MirrorCAPTCHA: Wild CAPTCHA, Wild Distribution, Wild Web-based Platform Meet Multimodal LLM Agents

Xiangyu Wu[1,2], Yuwei Hu[1], Tianyu Cui[2], Yueying Tian[3], Qing-Guo Chen[2], Zhao Xu[2], Weihua Luo[2], Kaifu Zhang[2], Yang Yang[1], Jianfeng Lu[1]

[1]Nanjing University of Science and Technology,
[2]Alibaba Group, [3]University of Sussex

Abstract

The path to fully autonomous web agents is currently hindered by a critical bottleneck: their limited ability to handle CAPTCHA. Existing agent benchmarks largely ignore this practical challenge, failing to evaluate an agent's real-world capacity to solve CAPTCHA. To bridge this gap, we conduct a comprehensive analysis of real-world CAPTCHA distributions and introduce **MirrorCAPTCHA**, a benchmark annotated with *Weighted Pass Rate* and a newly proposed metric *Completion Degree*. MirrorCAPTCHA is designed to serve as a "mirror" that faithfully reflects the automation capabilities of agents in real scenarios. We filter $2,095$ websites from Common Crawl, identify the CAPTCHA deployed on these sites, and cluster them into 18 distinct categories using *K-means* algorithm. To ensure practicality, we extract a web subgraph from Common Crawl covering these websites and use random walks to simulate real-world CAPTCHA encounter frequencies, yielding a realistic measure of agents' ability. Additionally, we develop a lightweight synthetic data pipeline to train `Ovis2-Agent-CAPTCHA-8B`, which significantly outperforms current state-of-the-art closed-source models on MirrorCAPTCHA, achieving a 9.4% higher average *Weighted Pass Rate* and a 2.13% higher average *Completion Degree* than the runner-up, `Gemini-2.5-Pro`.

1 Introduction

Multimodal web agents (He et al., 2024; Lai et al., 2024; Agashe et al., 2025; Huq et al., 2025), powered by multimodal large language models (MLLMs) (Wang et al., 2024; Lu et al., 2024; Chen et al., 2024), are designed to perform repetitive online tasks (*e.g.*, shopping, navigation, and booking) by simulating human behavior. However, a major obstacle to full automation is the requirement to pass CAPTCHA verification during common activities like registration and login. While agents

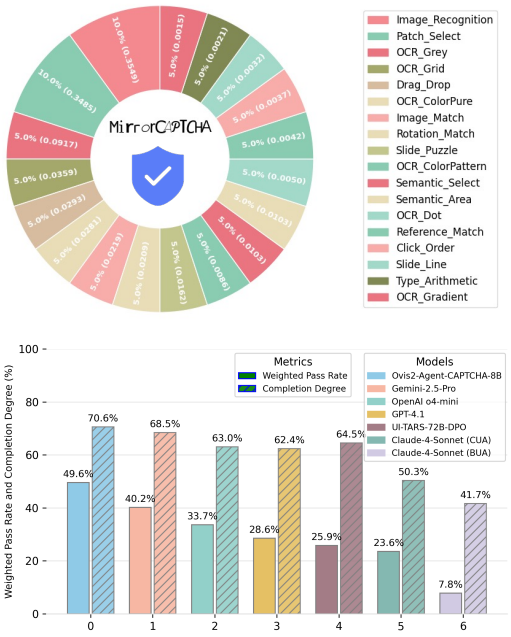

Figure 1: Up part: MirrorCAPTCHA distribution. `CUA` and `BUA` denote Computer-Use Agents and Browser-Use Agents. Bottom part: Performance of web agents.

can readily handle non-visual CAPTCHAs (*e.g.*, SMS, email), autonomously solving complex visual CAPTCHAs, such as grid selection, character recognition, and slider puzzles, remains essential for widespread deployment. Crucially, it remains unclear whether current agents can solve complex CAPTCHA in the wild with human-level speed and accuracy.

Mainstream web agent benchmarks (*e.g.*, VisualWebArena (Koh et al., 2024), AgentBench (Liu et al., 2024), and ST-WebAgentBench (Levy et al., 2025)) simulate real online environments but often omit prevalent CAPTCHA challenges. Although recent works have introduced CAPTCHA-specific benchmarks, notable limitations persist. For example, Open CaptchaWorld (Luo et al., 2025) proposes the first interaction-based benchmark, but its dataset is extremely small, which fails to reflect real-world distributions and also omits common Optical Character Recognition (OCR) CAPTCHA.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 31001–31017
July 2-7, 2026

Figure 2: MirrorCAPTCHA filters 2095 websites with deployed CAPTCHAs from desktop webpages, covering 18 categories, reflecting real-world CAPTCHA distribution. 1. Image Recognition, 2. Patch Select, 3. OCR Grey, 4. OCR Grid, 5. Drag Drop, 6. OCR ColorPure, 7. Image Match, 8. Rotation Match, 9. Slide Puzzle, 10. OCR ColorPattern, 11. Semantic Select, 12. Semantic Area, 13. OCR Dot, 14. Reference Match, 15. Click Order, 16. Slide Line, 17. Type Arithmetic, 18. OCR Gradient.

MCA-Bench (Wu et al., 2025) constructs a larger-scale synthetic benchmark, yet it is relatively homogeneous and lacks practical realism; moreover, some CAPTCHA types are insufficiently challenging, making it difficult to accurately assess agents' capabilities in the wild.

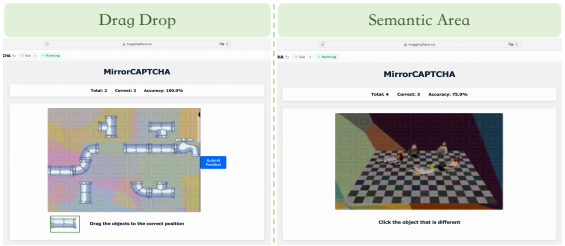

Figure 3: Web-based CAPTCHA evaluation platform.

To address these issues, we develop **MirrorCAPTCHA**, a benchmark designed to serve as a "mirror" of real-world CAPTCHA distributions and to accurately assess web agents' practical automation abilities. We filter out 2095 valid websites from Common Crawl (Crawl, 2007) that deploy active CAPTCHAs, and cluster them into 18 distinct categories. From these sites, we curate 1000 unique CAPTCHA puzzles spanning a range of deep learning and web interaction skills, as shown in Figure 2. Notably, CAPTCHA types occur with markedly different frequencies in the wild. To capture this, we extract a web subgraph from Common Crawl that covers the selected websites and estimate the encounter frequency of each CAPTCHA type via random walks on the subgraph until the visitation distribution converges. This ensures higher-frequency CAPTCHA types are assigned a larger weight in the evaluation. As a result, strong performance on MirrorCAPTCHA is more indicative of an agent's effectiveness in real-world deployment.

Beyond the standard *Weighted Pass Rate*, we introduce a customized metric, *Completion Degree*, for a subset of CAPTCHA types. While pass rate measures binary success, *Completion Degree* quantifies the "degree" to which an agent solves a CAPTCHA, offering a more nuanced measure of its reliability. All puzzles are tested on interactive webpages, as shown in Figure 3, to fully simulate the scenarios web agents face in the real world. Agents must perceive screenshots and execute actions like clicking, pressing keys, and dragging UI elements until the verification process terminates.

In addition, we develop a lightweight and scalable data synthetic pipeline to train a model named `Ovis2-Agent-CAPTCHA-8B` on 370k synthetic CAPTCHA samples. Experiments on MirrorCAPTCHA show that `Ovis2-Agent-CAPTCHA-8B` significantly outperforms state-of-the-art closed-source models. For example, on the high-traffic "Patch Select" category, it surpasses `Gemini-2.5-Pro` by 30.66% in *Weighted Pass Rate*. The model's strong performance on both new metrics, including a high score in *Completion Degree* on challenging puzzles, highlights its potential for real-world web automation and sets a new state-of-the-art for multimodal agents on CAPTCHA challenges.

2 Related Works

Web Agents. Web Agents (Gur et al., 2024; He et al., 2024; Lai et al., 2024; Agashe et al., 2025; Huq et al., 2025; Shao et al., 2025; Erdogan et al., 2025), built on large foundation models (Dubey et al., 2024; Yang et al., 2025), aim to simulate human behavior and automate repetitive web tasks. These agents typically follow a three-stage pipeline: perception (interpreting visual information from screenshots and text), planning/reasoning (decomposing tasks and generating actions), and execution (localizing UI elements and performing interactions). Recent advances such as AutoGPT (Significant Gravitas, 2023) demonstrate the ability to handle complex tasks with minimal user intervention. Similarly, multimodal agents like WebVoyager (He et al., 2024) and MMAC-Copilot (Song et al., 2024) leverage advanced models (e.g., GPT-4V (Yang et al., 2023) and Gemini Vision (Anil et al., 2023)) to process diverse inputs, including screenshots and videos. Training strategies for these agents commonly include data preprocessing and augmentation, as well as various fine-tuning methods, all aimed at improving their end-to-end performance.

Captcha Benchmarks and Models. Advances in deep learning have significantly improved CAPTCHA recognition. Early methods relied on CNNs for feature extraction (Thobhani et al., 2020; Tang, 2024), while later work combined CNNs and RNNs to handle variable-length CAPTCHA sequences (Hu et al., 2018; Derea et al., 2023). Generative adversarial networks (GANs) have also been used to synthesize large datasets for training CAPTCHA solving models (Shu and Xu, 2019; Ye et al., 2020). However, many such models are often style-specific and fail to generalize to the diverse CAPTCHA variants encountered in practice. Existing benchmarks suffer from similar limitations. BeCAPTCHA-Mouse, for example, focuses on mouse trajectories over synthetic CAPTCHA types, while Open CaptchaWorld (Luo et al., 2025) omits common OCR-based CAPTCHAs and is limited by a small dataset. MCA-Bench (Wu et al., 2025) evaluates vision-language models against synthetic, homogeneous CAPTCHA puzzles that do not reflect the diversity and complexity of real-world challenges. This gap, caused by a lack of benchmarks grounded in real-world distributions, prevents an accurate assessment of web agents' practical CAPTCHA-solving performance.

3 MirrorCAPTCHA

MirrorCAPTCHA is a carefully curated benchmark of real-world CAPTCHAs that are challenging for agents yet easy for humans. Most puzzles are collected directly from real websites and manually annotated, with a small portion sourced from MCA-Bench. MirrorCAPTCHA includes two metrics: *Weighted Pass Rate* (WPR) and a newly introduced metric, *Completion Degree* (CD), which applies to a subset of CAPTCHA types.

3.1 Desktop Web Curation

MirrorCAPTCHA targets web agents that operate in desktop browsing environments. To this end, we first collect a large list of frequently visited and accessible desktop websites. Common Crawl (Crawl, 2007) provides a web graph of global internet traffic over the past 6 months, comprising 156.1 million nodes and 2.1 billion edges. Each node denotes a website accessed from a specific device (*e.g.*, desktop or mobile), and each edge corresponds to a browsing transition.

From this graph, we select the top $15,000$ nodes based on degree as initial candidate sites. We then use a modified version of `WebVoyager` (He et al., 2024) to query `Claude-4-Sonnet` for assessing their accessibility, filtering out unreachable or inaccessible webpages (see Figure 4, top). The resulting corpus contains $10,000$ valid websites spanning diverse domains, including entertainment, media, and social network platforms.

3.2 CAPTCHA-Confronted Web Curation

The next step is to identify websites that deploy CAPTCHA mechanisms. Standard user behaviors (*e.g.*, directly registering or logging in by users) sometimes do not trigger CAPTCHA, since such actions are not always flagged as suspicious. We therefore deploy autonomous agents to systematically navigate and interact with registration and authentication workflows. This increases the likelihood of triggering CAPTCHA puzzles and better reflects the conditions encountered by real-world web agents.

Figure 4 (middle) illustrates the entire free exploration process. We observe that when the agent fails to trigger a CAPTCHA on the current website, it will randomly move to other sites and continue searching for CAPTCHA puzzles. In practice, agents may also attempt to force task completion even when their actions become irrational, which

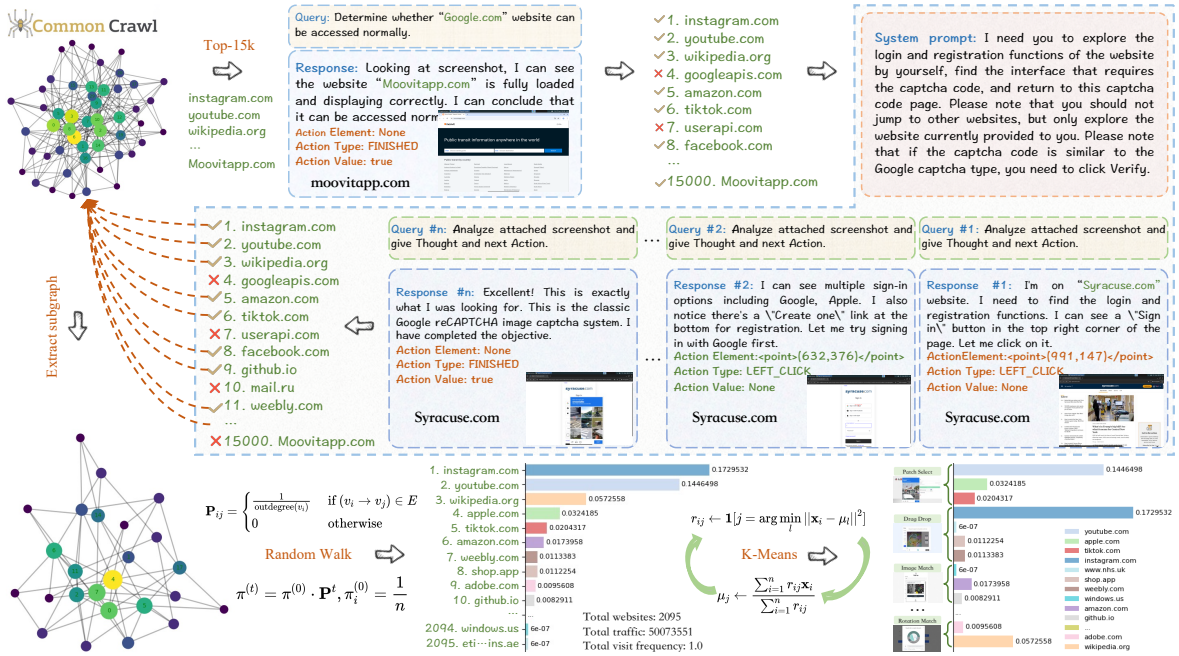

Figure 4: MirrorCAPTCHA construction pipeline. Top: modified WebVoyager querying `Claude-4-Sonnet` about website accessibility. Middle: `Claude-4-Sonnet` autonomously explores website functionalities that may trigger CAPTCHA (*e.g.*, registration, login, password reset, account recovery). Termination occurs when a CAPTCHA is triggered, the step limit is reached, or no CAPTCHA is confirmed. Bottom left: random walk for access probability estimation. Bottom right: K-means clustering for CAPTCHA categorization.

can introduce redundancy and noise. To mitigate this, we impose behavioral constraints:

- *Restrict exploration to Sign up or Log in*;
- *Prohibit navigation beyond given website*;
- *Prioritize interactions with UI elements likely to trigger CAPTCHA*.

These rules ensure reliability and reduce redundant exploration. Among the $10,000$ accessible websites, we identify $2,095$ sites that deploy CAPTCHAs, spanning multiple languages and diverse puzzle types. Although intuitive for humans, these puzzles remain challenging for agents. A further consideration is that the real-world encounter frequency of a specific CAPTCHA type is largely determined by the traffic of websites that host it. For example, high-traffic platforms (e.g., Google, Facebook, YouTube), moderately popular niche sites with relatively lower traffic (e.g., GitHub, Adobe), and numerous small websites with minimal traffic. To accurately model how often a web agent encounters each CAPTCHA type, a benchmark must account for this variation in website traffic among CAPTCHA deployments.

3.3 Web and CAPTCHA Access Probabilities

To simulate realistic cross-site browsing behavior, we perform random walks on a subgraph extracted from Common Crawl. Specifically, we extract the nodes and edges adjacent to the $2,095$ websites, obtaining a connected subgraph $\mathcal{G}_s$ with 15 million nodes and 75 million edges. Let $\mathbb{V} := \{v_1, v_2, ..., v_n\}$ denote its node set and $\mathbb{E} := \{e_1, e_2, ..., e_m\}$ the edge set. We define the transition matrix as:

$$\mathbf{P}_{ij} = \begin{cases} \frac{1}{\text{outdegree}(v_i)} & \text{if } (v_i \to v_j) \in \mathbb{E}, \\ 0 & \text{otherwise,} \end{cases} \tag{1}$$

where $\text{outdegree}(v_i)$ is the number of outgoing edges from the node v_i. We initialize the visit distribution uniformly over $\mathbb{V}$:

$$\pi_i^{(0)} = \frac{1}{n}, \quad \forall i = 1, 2, \ldots, n. \tag{2}$$

After t steps, the node visit distribution evolves as $\pi^{(t)} = \pi^{(t-1)} \cdot \mathbf{P}$. Equivalently, $\pi^{(t)} = \pi^{(0)} \cdot \mathbf{P}^t$, where $\pi_i^{(t)}$ denotes the probability of being at node v_i after t steps. After 10^8 steps (the cutoff used in our study), the final visit distribution is:

$$\pi^{(10^8)} = \pi^{(0)} \cdot \mathbf{P}^{10^8}, \tag{3}$$

where $\pi_i^{(10^8)}$ is the probability of visiting node v_i. The bottom-left panel of Figure 4 shows the resulting traffic distribution, where a handful of high-traffic websites (*e.g.*, `instagram.com`,

Table 1: Statistics of the MirrorCAPTCHA benchmark by category, including website coverage, visit traffic, relative frequency, example puzzles, task description, and number of samples.

No.	Type name	Covered	Traffic	Frequency	CAPTCHA description	Samples
1	Image Recognition	652	17961686	0.35871	Identify target objects grid in a 9-image grid	100
2	Patch Select	709	17449069	0.34847	Identify target objects patches in a 16-image grid	100
3	OCR Grey	95	4594340	0.09175	OCR: grayscale text, and line noise	50
4	OCR Grid	22	1798535	0.03592	OCR: grayscale text, grid background, and line noise	50
5	Drag Drop	58	1465736	0.02927	Drag small image to correct position on large image	50
6	OCR ColorPure	159	1408947	0.02814	OCR: color font, pure background, and color line noise	50
7	Image Match	18	1094478	0.02186	Select matching image from candidates based on reference	50
8	Rotation Match	6	1047564	0.02092	Rotate tile to correct position via slider	50
9	Slide Puzzle	103	811840	0.01621	Slide puzzle piece to correct position	50
10	OCR ColorPattern	46	428499	0.00856	OCR: color font, pattern background, and color line noise	50
11	Semantic Select	50	516790	0.01032	Select images from 3×3 grid following instructions	50
12	Semantic Area	34	514516	0.01027	Select the different icon from multiple similar ones	50
13	OCR Dot	57	248757	0.00497	OCR: grayscale text, pockmarked background, and line noise	50
14	Reference Match	26	209916	0.00419	Select from 3×3 grid based on references and instructions	50
15	Click Order	15	182529	0.00365	Click icons in specified sequence	50
16	Slide Line	13	159555	0.00319	Slide block to endpoint	50
17	Type Arithmetic	15	106586	0.00213	Solve arithmetic problem and enter result	50
18	OCR Gradient	17	74208	0.00148	OCR: grayscale font, gradient background, and line noise	50
Total	–	**2095**	**50073551**	**1.0**	–	**1000**

`youtube.com`, `wikipedia.org`) account for nearly half of all visits, and thus nearly half of CAPTCHA encounters. *The full visit probability distribution over $\mathbb{V}$ is provided in Appendix E.*

Next, we categorize CAPTCHA types into clusters by applying `K-means` clustering. For each website screenshot w_i, we extract the CLIP (Radford et al., 2021) image embedding $\mathbf{f}_i \in \mathbb{R}^d$:

$$\mathbf{f}_i = \phi(\mathrm{w}_i), \quad d = 512, \tag{4}$$

and stack the embeddings to form ($N = 2095$):

$$\mathbf{F} = [\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_N]^\top \in \mathbb{R}^{N \times d}. \tag{5}$$

We then apply `K-means` to partition $\mathbf{F}$ into K clusters $\{\mathrm{C}_1, \mathrm{C}_2, \ldots, \mathrm{C}_K\}$ using the standard iterative assignment-and-update procedure:

$$\begin{aligned} r_{ij} &\leftarrow \mathbf{1}\left[j = \arg\min_l \|\mathbf{f}_i - \boldsymbol{\mu}_l\|^2 \right], \\ \boldsymbol{\mu}_j &\Leftarrow \frac{\sum_{i=1}^N r_{ij}\mathbf{f}_i}{\sum_{i=1}^N r_{ij}}, \end{aligned} \tag{6}$$

where $\boldsymbol{\mu}_j$ is the centroid of cluster C_j, and r_{ij} indicates whether w_i is assigned to C_j. Starting from $K = 2$, we iteratively increase K, manually check and rectify the clustering results, and further split clusters when necessary to obtain finer-grained types. Ultimately, we obtain 18 distinct CAPTCHA types, as shown in the bottom-right panel of Figure 4.

Once clustering is complete, we compute the visit probability of each CAPTCHA type C_j by aggregating the node visit probabilities of all websites in that cluster:

$$p(\mathrm{C}_j) = \sum_{\mathrm{w}_i \in \mathrm{C}_j} \pi_i^{(10^8)}, \quad j = 1, 2, \ldots, K. \tag{7}$$

The resulting access frequency distribution over the 18 CAPTCHA types is summarized in Table 1, where the distribution shows a heavy-tailed pattern: a few dominant types (*e.g.,* distorted alphanumeric text and common image-based challenges) account for most real-world traffic, while many others occur rarely. This skew has direct implications for benchmark construction and for evaluating the robustness of automated CAPTCHA solvers.

The final step is to construct evaluation puzzles for each CAPTCHA type in proportion to its estimated visit frequency: higher-traffic categories are allocated more samples, thereby mirroring real-world CAPTCHA distribution patterns. However, some categories (*e.g.*, OCR Gradient, Type Arithmetic) have extremely low traffic. In a benchmark with $1,000$ samples, OCR Gradient would yield only 1-2 puzzles ($1,000 \times 0.00148$), which is overly sparse. Conversely, Image Recognition or Patch Selection could dominate with hundreds of samples, leading to redundancy.

To balance realism with statistical reliability, we cap the number of samples per category at 50 or 100, as shown in Table 1. *Appendix A compares our benchmark with OpenCaptchaWorld and MCA-Bench.* During evaluation, we still use the estimated visit frequencies as aggregation weights, ensuring that *Weighted Pass Rate* reflects real-world CAPTCHA distribution while avoiding ex-

treme sparsity or overrepresentation. We detail this weighting strategy in the following subsection.

3.4 Evaluation Metrics

MirrorCAPTCHA employs two metrics: *Weighted Pass Rate* (WPR) and *Completion Degree* (CD). WPR measures whether a model fully solves a CAPTCHA, weighted by real-world encounter probabilities, while CD quantifies how close a model comes to a complete solution. All CAPTCHA types can be evaluated with WPR, whereas CD is only evaluated for a subset of CAPTCHA types. For example, Image Match puzzles are strictly binary (match *vs.* non-match) and are therefore evaluated only with WPR.

Weighted Pass Rate (WPR). The visit probability $p(\mathrm{C}_j)$ for each CAPTCHA type is defined in Equation 7. Let N_i denote the number of puzzle samples in category i, and let S_i denote the number of puzzles that the model solves correctly. Then:

$$\mathbf{WPR} = \sum_{i=1}^{k} \left(p_i \times \frac{S_i}{N_i} \right) \times 100\% \qquad (8)$$

Completion Degree (CD). CD is defined for 12 categories using 4 task-specific metrics (*See Appendix D for full details*):

- `F1 score`: Image Recognition, Patch Select, Semantic Select, and Reference Match;
- `Levenshtein Distance`: OCR Grey, OCR Gradient, OCR Grid, OCR ColorPure, OCR ColorPattern, and OCR Dot;
- `Sequence Matching`: Click Order (one-to-one match count);
- `Angle Distance`: Rotation Match (angular difference metric);

Together, WPR and CD provide a more comprehensive evaluation of CAPTCHA-solving performance, capturing both strict success and partial progress.

4 Ovis2-Agent-CAPTCHA-8B

Unlike prior deep learning models that target a single CAPTCHA type (*e.g.*, 3×3 or 4×4 grids) with task-specific architectures, MirrorCAPTCHA evaluates the broader capability of MLLM-based web agents to solve diverse, real-world CAPTCHAs. This naturally raises a key question: *how can we improve a web agent's ability to generalize across a wide range of CAPTCHA types?*

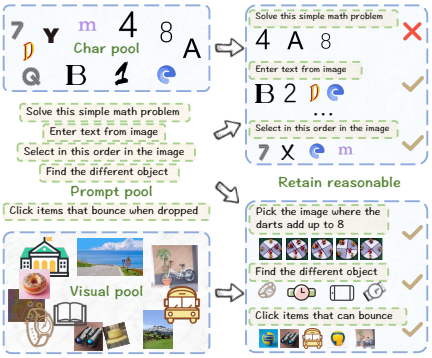

Figure 5: Synthesize CAPTCHA pipeline.

To this end, we design a lightweight and extensible CAPTCHA synthesis pipeline (Figure 5) that automatically generates puzzles by specifying simple, rule-based data organization templates. Using the resulting synthetic data, we perform SFT on Ovis2-8B (Lu et al., 2024) to obtain `Ovis2-Agent-CAPTCHA-8B`. The model is trained on 370k synthesized CAPTCHA samples (including Image Recognition, Patch Select, Semantic Area, OCR, and Type Arithmetic types), and is further augmented with a small amount of computer-use trajectory data to improve cross-scenario transfer. Training takes 35 hours on 64 H100 GPUs, enabling the model to acquire the visual grounding, semantic reasoning, and interaction skills required for CAPTCHA solving.

We benchmark `Ovis2-Agent-CAPTCHA-8B` against both open-source and closed-source baselines. Results show that it not only exceeds existing open-sourced models but also significantly outperforms closed-sourced systems on MirrorCAPTCHA, establishing a new baseline for CAPTCHA-solving web agents.

5 Experiments Analysis

5.1 Experimental Setup

We systematically evaluate both browser-use agents and computer-use agents, each equipped with different MLLM backbones, on the MirrorCAPTCHA benchmark. To ensure fairness, we adopt consistent prompting strategies and uniform evaluation metrics across all models. **Browser-use agents**, implemented with the set-of-mark (SOM) paradigm (Müller and Žunič, 2024), include OpenAI o4-mini (OpenAI, 2025), Gemini-2.5-Pro (Anil et al., 2023), Claude-4-Sonnet (Anthropic, 2025), and GPT-4.1 (OpenAI, 2025). **Computer-use agents** are deployed via the augmented WebVoyager framework (He et al., 2024),

Table 2: WPR on MirrorCAPTCHA for Browser-Use (OpenAI o4-mini, Gemini-2.5-Pro, Claude-4-Sonnet, GPT-4.1) and Computer-Use Agents (Claude-4-Sonnet, UI-TARS-72B-DPO, Ovis2-Agent-CAPTCHA-8B).

CAPTCHA Type	Browser-Use Agent				Computer-Use Agent		
	o4-mini	Gemini-2.5-Pro	Claude-4-Son	GPT-4.1	Claude-4-Son	UI-TARS	Ovis2-8B
Image Recognition	53.87	64.33	3.72	47.67	35.33	40.07	**66.67**
Patch Select	14.72	14.67	2.76	4.67	10.00	5.39	**45.33**
OCR Grey	50.00	**62.00**	30.00	54.00	38.00	48.00	54.00
OCR Grid	36.00	**54.00**	20.00	44.00	22.00	37.50	52.00
Drag Drop	0.00	0.00	0.00	0.00	0.00	0.00	0.00
OCR ColorPure	44.00	**62.00**	24.00	54.00	32.00	40.00	50.00
Image Match	6.67	**31.03**	14.29	13.33	30.00	23.33	6.67
Rotation Match	0.00	0.00	0.00	0.00	0.00	**4.00**	0.00
Slide Puzzle	0.00	0.00	0.00	0.00	0.00	**4.00**	0.00
OCR ColorPattern	62.00	**74.00**	28.00	64.00	32.00	52.00	60.00
Semantic Select	30.00	**56.67**	24.21	43.33	46.67	33.33	26.67
Semantic Area	0.00	0.00	0.00	0.00	40.00	**53.33**	16.67
OCR Dot	50.00	70.00	32.00	54.00	30.00	50.00	**74.00**
Reference Match	50.00	**83.33**	33.33	33.33	0.00	23.33	10.00
Click Order	0.00	0.00	0.00	0.00	0.00	**3.75**	1.25
Slide Line	0.00	0.00	0.00	0.00	50.00	**80.00**	20.00
Type Arithmetic	93.33	**96.67**	90.00	90.00	90.00	80.00	**96.67**
OCR Gradient	64.00	**70.00**	34.00	62.00	40.00	60.00	64.00
Average	33.66	40.22	7.77	28.58	23.58	25.85	**49.57**

Table 3: CD on MirrorCAPTCHA for Browser-Use (OpenAI o4-mini, Gemini-2.5-Pro, Claude-4-Sonnet, GPT-4.1) and Computer-Use Agents (Claude-4-Sonnet, UI-TARS-72B-DPO, Ovis2-Agent-CAPTCHA-8B).

CAPTCHA Type	Browser-Use Agent				Computer-Use Agent		
	o4-mini	Gemini-2.5-Pro	Claude-4-Son	GPT-4.1	Claude-4-Son	UI-TARS	Ovis2-8B
Image Recognition	84.27	80.88	22.83	66.77	76.22	72.11	**89.66**
Patch Select	72.82	70.22	33.49	57.53	61.60	57.24	**88.14**
OCR Grey	81.53	**84.52**	65.84	82.52	70.41	76.08	78.84
OCR Grid	76.13	**84.69**	56.21	**78.56**	60.26	72.68	78.10
OCR ColorPure	75.66	**85.30**	62.15	81.41	67.47	75.60	79.14
Rotation Match	0.00	0.00	0.00	0.00	0.00	**60.97**	60.27
OCR ColorPattern	85.20	**89.83**	64.52	86.20	68.26	76.15	79.53
Semantic Select	49.48	62.22	31.11	**64.32**	60.37	55.95	61.25
OCR Dot	76.00	**89.23**	60.54	82.85	67.78	77.49	88.33
Reference Match	69.68	**89.44**	39.32	58.97	0.00	55.95	49.27
Click Order	0.00	0.00	0.00	0.00	0.00	**11.46**	10.62
OCR Gradient	85.63	85.63	65.15	**89.33**	71.41	82.54	84.36
Average	63.03	68.50	41.68	62.37	50.32	64.52	**70.63**

covering Claude-4-Sonnet (Anthropic, 2025), UI-TARS-72B-DPO (Qin et al., 2025), and our Ovis2-Agent-CAPTCHA-8B model.

5.2 WPR and CD

Table 2 reports the WPR for all models. Due to inherent limitations of the browser-use execution framework (Müller and Žunič, 2024), such agents cannot perform operations like "Drag Drop", "Rotation match", "Slide Puzzle", "Semantic Area", "Click Order" or "Slide Line". Specifically, the SOM mechanism treats an image as a single object and cannot localize or manipulate elements within it. By contrast, computer-use agents directly simulate mouse interactions, enabling them to attempt all CAPTCHA categories.

Weighted Pass Rate. Ovis2-Agent-CAPTCHA-8B achieves the highest average WPR. Notably, on the high-traffic "Patch Select" category, it surpasses Gemini by 30.66 percentage points. For Claude-4-Sonnet (computer-use), WPR drops to zero in some categories, largely due to sensitivity to image resolution, which prevents reliable coordinate output for click actions. For Claude-4-Sonnet (browser-use), failures in "Image Recognition" and "Patch Select" stem from repeated memory read/write loops, reflecting a fundamental vulnerability of the framework.

Completion Degree. Table 3 shows that Ovis2-Agent-CAPTCHA-8B also achieves the best average CD. Interestingly, in categories such as "Patch Select", the WPR is relatively low across all models, while CD remains high, indicating that models select most patches correctly but often fail to complete the task perfectly. This suggests that higher image resolution, more query steps, or finer-grained behavioral optimization could close the gap between CD and WPR.

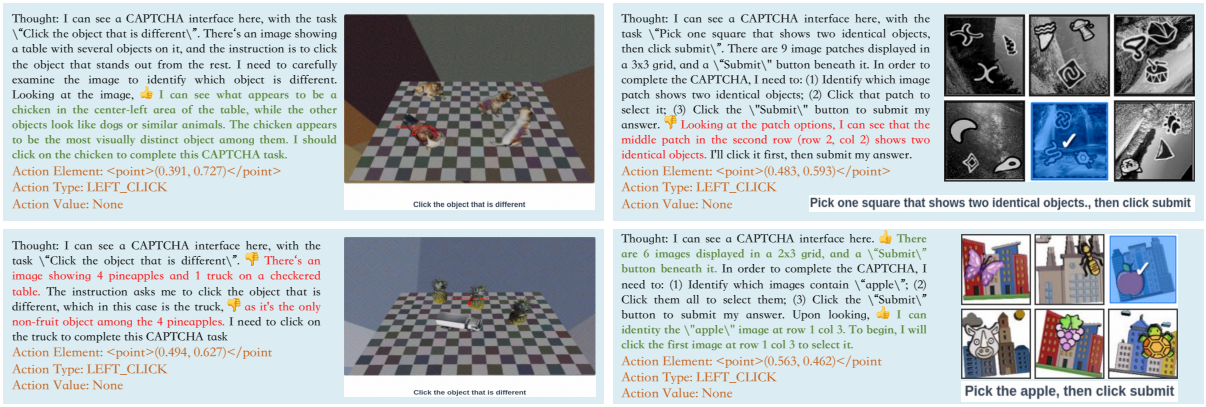

Figure 6: Correct case (left: Click the object that is different) vs. Error cases (right: Semantic Select).

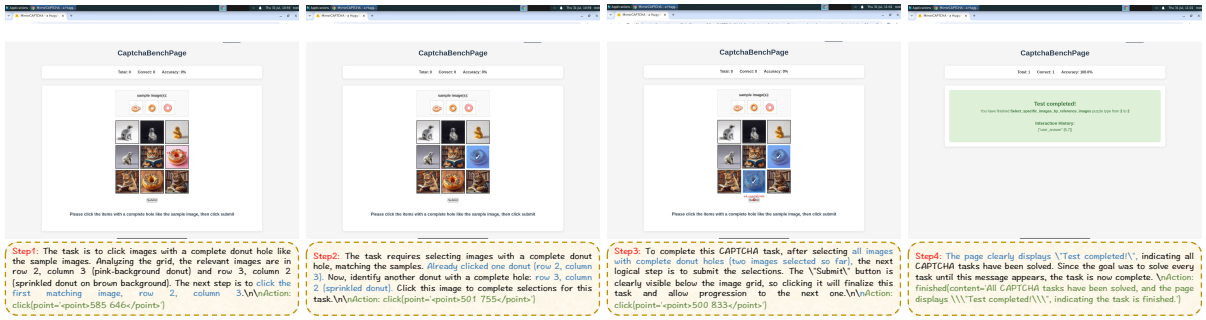

Figure 7: CAPTCHA puzzle: Click the items with a complete hole, as shown in the sample images.

5.3 Success and Failure Case Analysis

Figure 6 presents typical success and failure cases. For "Click the object that is different" case, the model identifies a chicken among dog-like animals and clicks the correct location. In the error case, however, it misclassifies the objects as 4 pineapples and 1 truck. Although it correctly identifies the truck as the outlier, the click misses the precise coordinates, causing task failure. Similarly, failure arises from difficulty in interpreting abstract icons. Here, the model guesses randomly rather than reasoning about semantic differences.

These cases highlight two key limitations: **(i)** imprecise visual recognition of object attributes and counts, and **(ii)** difficulty in extracting discriminative features from abstract images. Addressing these issues will require improved visual feature extraction and more reliable object classification.

5.4 Inference Process on CAPTCHA puzzle

Figures 7 provide a qualitative look into how web agents solve challenging CAPTCHA on the MirrorCAPTCHA benchmark. First, it accurately interprets the natural language instructions provided by the puzzle("items with a complete hole"). The agent then applies its visual reasoning skills to identify the correct target elements on the webpage. Finally, it translates this understanding into a series of precise interactive operations, including clicks, drags, and text entry, to complete the CAPTCHA task. This seamless integration of perception, reasoning, and execution highlights the agent's advanced capabilities. *More detailed reasoning examples are provided in Appendix F.*

6 Conclusion

In this paper, we introduce **MirrorCAPTCHA**, a benchmark designed to serve as a "mirror" of real-world CAPTCHA distributions. MirrorCAPTCHA filters 2,095 valid websites with deployed CAPTCHAs from Common Crawl, categorized into 18 types spanning both deep learning and web interaction tasks. To approximate real-world encounter frequencies, MirrorCAPTCHA employs random walks and evaluates performance using two metrics: *Weighted Pass Rate* and *Completion Rate*. In addition, we introduce `Ovis2-Agent-CAPTCHA-8B`, a model trained on a synthesized CAPTCHA dataset. Experimental results show that it significantly outperforms both open-source and closed-source counterparts, surpassing Gemini-2.5-Pro by 9.4% in Weighted Pass Rate and achieves the highest Completion Degree across most CAPTCHA categories.

7 Limitations

MirrorCAPTCHA primarily targets interactive CAPTCHAs dominated by vision and text. Consequently, it does not currently include less common but increasingly relevant multimodal CAPTCHA formats, such as audio CAPTCHAs for accessibility, video CAPTCHAs, or hybrid modality CAPTCHAs with visual interaction. In future work, we plan to extend our data collection and evaluation protocols to incorporate audio- and video-based CAPTCHA, enabling a more comprehensive assessment of truly multimodal web agents.

8 Ethical Considerations and Risk Mitigation

MirrorCAPTCHA is intended to evaluate the practical bottlenecks that web agents face when interacting with CAPTCHA-protected websites, rather than to provide a deployable system for bypassing CAPTCHA protections. Existing web-agent benchmarks, such as VisualWebArena and AgentBench, typically omit CAPTCHA challenges, leaving an important gap in evaluating real-world agent robustness and usability. MirrorCAPTCHA is designed to fill this gap.

Nevertheless, we acknowledge that CAPTCHA-related research is inherently dual-use. The benchmark, synthesis pipeline, trained models, and CAPTCHA distribution analysis could be misused to facilitate malicious automation, including mass account registration, credential stuffing, scraping of restricted content, and other unauthorized access behaviors. Such misuse could undermine website security, disrupt online services, and threaten user privacy.

To reduce these risks, we adopt a minimization and compliance principle: we collect only the information necessary for CAPTCHA analysis, anonymize screenshots and logs, and avoid storing cookies, account identifiers, or other unnecessary sensitive data. In addition, we implement a tiered release plan. The paper, evaluation metrics, and framework code will be public; the benchmark dataset and synthesis pipeline will be shared only under controlled access with a Data Use Agreement (DUA); and the Ovis2-Agent-CAPTCHA-8B model weights and full 370k training dataset will not be publicly released. The DUA will explicitly prohibit commercial CAPTCHA-solving, unauthorized automated access, and verification-bypassing uses. We will also clearly state these restrictions in both the paper and the repository.

Furthermore, our training data is based on synthetic CAPTCHAs rather than CAPTCHAs collected from real websites, which reduces direct impact on deployed services. If we identify serious weaknesses in specific CAPTCHA systems, we will consider responsible disclosure to the relevant vendors.

More broadly, we note that CAPTCHA security has historically evolved through an attacker–defender cycle: prior advances in solving early text-based CAPTCHAs helped motivate stronger mechanisms such as reCAPTCHA v2/v3. In the same spirit, our findings may help defenders identify vulnerable CAPTCHA types and inform stronger future verification methods. We therefore view MirrorCAPTCHA as a benchmark for research evaluation and defensive insight, not as a tool for operational bypass.

References

Saaket Agashe, Jiuzhou Han, Shuyu Gan, Jiachen Yang, Ang Li, and Xin Eric Wang. 2025. Agent s: An open agentic framework that uses computers like a human. In *ICLR*.

Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, and 1 others. 2023. Gemini: A family of highly capable multimodal models. In *arXiv*.

Anthropic. 2025. Claude api. Anthropic's Conversational AI API.

Zhe Chen, Weiyun Wang, Yue Cao, Yangzhou Liu, Zhangwei Gao, Erfei Cui, and 1 others. 2024. Expanding performance boundaries of open-source multimodal models with model, data, and test-time scaling. In *arXiv*.

Common Crawl. 2007. Common crawl. Common Crawl corpus contains petabytes of data, regularly collected since 2008.

Zaid Derea, Beiji Zou, Amal A. Al-Shargabi, Alaa Thobhani, and Amr Abdussalam. 2023. Deep learning based CAPTCHA recognition network with grouping strategy. *Sensors*, 23(23):9487.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, and 1 others. 2024. The llama 3 herd of models. In *arXiv*.

Lutfi Eren Erdogan, Nicholas Lee, Sehoon Kim, Suhong Moon, Hiroki Furuta, Gopala Anumanchipalli, Kurt Keutzer, and Amir Gholami. 2025. Plan-and-act: Improving planning of agents for long-horizon tasks. In *arXiv*.

Izzeddin Gur, Hiroki Furuta, Austin V. Huang, Mustafa Safdari, Yutaka Matsuo, Douglas Eck, and Aleksandra Faust. 2024. A real-world webagent with planning, long context understanding, and program synthesis. In *ICLR*.

Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. 2024. Webvoyager: Building an end-to-end web agent with large multimodal models. In *ACL*, pages 6864–6890.

Yu Hu, Li Chen, and Jun Cheng. 2018. A captcha recognition technology based on deep learning. In *ICIEA*, pages 617–620.

Faria Huq, Zora Zhiruo Wang, Frank F. Xu, Tianyue Ou, Shuyan Zhou, Jeffrey P. Bigham, and Graham Neubig. 2025. Cowpilot: A framework for autonomous and human-agent collaborative web navigation. In *arXiv*.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Russ Salakhutdinov, and Daniel Fried. 2024. Visualwebarena: Evaluating multimodal agents on realistic visual web tasks. In *ACL*, pages 881–905.

Hanyu Lai, Xiao Liu, Iat Long Iong, Shuntian Yao, Yuxuan Chen, Pengbo Shen, Hao Yu, Hanchen Zhang, Xiaohan Zhang, Yuxiao Dong, and Jie Tang. 2024. Autowebglm: A large language model-based web navigating agent. In *KDD*, pages 5295–5306.

Ido Levy, Ben Wiesel, Sami Marreed, Alon Oved, Avi Yaeli, and Segev Shlomov. 2025. St-webagentbench: A benchmark for evaluating safety and trustworthiness in web agents. In *arXiv*.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, and 1 others. 2024. Agentbench: Evaluating llms as agents. In *ICLR*.

Shiyin Lu, Yang Li, Qing-Guo Chen, Zhao Xu, Weihua Luo, Kaifu Zhang, and Han-Jia Ye. 2024. Ovis: Structural embedding alignment for multimodal large language model. In *arXiv*.

Yaxin Luo, Zhaoyi Li, Jiacheng Liu, Jiacheng Cui, Xiaohan Zhao, and Zhiqiang Shen. 2025. Open captchaworld: A comprehensive web-based platform for testing and benchmarking multimodal LLM agents. In *NeurIPS*.

Magnus Müller and Gregor Žunič. 2024. Browser use: Enable ai to control your browser.

OpenAI. 2025. Openai api. Accessed: 2023-12-01.

Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, and 1 others. 2025. Ui-tars: Pioneering automated GUI interaction with native agents. In *arXiv*.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, and 1 others. 2021. Learning transferable visual models from natural language supervision. In *ICML*, volume 139, pages 8748–8763.

Chenyang Shao, Xinyuan Hu, Yutang Lin, and Fengli Xu. 2025. Division-of-thoughts: Harnessing hybrid language model synergy for efficient on-device agents. In *WWW*, pages 1822–1833.

Yujin Shu and Yongjin Xu. 2019. End-to-end captcha recognition using deep cnn-rnn network. In *IMCEC*, pages 54–58.

Significant Gravitas. 2023. AutoGPT.

Zirui Song, Yaohang Li, Meng Fang, Zhenhao Chen, Zecheng Shi, Yuan Huang, and Ling Chen. 2024. Mmac-copilot: Multi-modal agent collaboration operating system copilot. In *arXiv*.

Shengyuan Tang. 2024. Research on captcha recognition technology based on deep learning. *Applied and Computational Engineering*, 81:41–46.

Alaa Thobhani, Mingsheng Gao, Ammar Hawbani, Safwan Taher Mohammed Ali, and Amr Abdussalam. 2020. Captcha recognition using deep learning with attached binary images. *Electronics*, 9(9).

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, and 1 others. 2024. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. In *arXiv*.

Zonglin Wu, Yule Xue, Xin Wei, and Yiren Song. 2025. Mca-bench: A multimodal benchmark for evaluating CAPTCHA robustness against vlm-based attacks. In *arXiv*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, and 1 others. 2025. Qwen3 technical report. In *arXiv*.

Zhengyuan Yang, Linjie Li, Kevin Lin, Jianfeng Wang, Chung-Ching Lin, Zicheng Liu, and Lijuan Wang. 2023. The dawn of lmms: Preliminary explorations with gpt-4v(ision). In *arXiv*.

Guixin Ye, Zhanyong Tang, Dingyi Fang, Zhanxing Zhu, Yansong Feng, Pengfei Xu, Xiaojiang Chen, Jungong Han, and Zheng Wang. 2020. Using generative adversarial networks to break and protect text captchas. *ACM Transactions on Privacy and Security*, 23.

A Comparison with Existing Benchmarks

Table 4 presents a detailed comparison of the three most recent CAPTCHA benchmarks. Both Open CaptchaWorld (Luo et al., 2025) and MCA-Bench (Wu et al., 2025) rely on a random distribution of CAPTCHA types, which fails to accurately reflect the real-world frequencies of these

Table 4: Comparison with Open CaptchaWorld and MCA-Bench benchmarks.

Dataset	CAPTCHA Distribution	Number of Categories	Data size	CAPTCHA Categories
Open CaptchaWorld	Random	20	225	Select Animal, Pick Area, Patch Select, Object Match, Misleading Click, Geometry Click, Image Recognition, Coordinates, Place Dot, Rotation Match, Image Matching, Connect Icon, Bingo, Dart Count, Dice Count, Slide Puzzle, Path Finder, Click Order, Unusual Detection, Hold Button
MCA-Bench	Random	20	4,000	3×3 Grid Select, 3×3 Jig-swap, Arithmetic Char, Arithmetic Select, Hollow Pattern, Distort Word, Classic Char, Sequential Letter, Bright Dist, Sliding Block, Align Sliders, Rotate Block, Geometry Shape, Rotation Letter, Color Discrimination, Vowel Select, Full-img Grid Select, Text-based Arithmetic, Common Sense, Invert Letter
MirrorCAPTCHA	Real-world	18	1,000	Image recognition, Patch Select, OCR Grey, OCR Grid, Drag Drop, OCR Color-pure, Image Match, Rotation Match, Slide Puzzle, OCR Color-pattern, Semantic Select, Semantic Area, OCR Dot, Reference Match, CLICK Order, Slide Line, Type Arithmetic, OCR Gradient

challenges. Open CaptchaWorld includes 20 categories but has an extremely small dataset of just 225 samples, which can lead to high randomness in evaluation. MCA-Bench is larger, with $4,000$ samples across 20 categories, but its synthetic puzzles do not capture the diversity and complexity found on real websites. In contrast, our MirrorCAPTCHA benchmark is grounded in real-world data. It features 18 CAPTCHA types collected from $2,095$ live websites, and its distribution is statistically derived from actual web traffic. With a total of $1,000$ puzzles, MirrorCAPTCHA provides a more realistic and reliable tool for evaluating web agents, making it a "mirror" that truly reflects an agent's performance in real-world scenarios.

B Detail on Puzzle Synthesis in Section 4

As shown in Figure 5, the synthesis pipeline uses three pools and structured templates: 1) Visual Pool: Open-source images (objects, scenes, animals, vehicles, etc.) used to build the visual content of CAPTCHA. 2) Char Pool: Character rendering templates with different fonts, colors, deformation patterns, and noise styles, used to generate OCR chars. 3) Prompt Pool: Natural-language instruction templates used to generate task descriptions.

For each CAPTCHA type, we design a structured template and randomly sample key factors, such as layout (*e.g.*, 3×3 or 4×4 grids), distractors (noise lines, background textures), and the rule for producing the ground-truth answer. The pipeline then samples from the three pools and assembles them based on the template to generate CAPTCHA images and annotations (including the correct answer and click coordinates). We also randomize parameters (font size, color, noise level, image content, etc.) to ensure diversity.

The synthetic tasks are designed to teach models general skills (visual grounding, grid selection, text recognition, and instruction-following interaction), not to memorize benchmark instances. We do not use real CAPTCHA instances or website screenshots for training, and we do not reuse any benchmark images, website styles, or instances during synthesis. Therefore, the synthetic data differs from MirrorCAPTCHA in distribution. MirrorCAPTCHA is mainly collected from real websites and contains more long-tail variations and interaction noise. Synthetic training is only used for capability pretraining/alignment. Evaluation is done in real web-interaction settings, so strong performance does not indicate memorization.

C Sensitivity analysis of the WPR weighting mechanism.

We conducted a comprehensive sensitivity analysis of the weighting mechanism used in WPR. Specifically, we evaluated several reasonable alternative weighting schemes and parameter settings, including different random-walk lengths, a restart-based variant analogous to PageRank (with a restart probability of 0.15), and different subgraph sizes. Table 5 reports the resulting CAPTCHA-type weights, and Table 6 shows the recomputed WPR scores for representative models under these variants.

Overall, the weighting distribution is highly stable across all settings. The same high-traffic CAPTCHA categories, such as *Image Recognition* and *Patch Select*, consistently receive the largest weights, while long-tail categories remain collectively small. As a result, recomputing WPR under these alternative weighting variants leads to only negligible numerical changes in model scores, and the relative ranking of all evaluated models remains unchanged. These results indicate that our conclusions are robust to reasonable variations in the weighting mechanism.

Table 5: Sensitivity analysis of CAPTCHA-type weights under different WPR weighting variants. The default setting uses a 15k-node subgraph and 10^8 random-walk steps. Variant A and B change the number of walk steps; Variant C adds a restart probability of 0.15; Variant D and E use different subgraph sizes. The overall heavy-tailed structure remains highly stable across all variants.

CAPTCHA Type	**Default (15k, 10^8)**	**Variant A (10^7)**	**Variant B (10^9)**	**Variant C (restart=0.15)**	**Variant D (Top 10k)**	**Variant E (Top 20k)**
Image Recognition	0.3587	0.3585	0.3587	0.3412	0.3650	0.3510
Patch Select	0.3485	0.3487	0.3485	0.3350	0.3520	0.3450
OCR Grey	0.0918	0.0917	0.0918	0.0980	0.0890	0.0950
OCR Grid	0.0359	0.0358	0.0359	0.0410	0.0335	0.0380
Drag Drop	0.0293	0.0294	0.0293	0.0320	0.0280	0.0305
OCR ColorPure	0.0281	0.0280	0.0281	0.0315	0.0275	0.0290
Image Match	0.0219	0.0220	0.0219	0.0245	0.0205	0.0230
Rotation Match	0.0209	0.0208	0.0209	0.0232	0.0195	0.0215
Long-tail types (sum)	0.0649	0.0651	0.0649	0.0731	0.0650	0.0670

Table 6: WPR of representative models under different weighting variants. Although the precise values vary slightly, the relative ranking of models remains unchanged across all settings.

Model	**Default**	**Var A**	**Var B**	**Var C**	**Var D**	**Var E**
Ovis2-Agent-8B	49.57%	49.61%	49.56%	49.82%	49.31%	49.74%
Gemini-2.5-Pro	40.22%	40.18%	40.22%	39.95%	40.50%	40.11%
o4-mini	33.66%	33.70%	33.66%	33.41%	33.82%	33.58%
GPT-4.1	28.58%	28.52%	28.58%	28.70%	28.41%	28.66%

D Detailed Completion Degree Metric

To capture partial correctness in CAPTCHA-solving tasks, we define `Completion Degree` (CD) as a family of fine-grained metrics that quantify *how close* a web agent's output is to the correct answer, even when the CAPTCHA is not fully solved.

We adopt four types of CD metrics: `F1 Score`, `Levenshtein Distance`, `Sequence Matching Accuracy`, and `Angle Distance Error`, each aligned with the nature of specific CAPTCHA categories.

▶ `F1 Score`

This metric is used for puzzles that require selecting one or more items from a set, such as:

- Image Recognition
- Patch Select
- Semantic Select
- Reference Match

Definition. The `F1 score` is the harmonic mean of precision and recall:

$$\mathrm{F}_1 = \frac{2 \times \mathrm{Precision} \times \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}},$$

where

$$\mathrm{Precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}, \quad \mathrm{Recall} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}},$$

where TP (True Positive) denotes correctly selected items, FP (False Positive) denotes wrongly selected items, and FN (False Negative) denotes missed correct items.

Explanation: These CAPTCHA types may contain multiple correct elements. F1 score balances correctness and completeness: selecting wrong patches lowers precision, while missing correct ones lowers recall. F1 scores are normalized to $[0, 1]$.

▶ `Levenshtein Distance`

This metric is used for CAPTCHA that involve text recognition:

- OCR Grey
- OCR Gradient
- OCR Grid
- OCR ColorPure
- OCR ColorPattern
- OCR Dot

Definition. The `Levenshtein distance` measures the minimum number of single-character edits (insertions, deletions, substitutions) needed to transform the predicted string s_1 into the ground truth s_2:

$$\mathrm{Lev}(s_1, s_2) \in \mathbb{N}^+.$$

We convert this distance into a similarity score:

$$\text{CD} = 1 - \frac{\text{Lev}(s_1, s_2)}{\max(|s_1|, |s_2|)}.$$

A score of 1 indicates a perfect match, while a score of 0 indicates completely different strings.

Explanation. This metric captures OCR-specific errors (*e.g.*, character substitutions) and awards partial credit when most of the string is correct.

▶ `Sequence Matching Accuracy`

This metric is specifically designed for:

- Click Order

Definition. Let P be the sequence of elements clicked by the agent and G the ground truth sequence. We compute:

$$\text{CD} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}\{P_i = G_i\},$$

where n is the sequence length, P_i is the i-th clicked element, and $\mathbf{1}\{\cdot\}$ is the indicator function.

Explanation. This metric measures position-wise accuracy. For example, if the CAPTCHA requires clicking "Cat → Dog → Bird", this metric counts how many items are in the correct position. This allows for partial credit even if the model only gets part of the sequence right.

▶ `Angle Distance Error`

This metric is used for puzzles that require rotation:

- Rotation Match

Definition. Given predicted rotation θ_p and ground truth θ_g, we compute:

$$\Delta\theta = \min\left(|\theta_p - \theta_g|, 360 - |\theta_p - \theta_g|\right),$$

which correctly accounts for circular periodicity (*e.g.*, $0^\circ \equiv 360^\circ$). The completion score is defined as:

$$\text{CD} = 1 - \frac{\Delta\theta}{\theta_{\max}},$$

where $\theta_{\max}$ is the maximum possible deviation (180°). A CD of 1 means the rotation is perfectly aligned, 0.5 means a misalignment of 90°, and 0 means an opposite alignment (180° difference).

Explanation. This metric gives proportionate credit for predictions that are close to the correct angle, which more accurately reflects an agent's capability in near-solved cases compared to a binary pass/fail judgment.

E Detailed Visit Probability of Websites

F Additional reasoning examples

We provide further examples of CAPTCHA-solving processes to illustrate how the web agent interacts with real-world webpages. These cases highlight the agent's ability to interpret and reason over CAPTCHA puzzles, make decisions, and execute corresponding interactive actions.

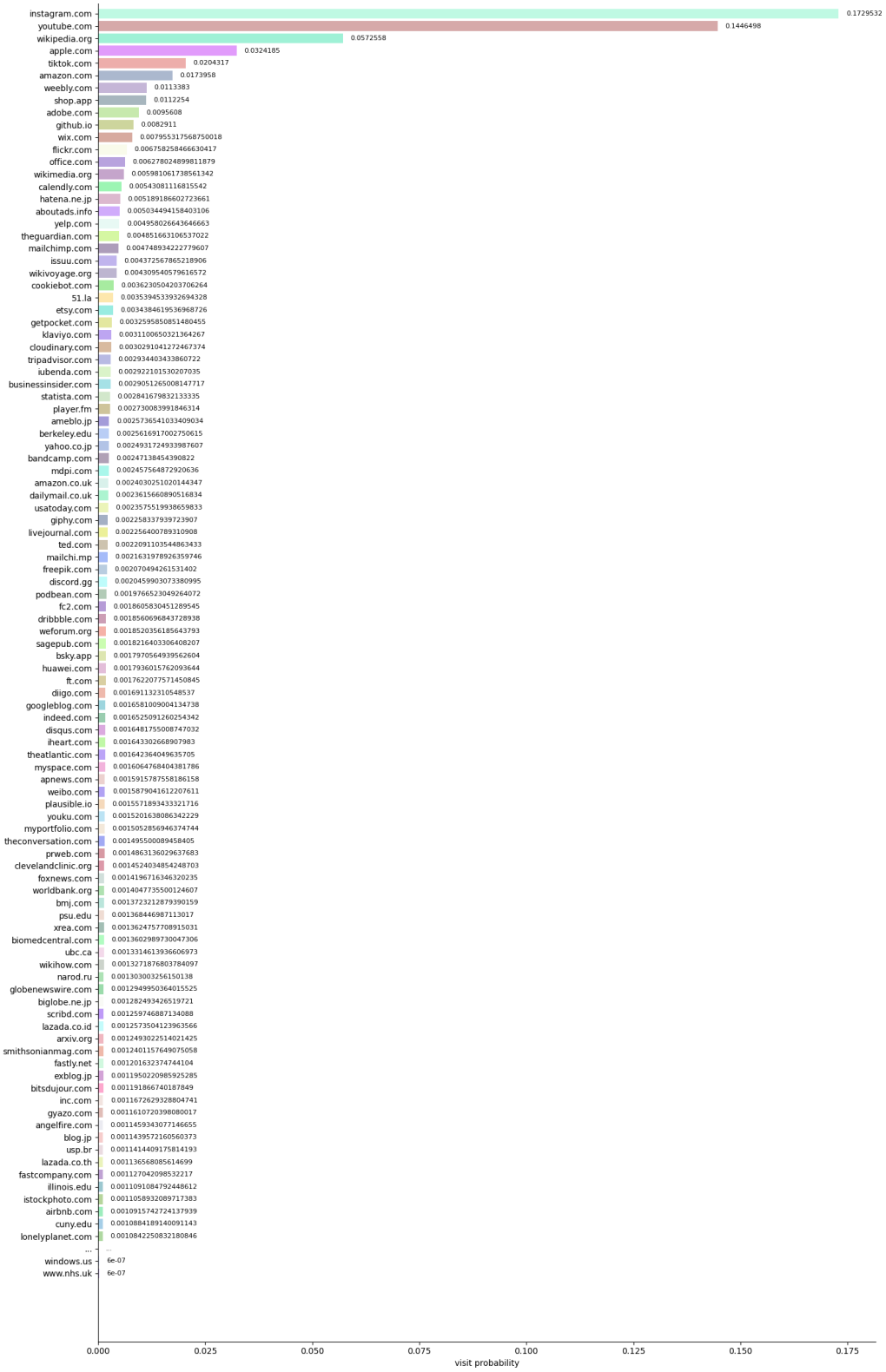

Figure 8: Detailed visit frequencies distribution of the 2095 websites with deployed CAPTCHAs. The top 10 websites account for nearly half of the total traffic, reflecting real-world internet patterns.

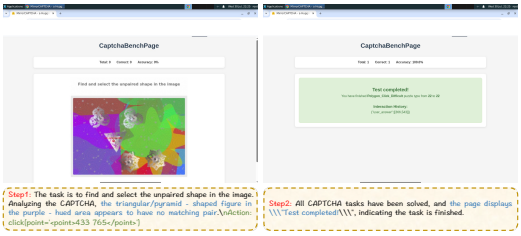

Figure 9: CAPTCHA puzzle: Find and select the unpaired shape in the image.

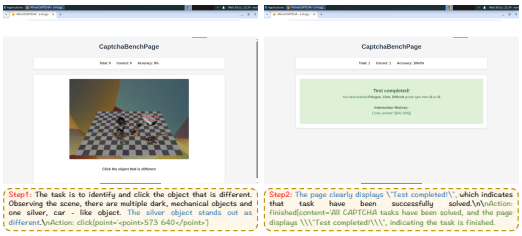

Figure 10: CAPTCHA puzzle: Click the object that is different.

Figure 11: CAPTCHA puzzle: Select all squares with taxis.

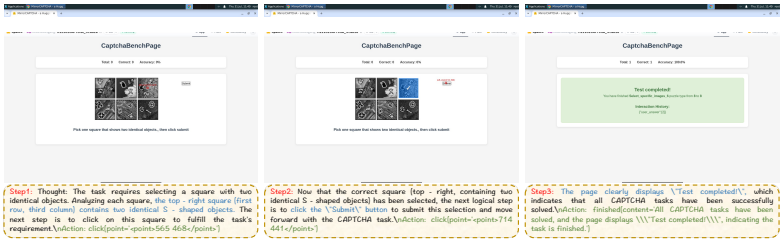

Figure 12: CAPTCHA puzzle: Pick one square showing two identical objects, then click submit.

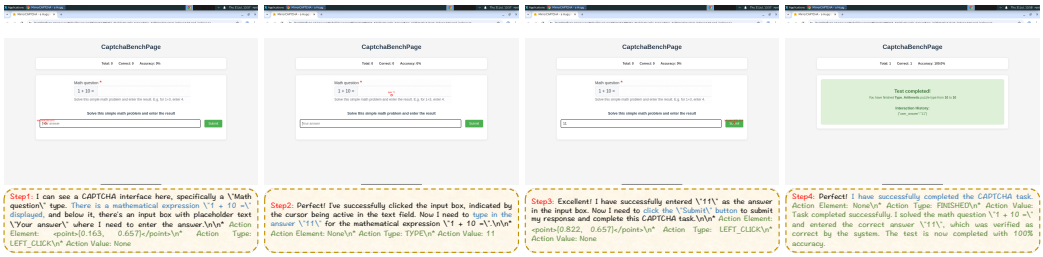

Figure 13: CAPTCHA puzzle: Solve the math problem and enter the result.

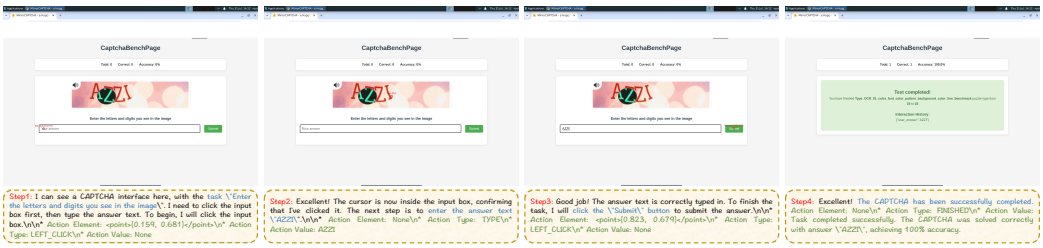

Figure 14: CAPTCHA puzzle: Enter the letters and digits shown in the image.

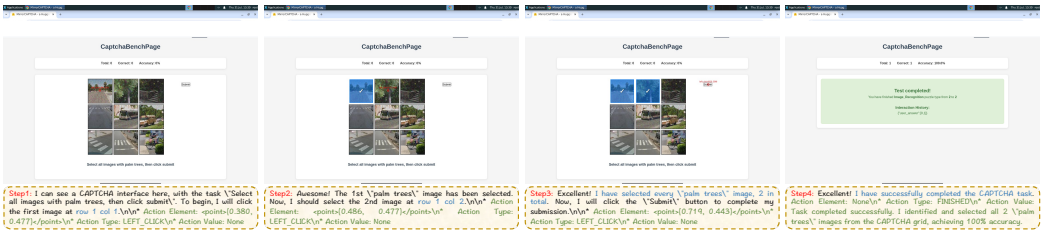

Figure 15: CAPTCHA puzzle: Select all images with palm trees, then click submit.

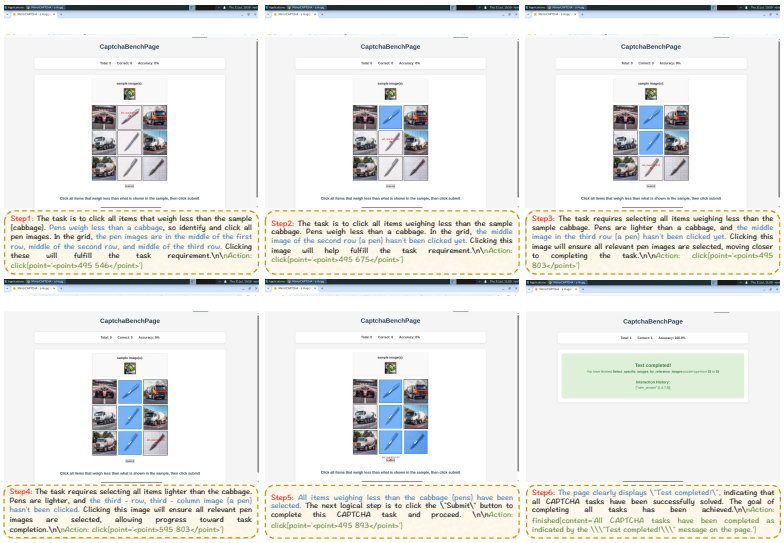

Figure 16: CAPTCHA puzzle: Click all items that weigh less than the sample.

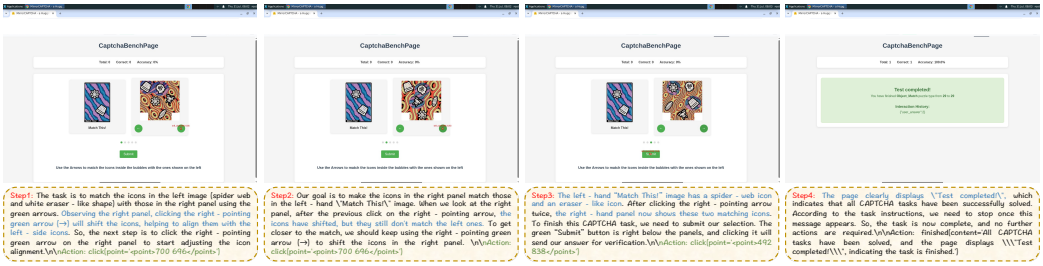

Figure 17: CAPTCHA puzzle: Use Arrows to match the icons inside the bubbles with those on the left.

Figure 18: CAPTCHA puzzle: Click the icons in order shown in the reference images.

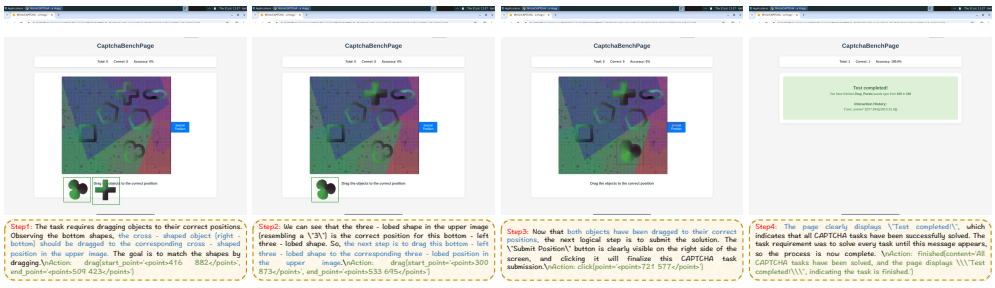

Figure 19: CAPTCHA puzzle: Drag the objects to their correct position.