.4.

Table 1: Dataset splits used in this work. All benchmarks are deployed as live web tasks on their respective target websites.

Dataset	Domain	Target Website	Train	Val	Test	Total
2WikiMultiHop [33]	QA	Wikipedia.com	150	75	75	300
FRAMES [34]	QA	Wikipedia.com	150	75	75	300
Webshop [35]	Shopping	Amazon.com	150	75	75	300
Deepshop [36]	Shopping	Amazon.com	75	37	38	150
Total			525	262	263	1050

Models We evaluate 14 multimodal LLMs selected to support model identity classification at two levels of granularity: model family (e.g., Qwen3-VL, Qwen3.5-VL) and specific model variant (e.g., Qwen3.5-9B vs. Qwen3.5-27B). Full details are given in Appendix A.1. Locally hosted open-weight models include: the GLM-4.6V [37], Qwen3-VL series [38], Qwen3.5-VL series [39], UI-TARS-1.5-7B [40] (a UI-specialist fine-tune of Qwen2.5-VL [41]), and the Gemma-4 series [42]. We additionally include Seed-2.0-Lite [43], an open-weight model accessed via OpenRouter. Proprietary frontier models, namely GPT-5.4 [44], Gemini-3.1 and Gemini-3-Flash [45], and Claude Opus 4.6 [46], are evaluated via their respective APIs.

Agent Harness We standardise our computer-use harness with Midscene.js [47], a JavaScript library that provides a standardised interface between multimodal LLMs and browser environments, enabling models to perceive and interact with web-based UIs by translating actions into Playwright commands. All agents share an identical harness configuration, ensuring that behavioural differences within an experiment are attributable to the underlying model rather than the harness. Since Midscene.js operates in pure-vision mode only, browser observations are limited to visual screenshots; this also suits our setting, as it ensures any identifying signal derives from visual reasoning and interaction behaviour rather than from differences in how models process structured markup. In Appendix B.6, we replicate within-harness attribution with a second harness, Browser Use [48], which gives the model both screenshots and HTML text. We find that fingerprints are strongly harness-conditioned, but that training on a mix of both harnesses recovers most of the performance.

Trace collection We instrument each page with a lightweight JavaScript observer injected at session initialisation. The observer attaches event listeners to the DOM and records every interaction event produced by the agent, including click coordinates and target element type, scroll direction and magnitude, keypress events and inter-keystroke timing, and navigation events with timestamps. All events are logged with millisecond-resolution timestamps relative to session start, yielding a raw event stream that is post-processed into the structured trace format $\tau = \{(s_t, u_t, \Delta t_t)\}_{t=1}^{T}$ defined in Section 3. A fresh browser context is instantiated for each session to prevent cross-session state leakage. Each agent completes every query in the dataset independently, yielding a labelled corpus of traces with model identity as the class label.

Classifiers We train five classifier families on the collected traces: Lasso Regression, Logistic Regression, Random Forest, XGBoost, and an LSTM network. We report results primarily for XGBoost, which achieves the strongest performance across datasets; full results for all classifiers are provided in Appendix A.3.

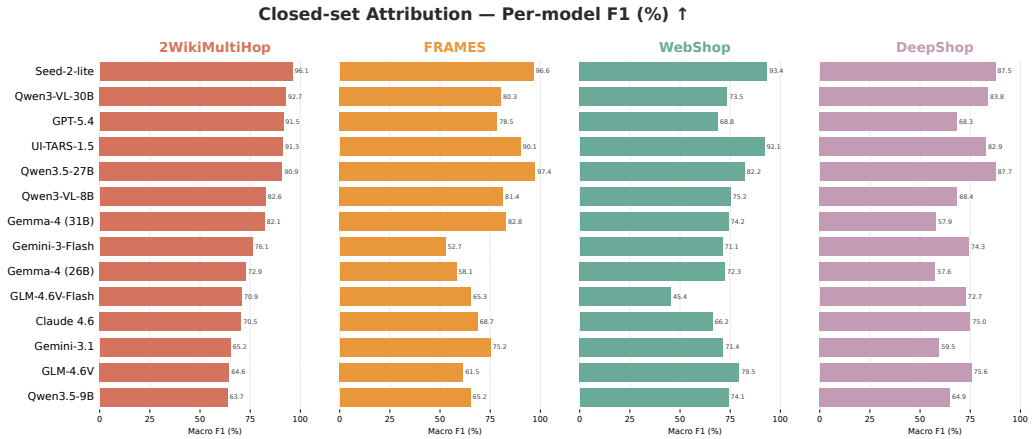


Figure 2: **Agents are identifiable from traces of their actions.** Across both information-seeking and shopping benchmarks, our XGBoost classifier trained on action traces reliably identifies the browsing agent. High scores across all four datasets indicate that agents leave consistent, distinctive traces in their browsing actions.

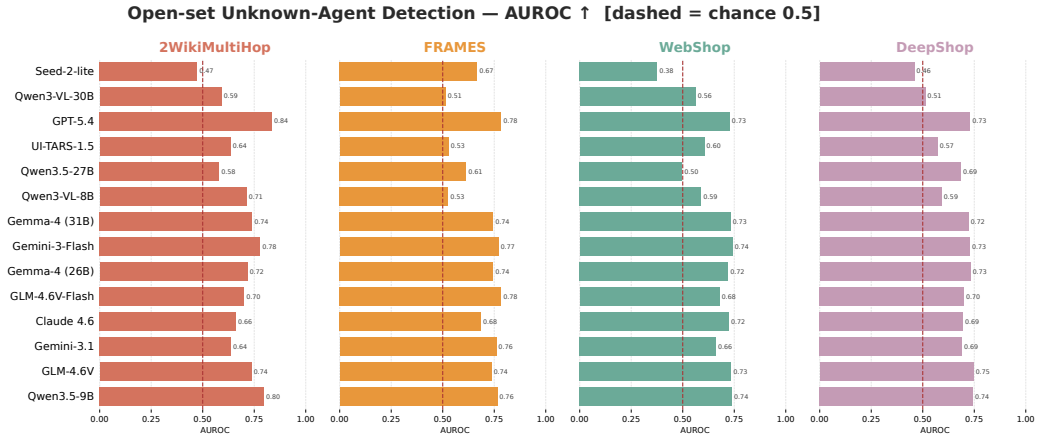


Figure 3: **Unknown agents are detectable above chance, but identification remains an open challenge.** Detection is consistently above chance across all four datasets, with the majority of agents exceeding AUROC 0.60 with our XGBoost classifier. A notable exception is Seed-2-lite, which scores below chance on 2WikiMultiHopQA, WebShop, and DeepShop despite achieving the highest closed-set identifiability. Nevertheless, robust open-set identification remains an open challenge.

Metrics To evaluate our classifiers in the closed-set fingerprinting setting, we report the per-LLM F1 score and the macro F1 across all classifiers. For the open-set fingerprinting setting, we report the AUROC for each classifier.

Hardware All open-source models are served via vLLM on a node equipped with two NVIDIA H100 GPUs. We additionally evaluate transfer between vLLM and SGLang in Appendix B.7.

5 Fingerprinting Results

5.1 Agents are identifiable from traces of their actions

Known LLMs are broadly fingerprintable from their actions. In the closed-set setting, agents are highly identifiable from action traces. Across all four benchmarks in Figure 2, our XGBoost classifier recovers the source model at roughly $10\times$ random chance, with per-agent F1 exceeding 70% for the majority of models on every dataset. Top performers such as Seed-2-lite (96.1%

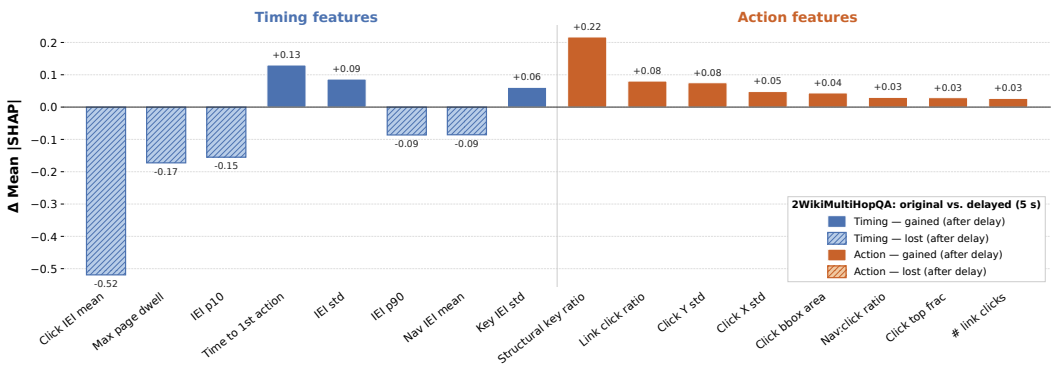


Figure 4: **Timing features are important but sensitive to perturbation, while action features are robust.** Each bar shows the change in mean |SHAP| for a feature when XGBoost is retrained on 5-second-delayed traces. Under normal conditions, timing features based on IEI statistics and time to first action dominate agent identification, but after training on delayed traces, our classifier relies on action-centred features like structural key ratio and click position to make predictions.

on 2WikiMultiHopQA) and UI-TARS-1.5 (92.1% on WebShop) are near-perfectly identifiable, suggesting their actions are highly consistent and distinct across episodes. Performance remains high even on the weakest pair (45.4% for GLM-4.6V-Flash on WebShop), well above the ∼7% random baseline for 14 classes. This extends to family-level attribution: grouping agents by model family preserves strong identifiability without version-specific labels (Table 11). We explore generalisation across tasks and sites in Appendix B.4, finding that single-task transfer is weak but pooling traces from multiple tasks on the same site recovers strong attribution.

Open-set fingerprinting is agent-specific and orthogonal to closed-set performance. Detection of unknown models is consistently above chance across all four datasets and most agents, with the majority exceeding AUROC≈ 0.60. However, agents that are easiest to classify when their identities are known (closed-set) are not easy to classify in an open-set setting. Most strikingly, Seed-2-lite (the best-identified agent in the closed-set setting) scores below chance on three of four datasets (AUROC 0.47 on 2WikiMultiHopQA, 0.38 on WebShop, 0.46 on DeepShop), while GPT-5.4 achieves the highest open-set AUROC overall (0.84 on 2WikiMultiHopQA) despite ranking third in closed-set F1.

This dissociation suggests that closed-set identifiability reflects how distinct an agent is within a known distribution, whilst open-set identifiability punishes models whose behaviour isn't uniquely distinct from that of known agents.

In general, this demonstrates that open-set detection is useful even when exact attribution is impossible: for a website host, recognising that a visiting trace belongs to no currently enrolled model is sufficient to trigger offline collection and later enrollment into the fingerprint database.

6 Analysis

6.1 Timing dominates the fingerprint, but actions are more robust to perturbation

Timing is the primary signal. We compute mean absolute SHAP values for the XGBoost classifier on 2WikiMultiHopQA before and after retraining on delayed traces in Figure 4. Without delays, our top features are overwhelmingly timing-based: IEI standard deviation, mean click IEI, and time to first action all receive substantially larger attributions than structural features like key ratio. Agents are distinguishable not primarily by what actions they take, but by their tempo: how long they pause before acting, how variable that pause is, and whether different action types carry their own characteristic delays. Full feature SHAP results are presented in Appendix B.5. Aside from these features, we show that classifier performance isn't tied to overall agent capability in Appendix B.3.

Actions carry the fingerprint when timing is disrupted. If classifiers rely on temporal signatures, adding random delays should be enough to break them. We test this by injecting a uniformly sampled

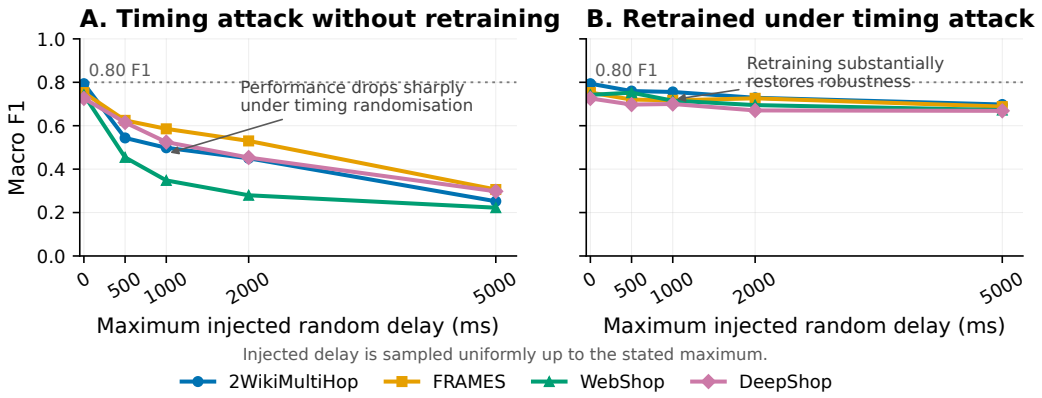


Figure 5: **Simple timing delays weaken unadapted classifiers but not adaptive ones. (Left)** Performance of an unadapted classifier, trained on clean traces and evaluated under increasing injected delays. **(Right)** Performance of a delay-adapted classifier, retrained on delayed traces and evaluated under the same perturbation. While delay injection substantially degrades the unadapted classifier, performance largely recovers after retraining.

random delay between agent actions at test time and evaluating XGBoost under increasing delay budgets in Figure 5. Without retraining, macro F1 drops sharply as injected delay grows, confirming that clean-trace classifiers are sensitive to disrupted timing rhythms. However, retraining on delayed traces largely recovers performance across all four datasets. The classifier shifts weight onto features that survive delay injection: residual timing variability, click-coordinate dispersion, structural key ratio, and link-click ratio. These are features grounded in **what** agents do and, not merely **when**.

Prompt normalisation. We also test whether prescribing a fixed browsing procedure in the task prompt suppresses model-specific behaviour (Appendix B.8). On WebShop, prompt normalisation lowers Macro-F1 of a classifier trained on the original prompts from 0.917 to 0.443. Retraining on normalised traces restores it to 0.955, and a classifier trained on both prompt regimes scores 0.889 and 0.910 on the original and normalised prompts respectively. Like delay injection, it hurts only a non-adaptive attacker, and it relies on models following the instruction.

6.2 Agent fingerprinting is efficient at both training and test time

Strong classifiers can be trained from a fraction of the traces. By varying the proportion of training data used to fit our XGBoost classifier, we find that fewer than one third of traces are sufficient to approach peak classification performance across all four datasets (Figure 6**A**). Gains diminish rapidly beyond this point, indicating that the behavioural signatures underpinning agent identity are both consistent and learnable from modest supervision.

Agent Identification improves steadily with observed events. Using our XGBoost classifier trained on full traces, we systematically reduce the number of events observed at test time to assess how quickly identity can be recovered mid-trajectory. As shown in Figure 6**B**, Macro-F1 improves steadily with the number of observed events rather than saturating early. Averaged over the four datasets, it is 0.27 after 10 events and 0.56 after 50, and reaches 0.72 after 100 events, close to the 0.75 achieved on complete traces (Appendix B.9). Identification is therefore possible mid-session, with accuracy improving as the agent continues to act.

7 Implications

7.1 Attack Surface: Agent-Targeted Exploits

Knowing the identity of a target makes an adversary's job easier, reducing the search space from all possible attacks to those known to be effective against the target [8]. A natural application is the class of threats known as AI Agent Traps: adversarial page content designed to manipulate, deceive, or

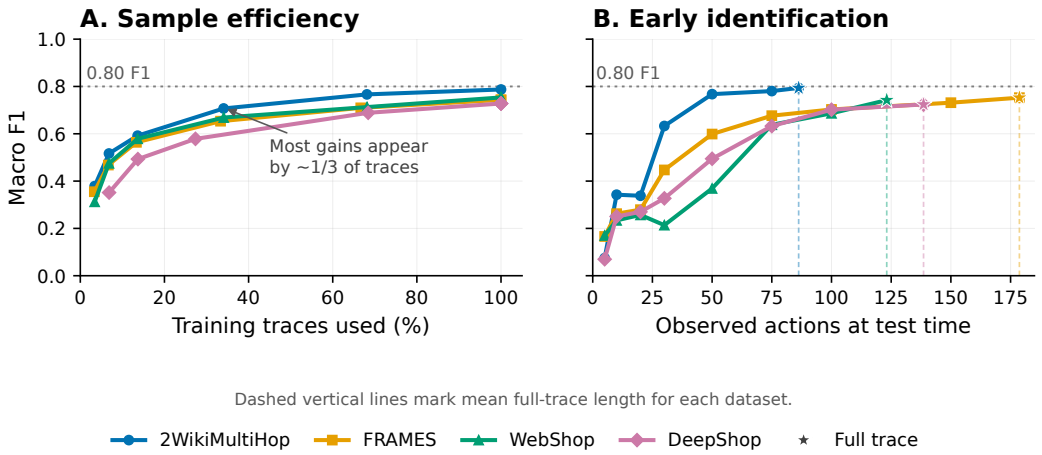


Figure 6: **Agent fingerprints emerge early and can be learned from short traces. (Left)** Training efficiency, measured by training classifiers using only the first k events from each training trace and evaluating them on full held-out traces. **(Right)** Identification speed, measured by training classifiers on full traces and evaluating them using only the first k events observed at test time. Dashed vertical lines mark the mean full-trace length for each dataset.

exploit visiting agents [49]. Passive fingerprinting lets each trap be conditioned on model identity, converting generic exploits into targeted ones across three attack surfaces we characterise below

Model-specific prompt injection. LLMs exhibit systematically different susceptibilities to prompt injection [31, 9]. Without identity information, an adversary must either deploy generic injections that may be ineffective, or mount a costly black-box search over model-specific failure modes, both of which are detectable and expensive. Action fingerprinting collapses this search *passively*: once the model is identified, the adversary can select directly from a known corpus of model-specific jailbreaks [30, 29] or initialise a targeted optimisation with a substantially reduced search space [31, 9, 32]. Because fingerprinting is passive, a visiting agent has no direct signal that it is being identified. Furthermore, since the injection payload can be conditioned on the inferred identity after fingerprinting completes, it is invisible to any visitor whose model does not match the target, making such attacks difficult to detect through conventional crawling or auditing.

Adversarial cost inflation. Sponge attacks aim to exhaust the computational budget of neural networks by inducing verbose or high-entropy reasoning [50], and OverThink [51] shows that decoy problems planted in web content can inflate a reasoning model's token use, charging the cost to whoever runs the model. Agent identifiability introduces a targeted variant: a site operator who knows which model is visiting can serve that model pages calibrated to maximise its token consumption, through navigational ambiguity, redundant content, or structures known to trigger extended reasoning, while serving normal content to all other visitors. This constitutes a denial-of-service vector targeting the *user's* inference budget rather than the host's compute, a threat class that to our knowledge has not been previously formalised. It is particularly consequential for high-cost frontier models such as Claude Opus 4.6 or GPT-5.4, where modest per-query cost increases compound significantly at scale.

Agent-specific access control. Beyond active exploitation, identity signals let site operators route content by model identity. We anticipate two near-term variants. *Blacklisting* denies service to specific models for cost, legal, or competitive reasons without blocking agents broadly. *Whitelisting*, the more adversarial inverse, serves model-conditioned content that misleads specific agents while appearing benign to others. This is especially dangerous because poisoned content may be invisible to auditors using a different model. Neither variant requires a fixed classifier: a site operator can *continuously retrain* on organic or simulated traces, extending coverage to new models without external data. Appendix B.4 supports this assumption: although single-task transfer is weak, pooling multiple tasks from the same website recovers strong attribution.

8 Limitations

Harness and setting. The fingerprint is conditioned on the harness: classifiers trained on one harness do not transfer to the other, and pooled training is required (Appendix B.6). We tested two harnesses, MidScene.js and Browser Use, on two websites and four tasks. Our threat model assumes that agent detection has already taken place, so we did not mix agent traces with human or other bot traffic.

Inference infrastructure. In Appendix B.7 we varied the inference engine for three open-weight models and found that classifier performance remains high. We did not vary GPU architecture or the location of the serving nodes, either of which may change inter-event timings and hence the timing features on which our classifiers rely.

9 Conclusion

We show that UI actions from agent traces can fingerprint the underlying model of an LLM-based web agent with up to 96% F1 across 14 frontier models. The signal is typically recoverable from few observed events, does not rely on network-layer visibility, and is passive. The practical consequence is a shift in the threat landscape for deployed agents. Target identification is the first step towards exploitation, and we show that this move can be executed passively during ordinary navigation by any party that controls a page visited by the agent. Every page visit thus becomes a potential attribution event on which model-specific injections, content poisoning, or budget exhaustion can be conditioned.

The operative question for the next generation of agent-aware web infrastructure is therefore no longer whether a client is human or automated, but which model is producing the behaviour. Within-agent attribution, rather than bot detection, is the axis on which both defences and cooperative protocols should be designed. We release our trace corpus and evaluation harness to make that axis measurable.

Future work should evaluate obfuscation against adaptive adversaries. In Appendix B.8, we show that system prompt normalisation sharply reduces classifier performance, but fails against classifiers trained on traces from agents with multiple prompt formats.

Overall, as LLM-based agents become a standard client of the web, their on-page behaviour is an identifying signal that warrants the same engineering scrutiny as the models themselves. Attackers and operators now have access to the same primitive, and the design of agent-aware infrastructure will be shaped by how each chooses to use it. Whether it is used to exploit agents or to serve them better, the primitive is now available to both.

Acknowledgements

LW and JM were supported by a Rhodes Scholarship. SM was supported by the AI Safety Fund and Microsoft Ltd. CR carried out this work under the ERC Horizon project DVPS.

References

[1] Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. WebGPT: Browser-assisted question-answering with human feedback, June 2022. URL http://arxiv.org/abs/2112.09332. arXiv:2112.09332.

[2] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. ReAct: Synergizing Reasoning and Acting in Language Models. September 2022. URL https://openreview.net/forum?id=WE_vluYUL-X.

[3] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Jirong Wen. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6): 186345, March 2024. ISSN 2095-2236. doi: 10.1007/s11704-024-40231-1. URL https://doi.org/10.1007/s11704-024-40231-1.

[4] Christos Iliou, Theodoros Kostoulas, Theodora Tsikrika, Vasilios Katos, Stefanos Vrochidis, and Ioannis Kompatsiaris. Detection of Advanced Web Bots by Combining Web Logs with Mouse Behavioural Biometrics. In *Digital Threats: Research and Practice*, volume 2, 2021. URL https://dl.acm.org/doi/10.1145/3447815.

[5] Alejandro Acien, Aythami Morales, Julian Fierrez, and Ruben Vera-Rodriguez. BeCAPTCHA-Mouse: Synthetic Mouse Trajectories and Improved Bot Detection. *Pattern Recognition*, 127: 108643, 2022. URL https://arxiv.org/abs/2005.00890.

[6] Peter Eckersley. How Unique Is Your Web Browser? In *Privacy Enhancing Technologies Symposium (PETS)*, pages 1–18. Springer, 2010. URL https://panopticlick.eff.org/static/browser-uniqueness.pdf.

[7] Antoine Vastel, Pierre Laperdrix, Walter Rudametkin, and Romain Rouvoy. FP-Stalker: Tracking Browser Fingerprint Evolutions. In *IEEE Symposium on Security and Privacy*, pages 728–741, 2018. URL https://arxiv.org/abs/1805.09046.

[8] Eric M. Hutchins, Michael J. Cloppert, and Rohan M. Amin. Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains. Technical report, Lockheed Martin Corporation, 2011. URL https://lockheedmartin.com/content/dam/lockheed-martin/rms/documents/cyber/LM-White-Paper-Intel-Driven-Defense.pdf.

[9] Dario Pasquini, Evgenios M. Kornaropoulos, and Giuseppe Ateniese. LLMmap: Fingerprinting for Large Language Models. pages 299–318, 2025. ISBN 978-1-939133-52-6. URL https://www.usenix.org/conference/usenixsecurity25/presentation/pasquini.

[10] Ethan Wang, Zubair Shafiq, and Yash Vekaria. FP-Agent: Fingerprinting AI Browsing Agents, May 2026. URL http://arxiv.org/abs/2605.01247. arXiv:2605.01247 [cs.CR].

[11] Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Yonatan Bisk, Daniel Fried, Uri Alon, and others. WebArena: A Realistic Web Environment for Building Autonomous Agents. *arXiv preprint arXiv:2307.13854*, 2023.

[12] Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2Web: Towards a Generalist Agent for the Web, 2023. _eprint: 2306.06070.

[13] T. Kabe and M. Miyazaki. Determining WWW user agents from server access log. In *Proceedings Seventh International Conference on Parallel and Distributed Systems: Workshops*, pages 173–178, 2000. doi: 10.1109/PADSW.2000.884534.

[14] Paul Huntington, David Nicholas, and Hamid R. Jamali. Web robot detection in the scholarly information environment. *J. Inf. Sci.*, 34(5):726–741, October 2008. ISSN 0165-5515. doi: 10.1177/0165551507087237. URL https://doi.org/10.1177/0165551507087237.

[15] Nick Geens, Johan Huysmans, and Jan Vanthienen. Evaluation of Web Robot Discovery Techniques: A Benchmarking Study. In Petra Perner, editor, *Advances in Data Mining. Applications in Medicine, Web Mining, Marketing, Image and Signal Mining*, pages 121–130, Berlin, Heidelberg, 2006. Springer. ISBN 978-3-540-36037-7. doi: 10.1007/11790853_10.

[16] Martijn Koster. The Web Robots Pages, 1994. URL https://www.robotstxt.org/orig.html.

[17] Martijn Koster, Gary Illyes, Henner Zeller, and Lizzi Sassman. Robots Exclusion Protocol. Request for Comments RFC 9309, Internet Engineering Task Force, September 2022. URL https://datatracker.ietf.org/doc/rfc9309. Num Pages: 12.

[18] Derek Doran and Swapna S. Gokhale. Web robot detection techniques: overview and limitations. *Data Mining and Knowledge Discovery*, 22(1):183–210, January 2011. ISSN 1573-756X. doi: 10.1007/s10618-010-0180-z. URL https://doi.org/10.1007/s10618-010-0180-z.

[19] Shen Fu, Dong Qin, Daji Qiao, and George T Amariucai. RUMBA-Mouse: Rapid User Mouse-Behavior Authentication Using a CNN-RNN Approach. In *2020 IEEE Conference on Communications and Network Security (CNS)*, pages 1–9. IEEE, 2020. ISBN 978-1-7281-4760-4.

[20] Yi Wang, Chengyv Wu, Yang Liao, and Maowei You. Optimizing Mouse Dynamics for User Authentication by Machine Learning: Addressing Data Sufficiency, Accuracy-Practicality Trade-off, and Model Performance Challenges, May 2025. URL http://arxiv.org/abs/2504.21415. arXiv:2504.21415 [cs].

[21] Penny Chong, Yuval Elovici, and Alexander Binder. User Authentication Based on Mouse Dynamics Using Deep Neural Networks: A Comprehensive Study. *IEEE Transactions on Information Forensics and Security*, 15:1086–1101, January 2020. doi: 10.1109/TIFS.2019.2930429. URL https://dl.acm.org/doi/abs/10.1109/TIFS.2019.2930429.

[22] Paul C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In *Advances in Cryptology (CRYPTO)*, pages 104–113. Springer, 1996. URL https://www.paulkocher.com/doc/TimingAttacks.pdf.

[23] Vera Rimmer, Davy Preuveneers, Marc Juarez, Tom Van Goethem, and Wouter Joosen. Automated Website Fingerprinting through Deep Learning. In *Network and Distributed System Security Symposium (NDSS)*, 2018. URL https://arxiv.org/abs/1708.06376.

[24] Payap Sirinam, Mohsen Imani, Marc Juarez, and Matthew Wright. Deep Fingerprinting: Undermining Website Fingerprinting Defenses with Deep Learning. In *ACM Conference on Computer and Communications Security (CCS)*, pages 1928–1943, 2018. URL https://arxiv.org/abs/1801.02265.

[25] Jack Cook, Jules Drean, Jonathan Behrens, and Mengjia Yan. There's always a bigger fish: a clarifying analysis of a machine-learning-assisted side-channel attack. In *Proceedings of the 49th Annual International Symposium on Computer Architecture*, ISCA '22, pages 204–217, New York, NY, USA, June 2022. Association for Computing Machinery. ISBN 978-1-4503-8610-4. doi: 10.1145/3470496.3527416. URL https://dl.acm.org/doi/10.1145/3470496.3527416.

[26] MITRE Corporation. MITRE ATT&CK: Adversarial Tactics, Techniques and Common Knowledge, 2025. URL https://attack.mitre.org.

[27] Shaked Zychlinski. A Whole New World: Creating a Parallel-Poisoned Web Only AI-Agents Can See, August 2025. URL http://arxiv.org/abs/2509.00124. arXiv:2509.00124 [cs] version: 1.

[28] Yixiang Zhang, Xinhao Deng, Zhongyi Gu, Yihao Chen, Ke Xu, Qi Li, and Jianping Wu. Exposing LLM User Privacy via Traffic Fingerprint Analysis: A Study of Privacy Risks in LLM Agent Interactions, October 2025. URL http://arxiv.org/abs/2510.07176. arXiv:2510.07176 [cs] version: 1.

[29] Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J. Zico Kolter, and Matt Fredrikson. Universal and Transferable Adversarial Attacks on Aligned Language Models, December 2023. URL http://arxiv.org/abs/2307.15043. arXiv:2307.15043 [cs].

[30] Xinyue Shen, Zeyuan Chen, Michael Backes, Yun Shen, and Yang Zhang. "Do Anything Now": Characterizing and Evaluating In-The-Wild Jailbreak Prompts on Large Language Models. *arXiv preprint arXiv:2308.03825*, 2024.

[31] Lukas Aichberger, Alasdair Paren, Yarin Gal, Philip Torr, and Adel Bibi. Attacking Multimodal OS Agents with Malicious Image Patches, March 2025. URL http://arxiv.org/abs/2503.10809. arXiv:2503.10809 [cs].

[32] Dominik Seip and Matthias Hein. Preference Redirection via Attention Concentration: An Attack on Computer Use Agents, April 2026. URL http://arxiv.org/abs/2604.08005. arXiv:2604.08005 [cs].

[33] Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing A Multi-hop QA Dataset for Comprehensive Evaluation of Reasoning Steps. In Donia Scott, Nuria Bel, and Chengqing Zong, editors, *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics. doi: 10.18653/v1/2020.coling-main.580. URL https://aclanthology.org/2020.coling-main.580/.

[34] Satyapriya Krishna, Kalpesh Krishna, Anhad Mohananey, Steven Schwarcz, Adam Stambler, Shyam Upadhyay, and Manaal Faruqui. Fact, Fetch, and Reason: A Unified Evaluation of Retrieval-Augmented Generation, January 2025. URL http://arxiv.org/abs/2409.12941. arXiv:2409.12941 [cs].

[35] Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. WebShop: Towards Scalable Real-World Web Interaction with Grounded Language Agents, February 2023. URL http://arxiv.org/abs/2207.01206. arXiv:2207.01206 [cs].

[36] Yougang Lyu, Xiaoyu Zhang, Lingyong Yan, Maarten de Rijke, Zhaochun Ren, and Xiuying Chen. DeepShop: A Benchmark for Deep Research Shopping Agents, June 2025. URL http://arxiv.org/abs/2506.02839. arXiv:2506.02839 [cs].

[37] V. Team, Wenyi Hong, Wenmeng Yu, Xiaotao Gu, Guo Wang, Guobing Gan, Haomiao Tang, Jiale Cheng, Ji Qi, Junhui Ji, Lihang Pan, Shuaiqi Duan, Weihan Wang, Yan Wang, Yean Cheng, Zehai He, Zhe Su, Zhen Yang, Ziyang Pan, Aohan Zeng, Baoxu Wang, Bin Chen, Boyan Shi, Changyu Pang, Chenhui Zhang, Da Yin, Fan Yang, Guoqing Chen, Jiazheng Xu, Jiale Zhu, Jiali Chen, Jing Chen, Jinhao Chen, Jinghao Lin, Jinjiang Wang, Junjie Chen, Leqi Lei, Letian Gong, Leyi Pan, Mingdao Liu, Mingde Xu, Mingzhi Zhang, Qinkai Zheng, Sheng Yang, Shi Zhong, Shiyu Huang, Shuyuan Zhao, Siyan Xue, Shangqin Tu, Shengbiao Meng, Tianshu Zhang, Tianwei Luo, Tianxiang Hao, Tianyu Tong, Wenkai Li, Wei Jia, Xiao Liu, Xiaohan Zhang, Xin Lyu, Xinyue Fan, Xuancheng Huang, Yanling Wang, Yadong Xue, Yanfeng Wang, Yanzi Wang, Yifan An, Yifan Du, Yiming Shi, Yiheng Huang, Yilin Niu, Yuan Wang, Yuanchang Yue, Yuchen Li, Yutao Zhang, Yuting Wang, Yu Wang, Yuxuan Zhang, Zhao Xue, Zhenyu Hou, Zhengxiao Du, Zihan Wang, Peng Zhang, Debing Liu, Bin Xu, Juanzi Li, Minlie Huang, Yuxiao Dong, and Jie Tang. GLM-4.5V and GLM-4.1V-Thinking: Towards Versatile Multimodal Reasoning with Scalable Reinforcement Learning, 2025. URL https://arxiv.org/abs/2507.01006. _eprint: 2507.01006.

[38] Qwen Team. Qwen3 Technical Report, 2025. URL https://arxiv.org/abs/2505.09388. _eprint: 2505.09388.

[39] Qwen Team. Qwen3.5: Towards Native Multimodal Agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

[40] Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, Wanjun Zhong, Kuanye Li, Jiale Yang, Yu Miao, Woyu Lin, Longxiang Liu, Xu Jiang, Qianli Ma, Jingyu Li, Xiaojun Xiao, Kai Cai, Chuang Li, Yaowei Zheng, Chaolin Jin, Chen Li, Xiao Zhou, Minchao Wang, Haoli Chen, Zhaojian Li, Haihua Yang, Haifeng Liu, Feng Lin, Tao Peng, Xin Liu, and Guang Shi. UI-TARS: Pioneering Automated GUI Interaction with Native Agents, January 2025. URL http://arxiv.org/abs/2501.12326. arXiv:2501.12326 [cs].

[41] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-VL Technical Report. *arXiv preprint arXiv:2502.13923*, 2025.

[42] Gemma Team. Gemma 4, 2026. URL https://deepmind.google/models/gemma/gemma-4/.

[43] ByteDance Seed Team. Seed2.0 Model Card, 2026. URL https://github.com/ByteDance-Seed/Seed2.0.

[44] OpenAI. Introducing GPT-5.4, April 2026. URL https://openai.com/index/introducing-gpt-5-4/.

[45] Google. Gemini 3.1 Pro. URL https://deepmind.google/models/gemini/pro/.

[46] Anthropic. Claude Opus 4.6. URL https://www.anthropic.com/news/claude-opus-4-6.

[47] YiBing Lin Xiao Zhou, Tao Yu. Midscene.js: Your AI Operator for Web, Android, iOS, Automation & Testing., 2025. URL https://github.com/web-infra-dev/midscene.

[48] Magnus Müller and Gregor Žunič. Browser Use: Enable AI to control your browser, 2024. URL https://github.com/browser-use/browser-use.

[49] Matija Franklin, Nenad Tomašev, Julian Jacobs, Joel Z. Leibo, and Simon Osindero. AI Agent Traps, March 2026. URL https://papers.ssrn.com/abstract=6372438.

[50] Ilia Shumailov, Yiren Zhao, Daniel Bates, Nicolas Papernot, Robert Mullins, and Ross Anderson. Sponge Examples: Energy-Latency Attacks on Neural Networks. In *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 212–231, September 2021. doi: 10.1109/EuroSP51992.2021.00024. URL https://ieeexplore.ieee.org/document/9581273.

[51] Abhinav Kumar, Jaechul Roh, Ali Naseh, Marzena Karpinska, Mohit Iyyer, Amir Houmansadr, and Eugene Bagdasarian. Overthink: Slowdown attacks on reasoning LLMs. *arXiv preprint arXiv:2502.02542*, 2025.

Appendix

A Additional Experimental Details

A.1 LLMs Used

Table 2: Models evaluated in this work. ‡UI-specialist model fine-tuned for GUI agent tasks. Active parameter counts shown for mixture-of-experts (MoE) models.

Model	Type	Params	Active	Access
Open-source — locally hosted				
Qwen3-VL-8B [38]	General	8B	8B	Local
Qwen3-VL-30B-A3B [38]	General	30B	3B	Local
Qwen3.5-9B [39]	General	9B	9B	Local
Qwen3.5-27B [39]	General	27B	27B	Local
GLM-4.6V [37]	General	106B	—	Local
GLM-4.6V-Flash [37]	General	9B	—	Local
UI-TARS-1.5-7B‡ [40]	UI-specialist	7B	7B	Local
Gemma-4-26B-A4B [42]	General	26B	4B	Local
Gemma-4-31B [42]	General	31B	31B	Local
Proprietary — API				
GPT-5.4 [44]	General	—	—	API
Gemini-3.1 [45]	General	—	—	API
Gemini-3-Flash [45]	General	—	—	API
Claude Opus 4.6 [46]	General	—	—	API
Seed-2.0-Lite [43]	General	—	—	API

A.2 Prompt Templates for Datasets

Task: Wikipedia Question Answering
You are browsing Wikipedia to answer a question.

Rules:

- Restrict browsing to `https://www.wikipedia.org`.
- Use the browser to gather evidence before answering.
- Output the final answer as `<answer>...</answer>`.

Question:
`{question}`

Task: Amazon Product Selection
You are a shopping assistant tasked with finding the product that best matches the description.

Instructions:

- Browse Amazon freely (search, filter, and view product pages).
- Add promising items to your cart and refine your selection.
- Remove items if better alternatives are found.
- Do not log in or proceed to checkout.

At completion, your cart should contain only the product(s) that best match the description.

Description:
`{description}`

A.3 Classifier Hyperparameters

We train five classifier families on the behavioural feature set described in Section A.5: a Random Forest, XGBoost, two Logistic Regression variants (L2 and L1), and a hybrid LSTM. All classifiers are selected by validation accuracy via cross-validated hyperparameter search; final models are refit on the training split only.

Random Forest. Exhaustive grid search (3-fold CV) over the space in Table 3. Features are used without scaling.

Hyperparameter	Search space
`n_estimators`	200, 400
`max_depth`	None (unlimited), 15, 30
`max_features`	`sqrt`, `log2`, 0.4
`min_samples_split`	2, 5

Table 3: Random Forest hyperparameter grid (exhaustive, $2\times3\times3\times2 = 36$ configurations).

XGBoost. Randomised search (40 iterations, 3-fold CV, `hist` tree method) over the space in Table 4. Features are used without scaling.

Hyperparameter	Search space
`n_estimators`	100, 200, 300, 400, 500
`learning_rate`	0.01, 0.05, 0.1, 0.2, 0.3
`max_depth`	3, 4, 5, 6, 7, 8
`subsample`	0.6, 0.7, 0.8, 0.9, 1.0
`colsample_bytree`	0.5, 0.6, 0.7, 0.8, 1.0
`reg_alpha`	0, 0.01, 0.1, 1.0
`reg_lambda`	0.5, 1.0, 2.0, 5.0

Table 4: XGBoost hyperparameter search space (randomised, 40 draws).

Logistic Regression. We train two variants with 3-fold grid search over $C \in \{0.01, 0.1, 1.0, 10.0\}$, with a maximum of 5,000 iterations. Features are z-score normalised (StandardScaler fit on training data only). LR-L2 uses the `lbfgs` solver; LR-Lasso uses the `saga` solver with full L1 regularisation.

LSTM. The AgentLSTM encodes the raw browser event sequence. Each event is represented as a token embedding (dimension 16) concatenated with five continuous scalars: log inter-event gap, log absolute session timestamp, spatial position (normalised click x/y or scroll depth), and log running per-type inter-event interval. The token vocabulary covers eight event types (click, keydown, scroll, navigate, beforeunload, focus, plus `<pad>` and `<unk>`). The LSTM final hidden state is concatenated with the pre-computed aggregate feature vector before the classification head.

Fixed training hyperparameters are listed in Table 5; the grid searched over hidden dimension and dropout is given in Table 6. Best configuration is selected by validation accuracy; the final model is refit from scratch on the training split for 50 epochs.

A.4 Harness Configuration

Overview. Each episode is executed by a Python orchestrator (`orchestrator.py`) that dispatches agent runs as isolated Apptainer container invocations. The container (`agent.sif`) packages a headless Chromium browser, the Playwright automation framework, and the MidScene `PlaywrightAgent` action loop. This design ensures that no browser state, cookies, or cached content persists across episodes or across agents.

Episode execution. For each (agent, question) pair, the orchestrator invokes the container with the agent's API credentials, the task prompt, and a unique episode ID. Inside the container,

Hyperparameter	Value
Embedding dimension	16
Number of LSTM layers	2
Optimizer	Adam
Learning rate	1×10^{-3}
Weight decay	1×10^{-4}
Batch size	16
Epochs	50
Loss	Cross-entropy

Table 5: Fixed LSTM training hyperparameters.

Hyperparameter	Search space
Hidden dimension	64, 128, 256, 512
Dropout	0.2, 0.4

Table 6: LSTM grid search space ($4 \times 2 = 8$ configurations, selected by validation accuracy).

`agent_runner.ts` navigates Chromium to the task start URL and runs `agent.aiAct(prompt)` until the task is complete or the per-episode timeout is reached. The browser viewport is fixed at 1280×768 pixels across all agents and tasks. The MidScene replanning cycle limit is set to 40 across all agents, capping the maximum number of action–observation loops per episode.

Event collection. Browser interaction events are captured via two complementary paths to ensure completeness:

- **Primary (push bridge):** A `page_tracer.js` init script is injected into every page context. It patches the DOM event listeners and calls `window.__pushTraceEvent` for each recorded interaction (click, keydown, scroll, navigate, beforeunload, focus). Playwright routes these calls over the Chrome DevTools Protocol (CDP) to a host-side handler, which timestamps each event relative to the episode wall-clock start and appends it to the episode buffer. No polling is required.
- **Secondary (backstop harvest):** A single `harvest()` call at episode end reads any events remaining in the in-page `__agentTrace.events` array that did not complete their CDP bridge call before page teardown (most commonly `beforeunload` events mid-navigation).

HTTP-level page navigations — invisible to the in-page tracer — are captured separately via Playwright's `framenavigated` listener and appended as `navigate` events with a `trigger: "http"` field.

Output format. Each completed episode is written as a JSON file at:

```
traces/{agent_id}/{dataset_name}/{timestamp}/{episode_id}.json
```

The file contains episode metadata (agent ID, model name, task type, timestamp), the agent's answer and optional verification result, the MidScene action log, and the full DOM event trace including all timestamped browser interactions.

Run configuration. Experiments are specified via YAML configuration files that define the agent set, dataset slices, and run parameters. Table 7 summarises the fixed run parameters used across all experiments reported in this paper. Collecting traces for all datasets takes approximately three days. The cost of generating text for API accessed models totals USD 2890. The API's used are provided by Anthropic, OpenAI and OpenRouter.

Parameter	Value
Browser	Chromium (headless)
Viewport	1280×768 px
Agent framework	MidScene `PlaywrightAgent`
Replanning cycle limit	40
Episodes per (agent, question)	1
Per-episode timeout	300 s
Parallel workers	5
Container runtime	Apptainer

Table 7: Fixed harness parameters used in all experiments.

A.5 Behavioural Features Collected

For each browsing episode, we extract a fixed-dimensional feature vector from the client-side event trace. Let an episode consist of an ordered sequence of DOM events

$$E = \{e_1, e_2, \ldots, e_T\},$$

where each event has a type (e.g., click, scroll, navigate), a timestamp, and optionally additional fields such as URL, scroll percentage, or screen coordinates. We also use the corresponding Midscene action log, which records the high-level actions issued by the agent through the browser harness.

Event subsets. From the raw event sequence, we define the following subsets:

$$E_{\text{click}},\ E_{\text{scroll}},\ E_{\text{nav}},\ E_{\text{keydown}},\ E_{\text{beforeunload}},\ E_{\text{focus}},$$

corresponding respectively to click, scroll, navigation, keydown, `beforeunload`, and focus events. Let M denote the Midscene action log for the same episode.

We define the page count P as the number of distinct pages visited in the episode, using the recorded page count when available and otherwise falling back to the number of unique URLs observed in the DOM trace.

Temporal dynamics. To characterise interaction rhythm, we extract the sequence of event timestamps

$$t_1, t_2, \ldots, t_T,$$

and compute inter-event intervals

$$\Delta_i = t_{i+1} - t_i \quad \text{for } i = 1, \ldots, T-1.$$

Behavioral Feature Set We extract 41 scalar features from each episode's browser event trace (DOM event log). Features are grouped into seven categories below. All features are computed purely from client-side browser events (click, keydown, scroll, navigate, beforeunload, focus) and require no access to model internals or outputs.

Feature	Description
Event volume	
`n_clicks`	Total number of click events
`n_scrolls`	Total number of scroll events
`n_navigations`	Total number of page navigation events
`n_keydowns`	Total number of keydown events
`n_focus`	Total number of input/textarea focus events
`n_events_total`	Total events across all types
`page_count`	Number of distinct pages visited
`n_unique_domains`	Number of unique hostnames visited
Global timing	

total_duration_s	Wall-clock episode duration (seconds)
t_first_action_ms	Time from episode start to first event (ms)
mean_iei_ms	Mean inter-event interval across all events (ms)
std_iei_ms	Standard deviation of inter-event intervals (ms)
median_iei_ms	Median inter-event interval (ms)
p10_iei_ms	10th percentile inter-event interval (ms)
p90_iei_ms	90th percentile inter-event interval (ms)
iei_trend	Ratio of mean IEI in the second half of the episode to the first half; values > 1 indicate the agent slows down as context grows
Per-type planning latency	
mean_click_iei_ms	Mean inter-click interval (ms)
std_click_iei_ms	Std. of inter-click intervals (ms)
mean_nav_iei_ms	Mean inter-navigation interval, approximating page dwell time (ms)
std_nav_iei_ms	Std. of inter-navigation intervals (ms)
max_page_dwell_ms	Maximum single-page dwell time (ms)
mean_key_iei_ms	Mean inter-keydown interval, approximating API keystroke latency (ms)
std_key_iei_ms	Std. of inter-keydown intervals (ms)
Scroll behavior	
max_scroll_pct	Maximum scroll depth reached, as a percentage of page height
mean_scroll_pct	Mean scroll depth across all scroll events
n_deep_scrolls	Number of scroll events reaching >60% page depth
scroll_reversals	Number of direction reversals in the scroll depth sequence
Click spatial distribution	
click_x_std	Standard deviation of click x-coordinates (pixels)
click_y_std	Standard deviation of click y-coordinates (pixels)
click_bbox_area_frac	Bounding-box area of all click positions as a fraction of the 1280×768 viewport
click_top_frac	Fraction of clicks in the top quarter of the viewport ($y < 192$px), capturing navbar/search-bar interaction
n_link_clicks	Number of clicks on anchor elements (href present)
link_click_ratio	n_link_clicks / n_clicks
Navigation strategy	
popstate_ratio	Fraction of navigations triggered by popstate (history back)
scroll_to_click_ratio	n_scrolls / n_clicks
actions_per_page	n_events_total / page_count
nav_to_click_ratio	n_navigations / n_clicks
keydowns_per_page	n_keydowns / page_count
focus_per_page	n_focus / page_count
structural_key_ratio	Fraction of keydowns that are structural keys (Enter, Arrow*, Tab, Escape, Backspace, Delete) vs. printable characters
Exit behavior	
mean_exit_scroll_pct	Mean scroll depth at beforeunload events, reflecting how far the agent had read before leaving each page

Table 8: The 41 behavioral features extracted from each episode trace

B Supplementary Results

Table 9: Agent identification F1 (%) across datasets and classifiers. Best F1 per model group in **bold**. Macro f1 for each dataset and classifier is at the bottom of the table.

Model	Clf.	2Wiki *(in-dom.)*	FRAMES *(in-dom.)*	Webshop *(in-dom.)*	DeepShop *(in-dom.)*
Proprietary Models					
GPT-5.4	RF	85.71	72.19	68.09	66.67
	XGB	**91.50**	**78.48**	**68.78**	**68.29**
	LSTM	76.19	73.85	65.90	46.15
	Lasso	78.53	66.23	61.29	58.67
	LR	77.38	64.47	61.96	61.11
Claude Opus 4.6	RF	67.90	58.54	**67.13**	56.34
	XGB	**70.51**	**68.71**	66.18	**75.00**
	LSTM	46.51	58.03	57.14	48.72
	Lasso	52.11	59.67	55.81	57.83
	LR	47.89	60.34	56.49	60.24
Gemini-3.1-Pro	RF	64.75	64.86	**75.17**	52.05
	XGB	**65.19**	**75.16**	71.43	59.46
	LSTM	61.73	59.88	72.48	**61.11**
	Lasso	54.29	57.55	70.67	42.25
	LR	51.47	58.16	68.00	43.24
Gemini-3-Flash	RF	66.67	**54.01**	66.23	61.33
	XGB	**76.12**	52.70	**71.14**	**74.29**
	LSTM	67.53	49.18	56.16	55.88
	Lasso	67.10	37.18	59.15	46.58
	LR	67.53	40.26	59.57	43.24
Seed-2.0-Lite	RF	93.51	95.36	93.33	**90.24**
	XGB	**96.05**	96.60	93.42	87.50
	LSTM	87.90	**96.69**	**96.05**	90.14
	Lasso	89.47	87.84	95.17	86.42
	LR	92.62	87.84	95.17	85.00
Open-Source Models					
Gemma-4-31B-it	RF	80.26	73.68	73.55	53.16
	XGB	**82.12**	82.80	**74.17**	57.89
	LSTM	73.37	**84.77**	67.69	**63.41**
	Lasso	81.38	79.74	60.92	55.56
	LR	81.88	77.63	61.99	56.34
Gemma-4-26B	RF	69.92	48.65	61.64	30.19
	XGB	**72.87**	58.11	**72.26**	**57.58**
	LSTM	52.17	**64.38**	58.76	38.60
	Lasso	56.91	52.70	58.39	53.12
	LR	56.67	51.35	57.35	53.12
GLM-4.6V	RF	60.43	54.01	75.52	67.47
	XGB	**64.56**	**61.54**	79.45	**75.61**
	LSTM	38.46	48.18	**80.75**	67.57
	Lasso	53.59	56.00	77.50	62.50
	LR	51.28	59.38	75.61	65.82
GLM-4.6V-Flash	RF	63.44	54.41	**54.10**	68.29
	XGB	70.86	65.28	45.38	**72.73**
	LSTM	**81.88**	**93.24**	35.59	60.24
	Lasso	56.25	57.33	24.24	59.52

Continued on next page

Table 9 – *continued from previous page*

Model	**Clf.**	**2Wiki** *(in-dom.)*	**FRAMES** *(in-dom.)*	**Webshop** *(in-dom.)*	**DeepShop** *(in-dom.)*
	LR	57.67	56.38	27.27	56.47
Qwen3-VL-30B	RF	92.52	**81.82**	63.16	77.78
	XGB	**92.72**	80.27	**73.53**	**83.78**
	LSTM	88.61	79.75	57.72	61.11
	Lasso	85.16	70.00	56.58	70.13
	LR	87.01	70.89	59.21	69.23
Qwen3-VL-8B	RF	**82.61**	**83.45**	70.73	**68.42**
	XGB	**82.61**	81.38	**75.15**	**68.42**
	LSTM	67.20	74.45	57.99	40.54
	Lasso	66.67	68.57	51.85	44.78
	LR	68.09	70.42	51.09	41.18
Qwen3.5-27B	RF	84.15	94.94	80.56	86.49
	XGB	**90.91**	97.40	**82.19**	**87.67**
	LSTM	64.20	**98.67**	70.50	75.32
	Lasso	74.84	97.40	74.17	86.11
	LR	76.25	96.77	73.47	84.93
Qwen3.5-9B	RF	58.72	55.12	73.02	59.70
	XGB	**63.72**	**65.22**	**74.07**	**64.86**
	LSTM	38.98	56.29	58.16	50.49
	Lasso	42.11	48.12	60.87	59.74
	LR	44.64	48.89	59.57	60.53
UI-TARS-1.5-7B	RF	89.93	88.20	90.14	77.78
	XGB	**91.28**	**90.07**	**92.09**	**82.86**
	LSTM	89.80	86.71	77.04	74.29
	Lasso	84.00	70.44	75.18	71.43
	LR	86.11	72.96	75.18	72.46
Macro F1 (14 models)					
All	RF	75.75	69.95	72.31	65.42
	XGB	**79.36**	**75.27**	**74.23**	**72.57**
	LSTM	66.75	73.15	65.14	59.54
	Lasso	67.31	64.91	62.99	61.05
	LR	67.61	65.41	63.00	60.92

Table 10: OOD Per-Agent identification F1 (%) across datasets and classifiers. We report the Macro F1 for each classifier at the bottom. Best F1 per model group in **bold**

Model	Clf.	2Wiki→FRAMES *(OOD)*	FRAMES→2Wiki *(OOD)*	DeepShop→Webshop *(OOD)*	Webshop→DeepShop *(OOD)*
Proprietary Models					
GPT-5.4	RF	17.58	**51.11**	**65.53**	59.71
	XGB	17.02	50.97	60.43	**62.05**
	LSTM	**36.30**	39.74	48.05	60.34
Claude Opus 4.6	RF	23.24	31.47	**50.37**	37.91
	XGB	**26.87**	30.69	46.65	37.22
	LSTM	14.61	**35.04**	27.42	**38.50**
Gemini-3.1-Pro	RF	16.62	32.07	**58.89**	40.58
	XGB	21.23	34.66	57.05	38.77
	LSTM	**21.73**	**37.75**	54.11	**52.91**
Gemini-3-Flash	RF	36.96	**55.08**	39.95	40.49
	XGB	**37.78**	53.45	**52.85**	**53.39**
	LSTM	34.62	45.56	40.72	44.37
Seed-2.0-Lite	RF	67.05	86.05	89.03	74.61
	XGB	69.79	**90.73**	**89.60**	**75.77**
	LSTM	**71.39**	84.73	48.80	73.74
Open-Source Models					
Gemma-4-31B-it	RF	49.01	42.11	46.89	44.84
	XGB	43.61	44.34	**48.26**	43.75
	LSTM	**52.37**	**51.94**	40.68	**52.57**
Gemma-4-26B	RF	35.19	51.39	21.34	25.00
	XGB	**35.77**	**55.26**	**28.51**	**35.78**
	LSTM	35.25	53.58	15.22	22.83
GLM-4.6V	RF	35.15	22.87	25.41	**65.08**
	XGB	**35.73**	**30.93**	**60.08**	64.80
	LSTM	34.61	21.56	19.23	43.65
GLM-4.6V-Flash	RF	41.16	48.18	**54.58**	**23.70**
	XGB	44.44	44.22	42.93	21.80
	LSTM	**80.13**	**80.89**	37.26	18.78
Qwen3-VL-30B	RF	67.11	75.37	44.16	55.32
	XGB	**67.34**	77.01	**49.28**	**60.50**
	LSTM	66.53	**77.44**	26.73	50.22
Qwen3-VL-8B	RF	**73.74**	59.96	55.16	60.06
	XGB	71.28	**64.73**	**57.34**	**64.90**
	LSTM	63.91	54.04	44.37	57.06
Qwen3.5-27B	RF	0.15	0.61	**64.68**	**63.09**
	XGB	1.86	0.62	63.22	57.14
	LSTM	**21.98**	**10.98**	28.13	41.57
Qwen3.5-9B	RF	30.81	36.59	40.36	32.84
	XGB	**31.25**	36.95	**44.09**	39.07
	LSTM	30.50	**38.81**	30.61	**47.55**
UI-TARS-1.5-7B	RF	**74.72**	84.05	80.29	80.81
	XGB	71.05	82.12	**81.82**	**83.15**
	LSTM	70.06	**85.62**	77.44	79.42
All Models (Macro F1)					
All	RF	40.61	48.35	52.62	50.29
	XGB	41.07	49.76	**55.87**	**52.72**
	LSTM	**45.28**	**51.26**	38.48	48.82

Table 11: **Model families are also highly identifiable from their action traces.** There is a slight increase in overall performance for our XGBoost classifier when predicting by model families.

Family	2Wiki	FRAMES	WebShop	DeepShop
Seed-2	95.3	95.9	93.9	87.2
GPT-5	90.2	80.8	74.4	69.1
Claude 4	64.4	65.8	67.2	73.5
Gemini-3	68.2	72.5	75.2	60.9
Gemini-3-Flash	77.2	52.4	71.8	68.5
Gemma-4	86.5	77.6	76.7	69.9
GLM-4.6V	68.4	75.9	67.1	71.0
Qwen3-VL	90.7	94.7	91.3	87.3
Qwen3.5	75.4	76.6	79.6	76.4
UI-TARS-1.5	93.0	91.9	90.5	79.4
All (weighted)	80.7	76.6	77.6	74.8

Table 12: **Agent identifiability at both the model and family level.** Macro F1 for agent identification across 2Wiki, FRAMES, WebShop, and DeepShop. *Shaded header rows* show family-level classification (10-way); indented rows show individual-model classification (14-way). Each cell reports the best score across classifiers (RF, XGB, LSTM, Lasso, LR), with heat shading encoding value. Bottom row: weighted-average F1 (best classifier: XGB in all).

Model	2Wiki	FRAMES	WebShop	DeepShop
Seed-2 *(family)*	95.3	95.9	93.9	87.2
Seed-2.0-Lite	96.1	96.7	96.1	90.2
GPT-5 *(family)*	90.2	80.8	74.4	69.1
GPT-5.4	91.5	78.5	68.8	68.3
Claude 4 *(family)*	64.4	65.8	67.2	73.5
Claude Opus 4.6	70.5	68.7	67.1	75.0
Gemini-3 *(family)*	68.2	72.5	75.2	60.9
Gemini-3.1-Pro	65.2	75.2	75.2	61.1
Gemini-3-Flash *(family)*	77.2	52.4	71.8	68.5
Gemini-3-Flash	76.1	54.0	71.1	74.3
Gemma-4 *(family)*	86.5	77.6	76.7	69.9
Gemma-4-31B-it	82.1	84.8	74.2	63.4
Gemma-4-26B	72.9	64.4	72.3	57.6
GLM-4.6V *(family)*	68.4	75.9	67.1	71.0
GLM-4.6V	64.6	61.5	80.8	75.6
GLM-4.6V-Flash	81.9	93.2	54.1	72.7
Qwen3-VL *(family)*	90.7	94.7	91.3	87.3
Qwen3-VL-30B	92.7	81.8	73.5	83.8
Qwen3-VL-8B	82.6	83.5	75.2	68.4
Qwen3.5 *(family)*	75.4	76.6	79.6	76.4
Qwen3.5-27B	90.9	98.7	82.2	87.7
Qwen3.5-9B	63.7	65.2	74.1	64.9
UI-TARS-1.5 *(family)*	93.0	91.9	90.5	79.4
UI-TARS-1.5-7B	91.3	90.1	92.1	82.9
All – family (weighted)	80.7	76.6	77.6	74.8
All – model (weighted)	79.4	75.3	74.2	72.6

B.1 Confidence Intervals for Closed-Set Results

Figure 7 reports per-model closed-set F1 with 95% bootstrap confidence intervals, obtained by resampling test traces with replacement 500 times and taking the percentile interval. Bars show the bootstrap mean, which can differ slightly from the single-run point estimates in Figure 2 because of resampling on small test sets. Intervals are widest on DeepShop (38 test episodes).

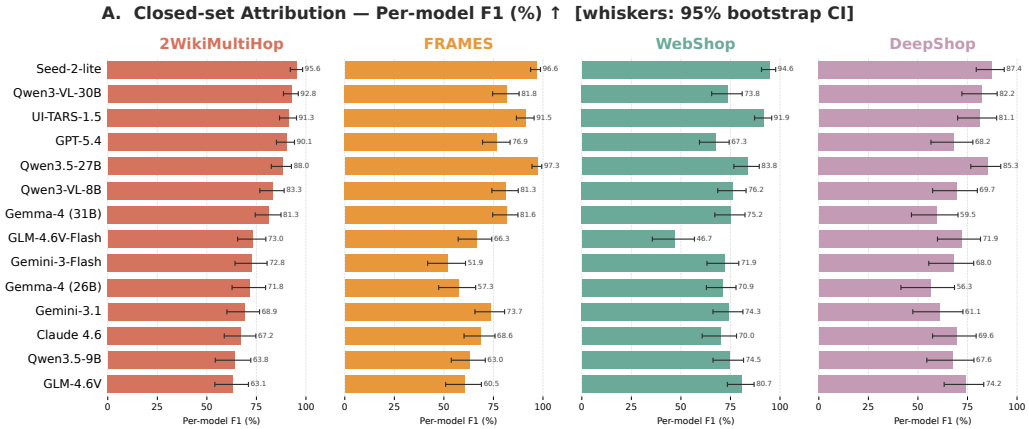


Figure 7: **Closed-set attribution with bootstrap confidence intervals.** Per-model F1 (%) with 95% bootstrap CIs over test traces; bars show the bootstrap mean.

B.2 Cost of Updating the Classifier as the Candidate Set Grows

We retrain the closed-set XGBoost classifier from scratch as the candidate set grows from 2 to 14 models, using MidScene.js traces from 2WikiMultiHopQA and WebShop and five seeds. Table 13 reports Macro-F1 and median fitting time at each candidate set size.

Table 13: **The classifier remains fast to retrain as the number of candidate models grows.** XGBoost is retrained from scratch on MidScene.js traces for the first k models. We report Macro-F1 (mean $\pm$ SD over five seeds) and median fitting time in seconds. Fitting time rises roughly linearly with k, reaching about 3 s at $k{=}14$, and median model size grows from about 0.2 MiB at $k{=}2$ to about 5.8 MiB at $k{=}14$.

		Macro-F1		Fit time (s)	
k	Chance ($1/k$)	2WikiMultiHopQA	WebShop	2WikiMultiHopQA	WebShop
2	0.500	0.968 ± 0.024	0.963 ± 0.029	0.23	0.25
4	0.250	0.919 ± 0.051	0.900 ± 0.053	0.72	0.71
6	0.167	0.883 ± 0.031	0.849 ± 0.070	1.13	1.16
8	0.125	0.851 ± 0.031	0.807 ± 0.045	1.60	1.64
10	0.100	0.807 ± 0.024	0.791 ± 0.030	2.10	2.13
12	0.083	0.793 ± 0.030	0.760 ± 0.021	2.57	2.62
14	0.071	0.779 ± 0.007	0.735 ± 0.004	3.01	3.13

Macro-F1 decreases as more candidates are added, from 0.968 and 0.963 at $k{=}2$ to 0.779 and 0.735 at $k{=}14$, but remains far above chance ($1/k$). Fitting time grows roughly linearly, reaching about 3 s at $k{=}14$.

B.3 Does Task Capability Predict Identifiability?

A central premise of our work is that each agent produces behaviourally distinct traces that are identifiable. A natural question is whether these traces are simply a proxy for task capability: if agents of similar capability tend to generate similar behavioural traces, our classifiers would be grouping agents by capability rather than recognising individual identity, and we would expect a significant correlation between task accuracy and identifiability. We explore this by examining whether such a correlation exists.

We measure capability as each agent's accuracy on FRAMES[34]. Each agent is given a maximum of 40 turns to attempt each of the 75 questions in the test set. We evaluate the agent's final response using an LLM-as-a-judge setup with `gpt-5.4-mini`, following the procedure described by Krishna et al. [34]; the evaluation prompt is provided below. Questions for which an agent failed to return a response within 40 turns are marked as incorrect; Table 14 reports completion counts per agent. We pair each agent's accuracy with its identifiability, defined as the macro-averaged F_1 of the best-performing classifier (XGBoost) in the closed-set fingerprinting setting, and compute both Pearson's r and Spearman's ρ across all 14 agents.

Figure 8 shows no meaningful relationship between the capability and identifiability. Neither correlation reaches statistical significance: Pearson's $r = 0.14$ ($p = 0.626$) and Spearman's $\rho = 0.05$ ($p = 0.852$). Agents span the full range of identifiability regardless of their accuracy: Claude Opus 4.6 achieves the highest accuracy (0.88) with moderate identifiability ($F_1 = 0.69$), while UITars-7B is among the most identifiable agents ($F_1 = 0.90$) yet records the lowest accuracy (0.03).

This supports the view that each agent carries a distinctive behavioural fingerprint, shaped by how it sequences actions, manages tool calls, and navigates pages, that is orthogonal to its task performance. Agents do not converge on shared patterns simply because they perform similarly; rather, behavioural identity persists across the capability spectrum.

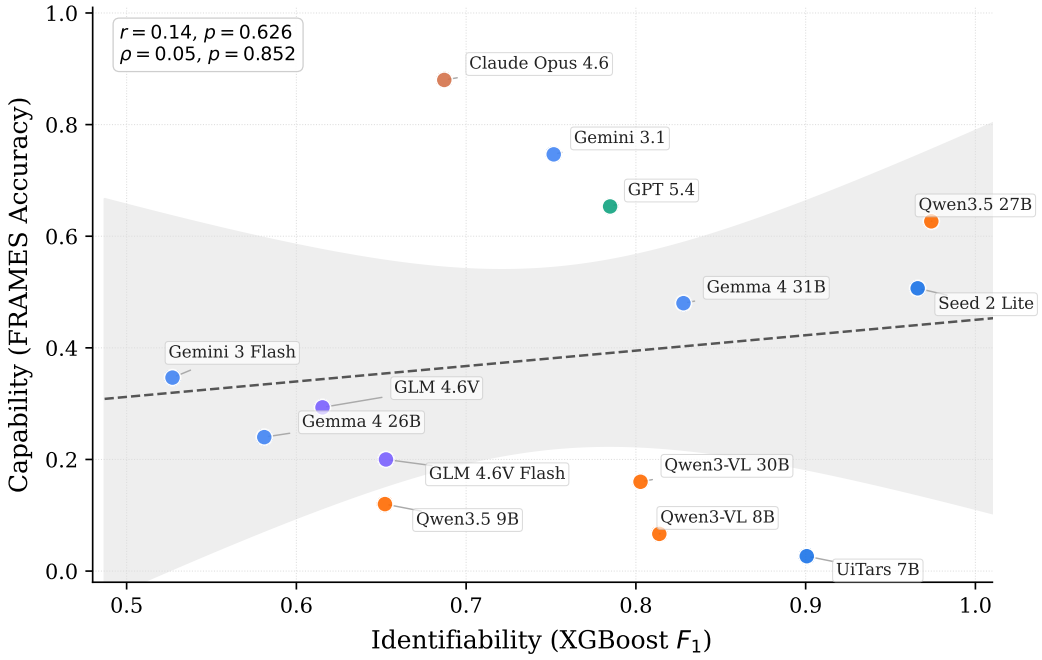


Figure 8: **Task capability does not predict agent identifiability.** Each point represents one of the 14 agents, with identifiability (XGBoost macro F_1 in the closed-set setting) on the x-axis and task capability (accuracy on FRAMES) on the y-axis. The dashed line shows the linear regression fit with its 95% confidence band. Neither Pearson's $r = 0.14$ ($p = 0.626$) nor Spearman's $\rho = 0.05$ ($p = 0.852$) is statistically significant, indicating that the behavioural signatures that make an agent identifiable are largely orthogonal to its task capability.

Table 14: Agent completion counts, penalised accuracy, and closed-set identifiability (XGBoost F_1) on the FRAMES test set (75 questions, 40-turn limit). Accuracy treats incomplete questions as incorrect. **Bold** denotes the highest value in each column.

Agent	Completed	Accuracy (%)	Ident. F_1 (%)
Claude Opus 4.6	72	**88.00**	68.71
Gemini 3.1	64	74.67	75.16
GPT-5.4	**75**	65.33	78.48
Qwen3.5-27B	54	62.67	**97.40**
Seed 2 Lite	50	50.67	96.60
Gemma-4-31B-it	58	48.00	82.80
Gemini 3 Flash	27	34.67	52.70
GLM-4.6V	46	29.33	61.54
Gemma-4-26B-A4B-it	26	24.00	58.11
GLM-4.6V Flash	39	20.00	65.28
Qwen3-VL-30B-A3B	35	16.00	80.27
Qwen3.5-9B	25	12.00	65.22
Qwen3-VL-8B	17	6.67	81.38
UITars-7B	10	2.67	90.07

Autorating Prompt

===Task===
I need your help in evaluating an answer provided by an LLM against a ground truth answer. Your task is to determine if the ground truth answer is present in the LLM's response. Please analyze the provided data and make a decision.

===Instructions===

1. Carefully compare the "Predicted Answer" with the "Ground Truth Answer".
2. Consider the substance of the answers – look for equivalent information or correct answers. Do not focus on exact wording unless the exact wording is crucial to the meaning.
3. Your final decision should be based on whether the meaning and the vital facts of the "Ground Truth Answer" are present in the "Predicted Answer".

===Input Data===

- Question: `{question}`
- Predicted Answer: `{LLM_response}`
- Ground Truth Answer: `{ground_truth_answer}`

===Output Format===
Provide your final evaluation in the following format:
"Explanation:" (How you made the decision?)
"Decision:" ("TRUE" or "FALSE")

Please proceed with the evaluation.

B.4 How well do our classifiers transfer across tasks and websites?

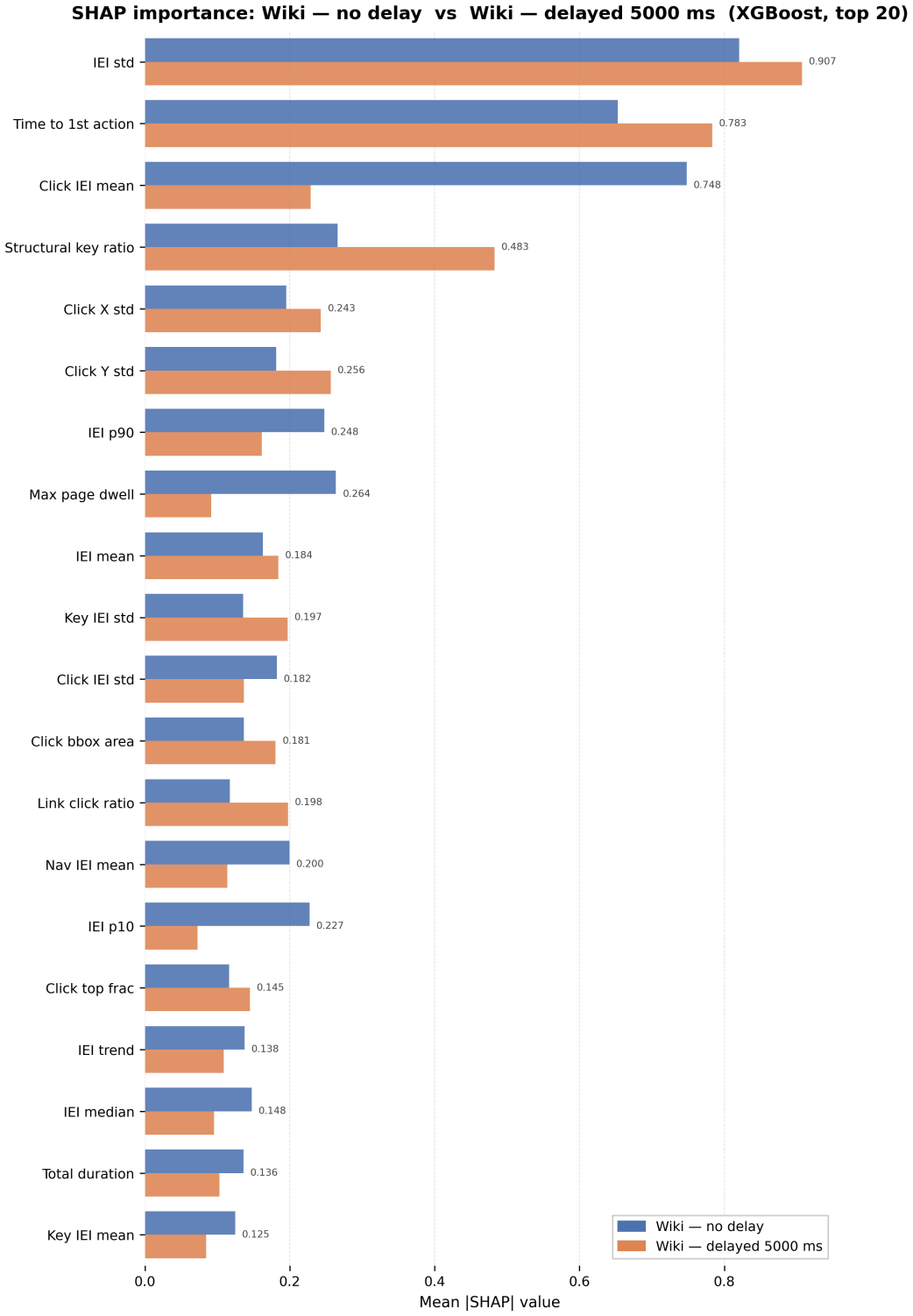
The main paper reports in-domain attribution, where train and test traces come from the same task distribution. Here we ask how far these fingerprints transfer across task and website boundaries. We compare three regimes: cross-task transfer, where a classifier is trained on one task and evaluated on another task hosted on the same website; pooled-site training, where traces from multiple tasks on the same website are combined before testing on each held-out test set; and cross-site transfer, where a classifier trained on one website is evaluated on another.

Table 15: **Training on diverse varied behaviours increases classifier performance.** We present generalisation results across task and website boundaries. Single-task transfer across tasks on the same site is substantially weaker than in-domain attribution, but pooling multiple tasks from the same website recovers strong performance. Cross-site transfer remains weak, suggesting that behavioural fingerprints are site-conditioned rather than universal.

Train	Test	Website	Setting	Macro F1
2Wiki	2Wiki	Wikipedia	in-domain	79.4
FRAMES	FRAMES	Wikipedia	in-domain	75.3
WebShop	WebShop	Amazon	in-domain	74.3
DeepShop	DeepShop	Amazon	in-domain	72.6
2Wiki	FRAMES	Wikipedia	cross-task	41.1
FRAMES	2Wiki	Wikipedia	cross-task	49.8
2Wiki + FRAMES	2Wiki test	Wikipedia	pooled-site	81.3
2Wiki + FRAMES	FRAMES test	Wikipedia	pooled-site	77.2
WebShop	DeepShop	Amazon	cross-benchmark	52.7
DeepShop	WebShop	Amazon	cross-benchmark	55.9
WebShop + DeepShop	WebShop test	Amazon	pooled-site	78.8
WebShop + DeepShop	DeepShop test	Amazon	pooled-site	70.9
Wikipedia pooled	Amazon test	Amazon	cross-site	29.7
Amazon pooled	Wikipedia test	Wikipedia	cross-site	26.0

The results in Table 15 show that behavioural fingerprints are not universal task-invariant signatures. Training on one Wikipedia task and testing on the other yields much weaker attribution than in-domain training, with macro F1 dropping from 79.4/75.3 in-domain to 41.1 and 49.8 under cross-task transfer. However, pooling 2WikiMultiHopQA and FRAMES training traces recovers strong attribution on both held-out test sets, reaching 81.3 on 2WikiMultiHopQA and 77.2 on FRAMES. This suggests that a site operator does not need a fingerprint that transfers from a single task to all possible tasks. Instead, diverse traces collected on the same website are sufficient to learn a robust site-conditioned identifier. Cross-site transfer remains much weaker, indicating that the identifying signal is shaped by the interaction between the model, task distribution, harness, and website interface.

B.5 Which features are important to our classifiers ?



B.6 Generalisation Across Agent Harnesses

Our main experiments train and test our classifiers on traces from a single agent harness, MidScene.js. In deployment, however, a website will be visited by multiple harnesses. To verify that our fingerprinting approach is not specific to MidScene.js and can handle a mixture of harnesses, we also collect traces with the Browser Use harness.

We evaluate six models: Qwen3.5-27B, GLM-4.6V, Gemma-4-26B-A4B-it, GPT-5.4, Gemini-3.1, and Claude Opus 4.6. We use 2WikiMultiHopQA and WebShop and retain the same task-level train, validation, and test splits used in the main experiments. We train classifiers on traces collected using MidScene.js, Browser Use, or a pooled set containing traces from both harnesses. All results are averaged across five classifier seeds and reported as mean Macro-F1 $\pm$ standard deviation across seeds.

Table 16: Cross-harness model attribution. Training and testing within the same harness produces strong attribution, whereas zero-shot transfer to an unseen harness falls to approximately chance ($1/6 \approx 0.167$). Training on both observed harnesses recovers strong performance when evaluating on either harness.

Train harness	Test harness		
	MidScene.js	Browser Use	Mixed
2WikiMultiHopQA			
MidScene.js	0.812 ± 0.012	0.099 ± 0.004	–
Browser Use	0.104 ± 0.007	0.795 ± 0.007	–
Mixed	0.750 ± 0.010	0.777 ± 0.012	0.758 ± 0.008
WebShop			
MidScene.js	0.827 ± 0.008	0.164 ± 0.013	–
Browser Use	0.214 ± 0.014	0.807 ± 0.003	–
Mixed	0.808 ± 0.009	0.738 ± 0.013	0.771 ± 0.008

As shown in Table 16, the original within-harness finding replicates with Browser Use: Macro-F1 ranges from 0.795 to 0.827 when training and evaluation use the same harness. However, classifiers trained using one harness do not transfer to the other. Cross-harness Macro-F1 ranges from 0.099 to 0.214, approximately the six-way chance level.

Training on pooled traces from both harnesses recovers most within-harness classifier performance, with Macro-F1 between 0.738 and 0.808 when evaluated on either harness separately. The fingerprint reflects an interaction between model and harness: the harness fixes what actions are available and how the interaction is structured, and the model decides how to use them. Training on a fixed harness therefore identifies model-specific variation within that setting rather than yielding a harness-invariant classifier.

B.7 Generalisation Across Inference Engines

Because timing features contribute substantially to our fingerprinters, a possible confound is that the classifier identifies the inference infrastructure rather than the underlying model. We investigate this using three open-weight models—GLM-4.6V, GLM-4.6V-Flash, and Gemma-4-26B-A4B-it—served under both vLLM and SGLang. All agents use the same Browser Use harness and identical task and browser configurations.We evaluate classifiers using the full feature set, action-timing features only, and action-content features only. Results are averaged across five classifier seeds.

Action-timing features describe intervals between browser events and therefore capture the complete observe–reason–act loop rather than privileged server-side decoding latency. Action-content features describe observable behaviour such as action types, click positions, key usage, scrolling, and navigation structure.

Table 17: Model attribution across inference engines. Zero-shot transfer between vLLM and SGLang remains strong in the tested three-model setting. Action-content features remain predictive without timing information, although timing features achieve higher Macro-F1.

Train $\rightarrow$ test	Full features	Action timing	Action content
vLLM $\rightarrow$ SGLang	0.843 ± 0.005	0.848 ± 0.002	0.702 ± 0.005
SGLang $\rightarrow$ vLLM	0.829 ± 0.005	0.802 ± 0.005	0.727 ± 0.008
Mixed $\rightarrow$ vLLM	0.968 ± 0.004	0.934 ± 0.008	0.735 ± 0.010
Mixed $\rightarrow$ SGLang	0.936 ± 0.007	0.937 ± 0.002	0.706 ± 0.005

Table 17 shows that zero-shot transfer between inference engines remains strong: classifiers trained on one engine and evaluated on the other achieve Macro-F1 between 0.829 and 0.843. Pooling traces from both engines further increases full-feature Macro-F1 to 0.936–0.968.

Timing-only attribution also transfers strongly across the two engines. Nevertheless, action-content features alone achieve Macro-F1 of 0.702–0.727, demonstrating that identification is not exclusively driven by timing. In this controlled subset, the fingerprint is therefore considerably less sensitive to the inference engine than to the agent harness. This experiment covers only three models and two serving engines, however, and does not systematically vary GPU hardware or other deployment parameters.

B.8 Prompt Normalisation as a Defence

In addition to randomising interaction timing, we evaluate whether prescribing a standard browsing procedure can suppress model-specific behavioural variation. Our prompt-normalisation defence instructs agents to issue one concise search, inspect results in order, verify product constraints, avoid unnecessary revisits, take one action at a time, and stop after selecting the best product. We evaluate this defence on WebShop using Browser Use and four vLLM-served models: Gemma-4-26B-A4B-it, GLM-4.6V, Qwen3.5-27B, and Qwen3-VL-30B-A3B. Results are averaged across five classifier seeds.

We replace the original WebShop instruction with the normalised prompt template shown in Box B.8.

Box 1: Prompt-normalisation defence

```
Find the product that best satisfies this request:

"{question}"

Follow this browsing procedure:

    1. Begin with one concise product search.
    2. Inspect relevant results in their displayed order.
    3. Verify the requested constraints on product pages.
    4. Avoid revisiting pages unless it is necessary to verify a constraint.
    5. Take one browser action at a time.
    6. Add only the best matching product or products to the cart, then stop.

Do not create an account, sign in, proceed to checkout, or make a purchase.
```

Table 18: Effect of prompt normalisation on model attribution. Normalisation disrupts a classifier trained on the original prompt distribution, but performance recovers when the attacker has access to traces generated under the normalised policy.

Classifier training data	Original test prompt	Normalised test prompt
Original prompts	0.917 ± 0.007	0.443 ± 0.029
Normalised prompts	–	0.955 ± 0.002
50/50 mixed prompts	0.889 ± 0.009	0.910 ± 0.005

As shown in Table 18, prompt normalisation substantially reduces attribution by a classifier trained on the original behaviour distribution, lowering Macro-F1 from 0.917 to 0.443. However, this reduction primarily reflects distribution shift rather than the removal of identifying information. Retraining on normalised traces recovers Macro-F1 to 0.955, while training on an equal mixture of the two prompt regimes achieves 0.889 and 0.910 on the original and normalised test distributions, respectively. Prompt normalisation may therefore increase the data required by an attacker when the policy is private or changes over time, but it does not provide robust protection once matching labelled traces are available. Moreover, this defence relies on model compliance with textual instructions; we do not evaluate a harness-enforced normaliser that directly standardises action timing, coordinates, or interaction sequences.

B.9 Identification versus Observed Events

We train XGBoost on full traces and evaluate using only the first k events of each test trace, reporting Macro-F1 averaged over the four datasets. Table 19 complements Figure 6B with absolute event counts. Performance is low for the first 20 events, then rises steadily and reaches 95% of full-trace Macro-F1 after 100 events.

Table 19: **Identification improves steadily with the number of observed events.** Macro-F1 of a classifier trained on full traces and evaluated on the first k events of each test trace, averaged over the four datasets. The range is across datasets. Per-model F1 is computed over the 14 models.

Events	Mean Macro-F1	Range	% of full trace	Median per-model F1	Best per-model F1	Models with F1 ≥ 0.5
5	0.12	0.07–0.17	16%	0.02	0.60	0.8 / 14
10	0.27	0.23–0.34	36%	0.24	0.81	2.5 / 14
20	0.29	0.26–0.34	38%	0.26	0.71	2.0 / 14
30	0.41	0.21–0.63	54%	0.40	0.70	5.0 / 14
50	0.56	0.37–0.77	74%	0.55	0.86	8.5 / 14
75	0.68	0.63–0.78	91%	0.68	0.89	12.5 / 14
100	0.72	0.69–0.79	95%	0.71	0.93	13.0 / 14
150	0.74	0.70–0.79	98%	0.74	0.94	13.0 / 14
Full	0.75	0.72–0.79	100%	0.76	0.94	13.8 / 14

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state three claims: (i) agent actions and timing are sufficient to identify the underlying model with up to 96% F1, (ii) the attack generalises across model sizes and families and supports open-set detection, and (iii) randomised timing delays do not provide robust protection against an adapted classifier. Each claim is supported by experiments in Sections 5 and 6 and Figures 2 to 6, with the 96% number matching the per-agent F1 reported for Seed-2-lite on 2WikiMultiHopQA.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 8 (Limitations) discusses the dependence on the harness, the two harnesses and two websites tested, the lack of mixed human and agent traffic, the inference engine and hardware not varied beyond one server. In the main text, we acknowledge the weakness of open-set identification and room for improvement.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: The paper is empirical and contains no formal theoretical results.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 4 specifies the four datasets and their splits (Table 1), the 14 evaluated models with citations (Section A.1), the agent harnesses (Midscene.js, Browser Use) in Apppendix B.6, the trace collection procedure (injected JS observer, fresh browser per session, millisecond timestamps), the five classifier families with hyperparameters (Appendix A.3), and the closed-set and open-set evaluation protocols. Behavioural feature definitions are given in Appendix A.5. We additionally release the trace corpus and harness.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

Justification: We release our labelled trace corpus across the four web environments and the browser-instrumentation harness, with documentation sufficient to reproduce the closed-set and open-set classification results. A GitHub repository is provided at `https://github.com/KabakaWilliam/known_actions/`.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Train/validation/test splits are reported per-dataset in Table 1. Classifier families (Lasso, Logistic Regression, Random Forest, XGBoost, LSTM) and their hyperparameter settings are described in Appendix A.3. Feature definitions are in Appendix A.5, and evaluation protocols (per-agent F1, macro F1, leave-one-agent-out AUROC) are specified in Sections 3 and 4.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Closed-set per-model F1 is reported with 95% bootstrap confidence intervals over test traces (Appendix B.1), and the new harness, inference-engine, prompt-normalisation, scaling and open-set experiments report mean and standard deviation over five classifier seeds. The main-text per-model F1 values and the open-set AUROCs for the 14 leave-one-agent-out runs are single-run point estimates. We did not repeat trace collection, as the cost of 14 frontier models across four live environments precluded it.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Section 4 reports that all open-source models were served via vLLM on a node with two NVIDIA H100 GPUs, with closed-source frontier models accessed via their respective APIs and Seed-2-lite via OpenRouter. Classifier training (XGBoost, Random Forest, Lasso, Logistic Regression, LSTM) is lightweight and runs on a single CPU. Total classifier hyperperams sweep are reported in Appendix A.3. It takes 3 days to collect all traces for the agents. The total cost of generating the traces for closed-source models is provided in Appendix A.4. The additional Browser Use trace collection cost USD 680 and the SG-Lang ablation was about 45 H100-hours.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research involves no human subjects, scraped personal data, or copyrighted material beyond standard public benchmarks (2WikiMultiHop, FRAMES, WebShop, DeepShop). Traces were generated by automated agents under our control on public websites used for benchmark evaluation in prior work. We have reviewed the NeurIPS Code of Ethics and conform to it.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Section 7 (Implications) explicitly characterises the negative impacts of agent fingerprinting: model-specific prompt injection, adversarial cost inflation targeting the user's inference budget, and agent-specific access control including model-conditional content poisoning.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The released artefacts are interaction traces from automated agents on public benchmark websites and a browser-instrumentation harness. The traces contain no personally identifying information, no scraped private content, and no exploit code. The classifiers used in the paper are commodity supervised models that any practitioner can retrain from a small number of traces (as described in Section 6), so gating the corpus would impose friction on defensive research without materially affecting an adversary's ability to reproduce the attack. We therefore release the corpus and harness openly to support defensive and auditing research.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All existing assets are cited: the four benchmarks (2WikiMultiHop, FRAMES, WebShop, DeepShop), the 14 LLMs evaluated, and the Midscene.js harness. Models are accessed under their respective terms of use (open-weight licences for locally hosted models, API terms for closed-source models, OpenRouter terms for Seed-2-lite). Benchmark datasets are used in their original form for evaluation, consistent with their published licenses.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The corpus of agent interaction traces and the browser-instrumentation harness are documented with a README describing the trace schema, the per-session fields, the JavaScript observer, the per-dataset splits, and example scripts to reproduce the closed-set and open-set classification results.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The study involves no human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: LLMs are the subject of study of the paper. The 14 evaluated models, their access modalities (local vLLM, OpenRouter, vendor APIs), and the Midscene.js harness through which they interact with web environments are documented in Section 4 and Appendix A.1.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.