CAPTCHAARENA: A LARGE-SCALE, FINE-GRAINED DATASET FOR TRAINING COMPUTER-USE AGENTS ON INTERACTIVE CAPTCHAS

Anonymous authors
Paper under double-blind review

ABSTRACT

Interactive CAPTCHAs remain challenging for computer-use agents, while existing datasets face trade-offs among type coverage, interaction fidelity, and trajectory supervision. To address these gaps, we present **CaptchaArena**, the first large-scale, fine-grained training dataset for interactive CAPTCHA solving. It contains 50K puzzles across 20 CAPTCHA types and 5 interaction modes, with every solution verified through execution. **CaptchaArena** provides 50K screenshot-action trajectories, including 46K with step-by-step reasoning annotations. It also includes fine-grained pixel-mask annotations for irregular targets. Using **CaptchaArena**, we train **CaptchaAgent**, a single 9B policy for all 20 CAPTCHA types, with supervised fine-tuning followed by reinforcement learning. The environment verifier directly provides the RL reward. Supervised fine-tuning reaches 70.5 Pass@1, and reinforcement learning further improves it to 71.7, while also improving performance on two external benchmarks. These results demonstrate the value of large-scale, fine-grained computer-use supervision for training interactive CAPTCHA agents.

1 INTRODUCTION

Interactive CAPTCHAs expose a critical gap in current computer-use agents. These agents can now perform long-horizon tasks using screenshots and low-level mouse and keyboard actions (Xie et al., 2024). However, a single failed CAPTCHA can stop an entire workflow and block everything downstream. Existing benchmarks often avoid measuring this, and each treats CAPTCHAs differently. Online-Mind2Web (Xue et al., 2025) removes CAPTCHA-protected websites during construction. VisualWebArena (Koh et al., 2024) self-hosts every site. Yet CAPTCHAs are worth evaluating because solving them requires the core loop of computer use. An agent must read the web page, decide what to do, act on the right pixels, and verify the resulting state change.

Training data for CAPTCHA interaction remain limited. Existing CAPTCHA resources involve trade-offs among type coverage, interaction fidelity, and trajectory supervision. ReCAP (Chen et al., 2026) and CaptchaMind (Wang et al., 2026) provide step-by-step interaction trajectories with reasoning, but cover 7 and 8 challenge types. MirrorCAPTCHA (Wu et al., 2026a) evaluates 18 types, but its 370K synthetic training samples provide only answer and click-coordinate supervision. Open CaptchaWorld (Luo et al., 2025) covers 20 types but evaluates agents through the browser-use framework. This framework cannot handle some CAPTCHA types that require fine-grained interaction. For example, on MirrorCAPTCHA, every browser-use agent scores 0 on 6 of its 18 types (Wu et al., 2026a). MCA-Bench (Wu et al., 2026b) predicts structured solutions and action traces rather than executing them step-by-step in a live browser environment. Its predicted clicks are graded against bounding boxes, while CAPTCHA-X (Song et al., 2025) uses manually defined geometric acceptance regions. Such region-based grading can over-accept areas outside irregular targets.

We introduce **CaptchaArena** to close these gaps (Fig. 1). It is a large-scale, fine-grained computer-use training dataset containing 50K interactive puzzles across 20 types and 5 interaction modes, including single-click, multi-click, arrow-cycle, real-time, and text-entry. Puzzle quality is manually inspected. For irregular targets, clicks are graded against pixel-masks. Each puzzle has an executable reference solution that is run in a real browser and must pass the environment verifier. The same

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

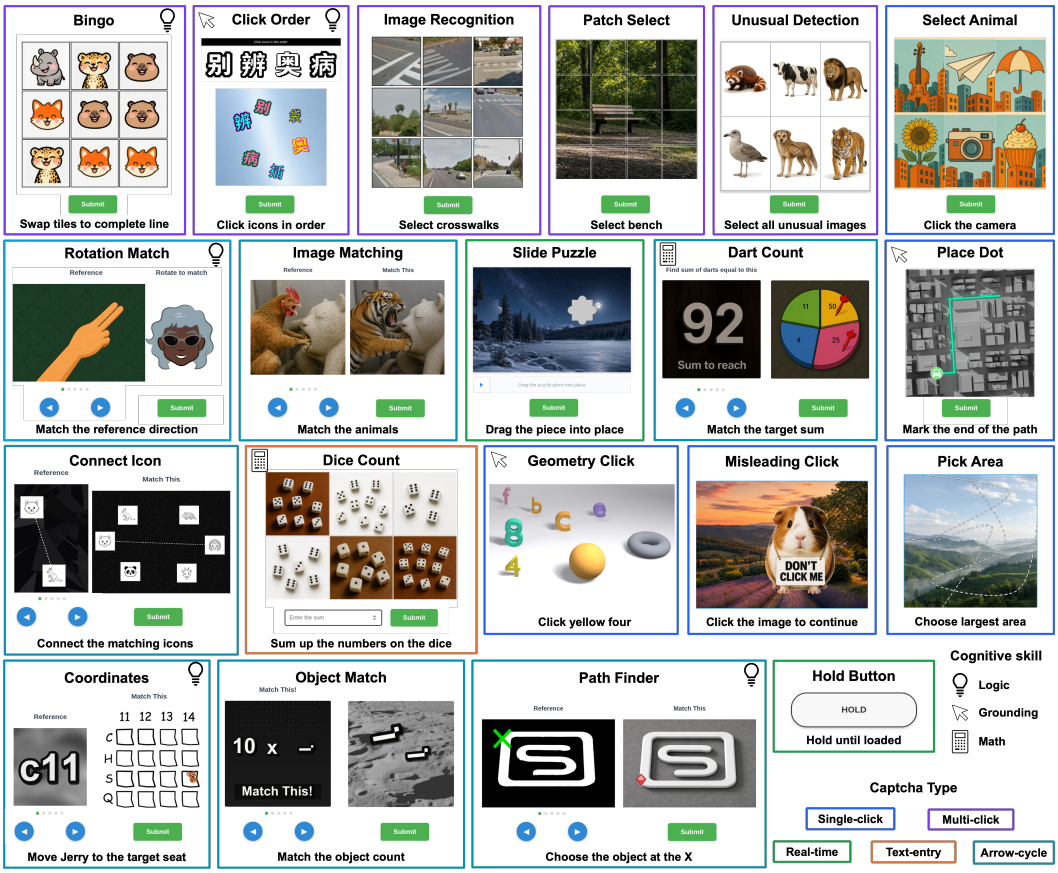


Figure 1: Representative puzzles from the 20 CAPTCHA types in **CaptchaArena**, cropped for display. Border colors mark interaction modes and corner icons mark cognitive skills.

execution produces one screenshot-action trajectory per puzzle. A teacher model adds step-by-step reasoning-annotations and a separate judge checks each annotation, producing 46K reasoning-annotated trajectories.

We release **CaptchaAgent**, a single policy built on Qwen3.5-9B (Qwen Team, 2026). It is fine-tuned on a subset of the reasoning-annotated trajectories and then further trained with reinforcement learning. The same environment verifier used to validate puzzle solutions directly provides the RL reward. Supervised fine-tuning raises Pass@1 from 11.4 for the untrained backbone to 70.5. Reinforcement learning further raises Pass@1 to 71.7, and improves the score on two external benchmarks (App. I). Humans reach 94.1 on the same 4,000 puzzles through the same interface.

Contributions.

- **CaptchaArena**. A large-scale, fine-grained computer-use training dataset containing 50K interactive CAPTCHA puzzles across 20 types and 5 interaction modes. Every solution is verified through execution, and the dataset includes 46K reasoning-annotated trajectories and pixel-mask supervision.
- **CaptchaAgent**. A single 9B policy covering all 20 types, fine-tuned on a subset of the reasoning-annotated trajectories and then trained with reinforcement learning. The environment verifier provides the RL reward.
- **Error analysis**. A per-type analysis of remaining failures, comparing **CaptchaAgent** with the untrained backbone and human performance on the same puzzles.

Table 1: Comparison of CAPTCHA benchmarks and training datasets, as reported by each paper.

Benchmark	Types	CU	Traj.	Reasoning	RL reward	Click Eval.	Scale
Evaluation-only benchmarks							
Open CaptchaWorld	20	✗	✗	✗	–	–	463†
Halligan	26	✓	✗	✗	–	tol.	2,600
Oedipus	4	✗	✗	✗	–	–	400
HLL	10	✓	✗	✗	–	tol.	–
CAPTCHA-X	7	✓	✗	✓	–	region	1,839
CAPTURE	25	✗	✗	✗	–	–	61,464
Training datasets							
MCA-Bench	20	✗	✗	✗	✗	box	180K
MirrorCAPTCHA	18	✓	✗	✗	✗	–	370K
ReCAP	7	✓	✓	✓	✗	–	160K
CaptchaMind	8	✓	✓	✓	✓	box	16K
CaptchaArena (ours)	**20**	✓	✓	✓	✓	**tol.+mask**	**42K**

Scale denotes evaluation size for evaluation-only benchmarks and training size for training datasets. *CU* denotes direct low-level computer use rather than structured prediction or browser-use/SoM (Yang et al., 2023; Müller & Žunič, 2024). *Traj.* denotes multi-step screenshot-action trajectories with intermediate observations. Halligan and HLL report positional tolerance (*tol.*) without specifying its geometry or threshold. A dash denotes not applicable or not reported. † Later release; the paper reports 225.

2 Related Work

Type coverage. Existing CAPTCHA benchmarks vary widely in type coverage (Table 1). ReCAP (Chen et al., 2026) and CaptchaMind (Wang et al., 2026) provide training data for 7 and 8 CAPTCHA types. MCA-Bench (Wu et al., 2026b) offers over 180K training data for 20 types, but trains task-specific LoRA adapters rather than a unified policy. MirrorCAPTCHA (Wu et al., 2026a) evaluates 1,000 puzzles across 18 types, including 6 OCR variants. However, its 370K synthetic training samples provide only answer and click-coordinate supervision. Open CaptchaWorld (Luo et al., 2025) covers 20 types. Its current release contains 463 puzzles, versus 225 in the original paper. We adopt its task types. CAPTURE (Zhang et al., 2025) and Halligan (Teoh et al., 2025) evaluate 25 and 26 CAPTCHA types. Oedipus (Deng et al., 2025) evaluates only 4 CAPTCHA types.

Interaction fidelity. Computer-use evaluation places an agent in a closed-loop perception-action process (Zhou et al., 2024; Xie et al., 2024; Xue et al., 2025). However, mainstream benchmarks generally do not evaluate CAPTCHA solving. Akkil et al. (2026) treat CAPTCHAs as external constraints during live evaluation, while HLL (Song et al., 2026) identifies CAPTCHA verification as a deployment bottleneck and finds frontier agents remain brittle. CAPTCHA-specific benchmarks also differ in their interaction interfaces. Open CaptchaWorld evaluates agents with the browser-use framework, whose actions often target indexed web elements rather than raw screen coordinates. MCA-Bench (Wu et al., 2026b) produces structured outputs rather than step-by-step execution in a live browser.

Trajectory supervision. Existing resources also differ in the supervision they provide. CAPTCHA-X (Song et al., 2025) provides reasoning and action annotations for real-world CAPTCHA puzzles. CaptchaMind (Wang et al., 2026) provides process-level annotations across 8 types and trains a unified policy with SFT followed by GRPO. ReCAP (Chen et al., 2026) collects large-scale reasoning-action and self-correction trajectories across 7 types. Although ReCAP improves zero-shot transfer overall, its trained models still underperform their corresponding base models on some unseen CAPTCHA types. MirrorCAPTCHA (Wu et al., 2026a) instead trains a unified agent with a synthetic data pipeline, while MCA-Bench (Wu et al., 2026b) fine-tunes task-specific LoRA adapters. Neither provides reasoning-annotated screenshot-action trajectories across broad CAPTCHA type coverage.

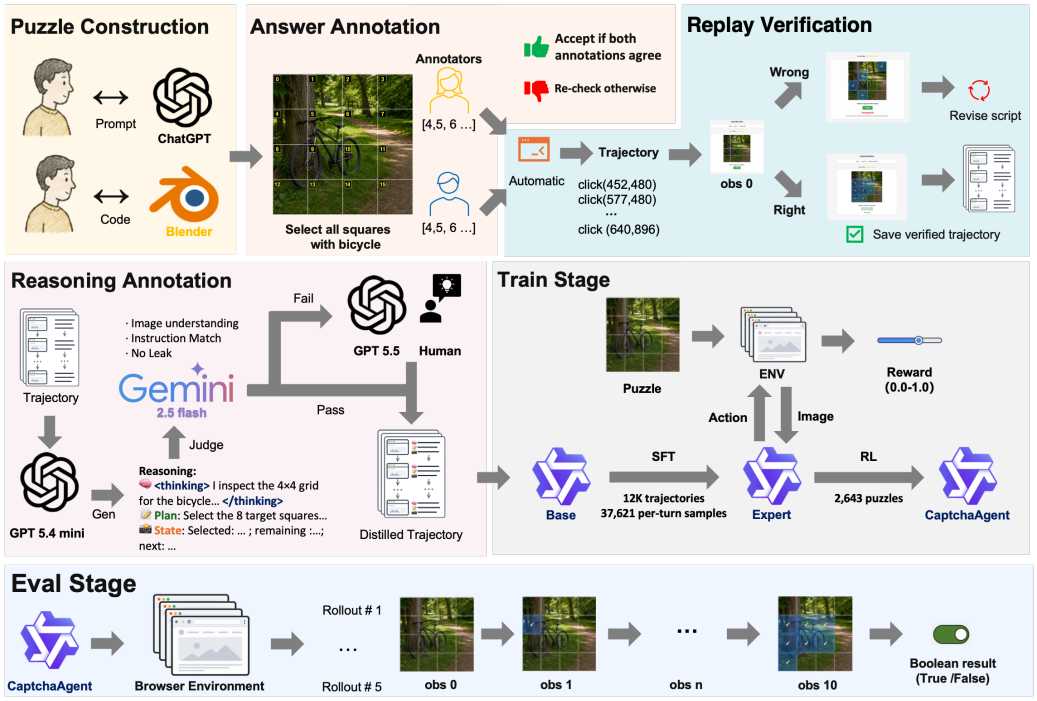


Figure 2: Pipeline from puzzle construction and verification to reasoning annotation, training, and evaluation.

3 CaptchaArena

CaptchaArena is a large-scale, fine-grained computer-use training dataset for CAPTCHA solving in a live interactive environment. It contains 50K puzzles across 20 CAPTCHA types and 5 interaction modes. Each type contributes 2,100 training puzzles and 200 each for validation and test, giving 42K, 4K, and 4K puzzles, respectively (Table 2). Unlike static CAPTCHA datasets that pair each challenge with an image and an answer, **CaptchaArena** provides closed-loop interaction supervision for training computer-use agents. Agents observe the page through a fixed 1280×1080 viewport and interact through low-level pointer and keyboard actions. Fig. 2 shows the pipeline from puzzle construction and verification to annotation, training, and evaluation. Every task requires visual recognition. Some tasks additionally require grounding for pixel-precise localization, logical reasoning, or mathematical computation (Table 4). Two authors annotated these skills independently, and the first author adjudicated disagreements.

3.1 Puzzle Construction

Action space. The agent has 5 tool calls: screenshot, click(x, y), type_text(s), drag(x_0, y_0, x_1, y_1) and timed hold$(x, y, \Delta t)$ (App. A). At each turn, it observes the current viewport with the task prompt, emits one action, and the environment returns the next screenshot. Drag and hold support interactions that cannot be expressed through clicks and text input alone. An episode ends when the puzzle is submitted. The agent presses submit, except in 4 types where submission occurs automatically after the solving action or when a timer expires. The same schema is used during replay, annotation, supervised fine-tuning, reinforcement learning and evaluation.

Puzzle generation. Puzzles are generated under task-specific constraints. Geometry Click and Path Finder are procedurally generated using Blender,[1] which provides precise control over geometry, viewpoint and lighting together with exact ground truth. The images for most other types are

[1] https://www.blender.org/

Table 2: Dataset statistics for each CAPTCHA type.

Task Type	Train	Val	Test	Task Type	Train	Val	Test
Geometry Click	2,100	200	200	Connect Icon	2,100	200	200
Misleading Click	2,100	200	200	Coordinates	2,100	200	200
Pick Area	2,100	200	200	Dart Count	2,100	200	200
Place Dot	2,100	200	200	Image Matching	2,100	200	200
Select Animal	2,100	200	200	Object Match	2,100	200	200
Bingo	2,100	200	200	Path Finder	2,100	200	200
Click Order	2,100	200	200	Rotation Match	2,100	200	200
Image Recognition	2,100	200	200	Hold Button	2,100	200	200
Patch Select	2,100	200	200	Slide Puzzle	2,100	200	200
Unusual Detection	2,100	200	200	Dice Count	2,100	200	200
				Total	**42,000**	**4,000**	**4,000**

generated with ChatGPT Images 2.0 (OpenAI, 2026c). Three types use openly licensed icons and a public reCAPTCHA-V2 image set for part of their visual content (App. C). For images generated by ChatGPT, prompts diversify object identity, layout, visual style and distractors while preserving task semantics. Each task is described in App. B. No proprietary CAPTCHA assets are scraped or redistributed.

Checking image quality. A generator does not always return the prompt request. Dice Count shows several dice in one image, and the task is to enter their total. For example, we may ask the model to generate an image whose dice sum to 23, and the dice in the returned image may sum to a different value. Every generated image is therefore inspected for quality by two authors independently. An image is discarded and regenerated if either author rejects it. The two arithmetic types are checked further. Every Dice Count and Dart Count image is manually checked and labeled with the correct result. The rendered puzzle is checked separately (App. C).

Difficulty design. Where applicable, incorrect candidates are drawn from the same source as the correct candidate. They share its scene, style and framing, and differ only in the property being asked about. Answer positions are engineered as well. In procedurally generated types, positions are allocated by explicit quotas. Single-choice positions and arrow-cycle steps are near-uniform. The number of correct items in a multi-select task is uniform. Guessing a fixed count therefore offers no systematic advantage, and neither does guessing a fixed cycle position. Localization types are exempt, since their answers follow from the target itself (App. C).

3.2 Answer Annotation

Ground truth comes from three sources: generation-time labels, manual annotation, and renderer-derived masks. Most types bind the image to its label at generation time, therefore the reference answer is computed by code rather than read from the rendered result. Types whose answers are a precise click or a set of selected patches are annotated by hand instead. Place Dot and Patch Select are independently annotated by two annotators, with the first author resolving disagreements. Localization types with irregular targets, such as Geometry Click, store a pixel mask. Irregular targets are evaluated against a mask rather than a bounding box (App. D). For irregular-target tasks other than Misleading Click, the server accepts a click iff the corresponding mask pixel is set. Misleading Click uses the complementary rule, therefore its mask marks the region the agent must avoid. All 7,500 puzzles of the 3 irregular-target types carry a mask. Types whose target is a single point are exempt and use a stored coordinate with an explicit tolerance.

3.3 Replay Verification

Executing a puzzle's solution both verifies it and turns it into training data. A stored answer may be a semantic label, a coordinate, a set of coordinates, or a value. We compile it into an executable sequence of browser actions and run that sequence in the environment, recording a screenshot before every action. Every puzzle therefore yields one screenshot-action trajectory. The same run also

checks the answer. A puzzle is retained only if the final state is accepted by the page's verifier. A puzzle often admits more than one solution, and the same solution can be ordered in more than one way. Therefore, the environment can produce many trajectories for one puzzle, including ones that take a detour and recover from it. We release the environment and one accepted trajectory per puzzle.

4 CAPTCHAAGENT

CaptchaAgent is a single policy over all 20 types, trained in two stages. The first stage is supervised fine-tuning on reasoning-annotated, replay-verified trajectories from §3. The second stage is reinforcement learning with the environment verifier as reward. A single set of model parameters, tool schema, and observation format is shared across all task types, with no per-task adapter or task-conditioned switching. §4.1 describes reasoning annotation, §4.2 supervised fine-tuning, and §4.3 reinforcement learning.

4.1 REASONING ANNOTATION

Replay yields one screenshot-action trajectory per puzzle (§3). The actions are correct, but provide no explanation for why they were chosen. We therefore annotate every step with reasoning.

Teacher model. We give a single teacher (GPT-5.4-mini, OpenAI, 2026a) the screenshot at each turn with the correct action, and ask it to produce the reasoning that leads to that action. The action is described in task-level terms rather than as a coordinate, thus the teacher must locate the target in the image.

Judge filtering. An independent judge (Gemini-2.5-Flash, Comanici et al., 2025) evaluates each generation twice. A sample is admitted only if both checks accept it. The judge comes from a different model family, so no model evaluates its own output. It inspects image and text together and rejects on three criteria. *Consistency.* Claims in the annotation must agree with the verified solution. *No hindsight.* Answer-revealing phrasing is rejected, while first-person perceptual descriptions are allowed. *Decisiveness.* Hedging and tentative language are rejected because the policy must commit to an action. Descriptions of icon and tile appearance are allowed because visual grounding is part of the annotation.

Additional verification. Rejected samples are returned to the teacher with the judge's feedback for regeneration. Because the judge can also make visual errors, samples still rejected after three attempts are checked by a stronger vision model (GPT-5.5, OpenAI, 2026b). It also checks every sample from Click Order and Connect Icon. In Click Order, weaker models may misread the glyphs, while in Connect Icon, they may miss the faint dashed links after an icon is moved. They may then misidentify which icons are connected. Samples disputed by the stronger model are regenerated under the same feedback loop. An author manually reviews, repairs, or discards any remaining cases.

Annotated data. Annotation covers the 42K training and 4K validation puzzles, giving 46K reasoning-annotated trajectories. Each trajectory carries the screenshot before every action, the action, and the reasoning that leads to it. Because the chat template removes reasoning from previous turns, we expand each T-turn trajectory into T per-turn samples. Each turn is therefore supervised in the same format used at inference. Test puzzles are used only for evaluation and are never annotated.

4.2 SUPERVISED FINE-TUNING

We fine-tune on a multi-task subset spanning all 20 types. We use 600 of each type's 2,100 training puzzles, 12,000 in total. Per-turn expansion in §4.1 gives 37,621 training samples. The remaining 1,500 per type are held out and reserved for reinforcement learning on the same environment. For SFT, the 200 validation puzzles per type are used to monitor training, but not for checkpoint selection.

The backbone is Qwen3.5-9B (Qwen Team, 2026) with rank-64 LoRA adapters (Hu et al., 2022) applied only to the language model. The vision encoder and the vision–language aligner are frozen. Training and evaluation use the same screenshot resolution and preprocessing. We train for 3 epochs and report the final checkpoint. App. K gives the full configuration.

4.3 Reinforcement Learning

Supervised fine-tuning leaves two failure modes unresolved. The policy does not always follow a task through to a submission, and its actions do not always land precisely enough for the verifier to accept them. We therefore continue training the supervised policy with GRPO (Shao et al., 2024) against the live environment. The environment verifier provides the reward. No reward model is learned, no preference data is collected, and no step-level reward annotation is required. The same verifier used to accept dataset solutions scores each rollout. Episodes are capped at 15 turns.

Two design choices are important. First, difficulty is mined per puzzle rather than per type, since GRPO provides no learning signal when every rollout for a puzzle receives the same reward. Second, the reward is monotone both in submitting and in closeness to the answer, thus the policy is never rewarded for withholding an uncertain answer. App. L gives the task pool, the reward formula, the optimization configuration, and the shortcuts observed before each reward term was introduced.

5 Experiments

5.1 Evaluation Setup

We evaluate every system through the same interface. Each receives screenshots as input and predicts pixel-level actions, without an agent framework, set-of-mark overlay or accessibility tree. The test set contains all 4,000 puzzles across 20 types, with 200 puzzles per type. **CaptchaAgent** is sampled 5 times per puzzle, and every other system once. An episode ends when the page submits or the 15-step cap is reached. It counts as solved only when the server-side verifier accepts the submitted state. We report Pass@k using the unbiased estimator of Chen et al. (2021), including Pass@5 for **CaptchaAgent** (App. E). Because every type contributes the same number of test puzzles, the overall score is the unweighted mean across types. We also report the full per-type breakdown.

Comparisons. We compare against the untrained Qwen3.5 backbone at 9B, 27B, and 35B-A3B, six open-weight GUI agents from 7B to 9B, three closed-source models, and a human reference of two annotators (App. G). The 9B backbone differs from **CaptchaAgent** only in its weights.

5.2 Results

Table 3 reports every type. The untrained 9B backbone averages 11.4 Pass@1, and scaling within the Qwen3.5 family to 27B and 35B-A3B does not improve performance (Table 6). The composition of this score is more informative than the average itself. The 9B backbone averages 56.6 on the 4 types that submit automatically, but at most 0.5 on the 16 that require an explicit submit action. Its overall submit rate is 20.5% (Table 7). Its dominant failure is procedural rather than perceptual (App. J.2).

Across the evaluated systems, scale alone does not explain performance. The six open-weight GUI agents top out at 35.2 (Table 6), while the three closed-source models, GPT 5.4 (OpenAI, 2026a), Gemini 3.5 Flash (Google DeepMind, 2026) and Claude Sonnet 4 (Anthropic, 2025), span 49.4 to 69.2. **CaptchaAgent** reaches 70.5 average Pass@1 and 86.0 Pass@5, with a 96.8% submit rate. 8 types reach 90 or higher at Pass@1, 7 of those sit within 5 points of the human reference, and 4 types in all match or exceed it. The remaining gap to the human average of 94.1 is concentrated rather than uniform. Four types account for three-fifths of the 23.6-point gap.

Reinforcement learning. The best RL checkpoint improves average Pass@1 by +1.2 over the supervised policy, or +1.5 excluding Click Order, with 16 of 20 types improving (full paired Pass@k in Table 5). The largest gains span the difficulty range rather than sitting at its bottom. Connect Icon improves by +5.1 from 71.7, Place Dot by +4.0 from 16.1, Image Recognition by +3.5 from 56.8, and Coordinates by +3.3 from 96.6, while the two weakest types move only slightly: Dice Count by

Table 3: Per-type Pass@1 on the full test split, with 200 puzzles per type. Bold marks the best model result in each row; the human reference is excluded from this comparison.

	Untrained Base	GUI agents		Closed-source			**CaptchaAgent**		
Type	Qwen3.5-9B	Venus2-9B	GUI-Owl-8B	GPT-5.4	Gemini	Claude	SFT	RL	Human
Geometry Click	0.0	**93.5**	89.5	92.0	79.5	89.5	68.2	68.2	96.5
Misleading Click	93.5	25.5	82.0	23.0	87.0	83.0	**99.1**	98.9	100.0
Pick Area	35.5	23.0	33.0	37.5	37.5	36.0	59.8	**60.6**	97.5
Place Dot	0.0	9.5	22.5	**88.5**	63.0	15.5	16.1	20.1	100.0
Select Animal	0.0	28.5	99.5	98.5	90.5	**100.0**	99.3	99.4	98.0
Bingo	0.0	0.5	4.5	20.0	**93.5**	48.0	56.8	56.3	100.0
Click Order	0.0	0.0	62.5	81.5	62.0	58.0	**93.2**	90.1	97.5
Image Recognition	0.0	10.5	54.0	**64.0**	62.5	39.5	56.8	60.3	65.5
Patch Select	0.0	1.0	5.0	9.0	**12.5**	12.0	9.6	10.4	73.5
Unusual Detection	0.0	2.0	5.5	16.5	73.5	13.5	85.1	**85.7**	83.5
Connect Icon	0.0	48.5	13.5	45.0	**90.5**	60.5	71.7	76.8	98.5
Coordinates	0.0	62.0	29.5	28.5	77.5	87.5	96.6	**99.9**	98.5
Dart Count	0.0	99.0	28.0	43.5	82.5	94.5	99.0	**99.2**	98.0
Image Matching	0.0	50.0	95.0	93.5	90.0	93.5	98.2	**98.5**	99.0
Object Match	0.5	73.5	28.0	47.5	71.0	66.5	89.1	**91.5**	95.5
Path Finder	0.0	66.5	14.0	54.5	**78.0**	49.5	71.9	72.9	100.0
Rotation Match	0.0	3.5	10.0	22.0	70.0	40.5	91.5	**94.5**	100.0
Hold Button	97.5	**100.0**	0.0	**100.0**	**100.0**	**100.0**	99.9	**100.0**	98.0
Slide Puzzle	0.0	1.0	2.0	13.5	40.0	16.5	42.2	**43.9**	99.5
Dice Count	0.0	6.5	1.0	9.0	**22.5**	7.5	5.7	7.3	83.0
Average	11.4	35.2	34.0	49.4	69.2	55.6	70.5	**71.7**	94.1

$+1.6$ and Patch Select by $+0.8$. The improvement is larger at Pass@1 ($+1.2$) than at Pass@5 ($+0.8$). This pattern is consistent with RL improving first-rollout reliability more than expanding the set of puzzles the policy can solve across repeated attempts. Click Order is the only substantial regression, decreasing by 3.1 points. RL also raises Pass@1 from 47.2 to 51.0 on Open CaptchaWorld and from 13.6 to 20.0 on Halligan (App. I).

5.3 Error Analysis

We diagnose failures with 3 signals: submit rate, Pass@1, and Pass@5. A low submit rate shows that the policy often stops without finishing the episode. A low Pass@5 indicates that the policy cannot find a valid solution even across repeated attempts. A large gap between Pass@1 and Pass@5 indicates that a valid solution is reachable but not reliably reached on the first rollout. After SFT, 4 types remain below 45 Pass@1: Place Dot, Patch Select, Slide Puzzle, and Dice Count. All 4 use all-or-nothing grading, where a single local error invalidates the episode. Their failures fall into 3 patterns.

Non-submission. A rollout that never submits is scored as a failure regardless of page state. Untrained models submit on 20.0% to 20.5% of rollouts. Training largely removes the failure. The policy submits on 96.8% of rollouts, and on at least 99% in 13 of the 20 types (App. F). Dice Count remains an exception. The supervised policy achieves only a 54.4% submit rate, so nearly half its episodes are scored as failures with the answer never checked. The task itself is also hard. Humans reach 83.0 here, their third-lowest score. The best system we evaluate reaches 22.5. The 5.7 Pass@1 therefore reflects both task difficulty and non-submission rather than pure counting ability.

Weak grounding. Two types remain low even across repeated attempts. Patch Select is difficult for every evaluated system. No model exceeds 12.5, while humans reach 73.5. Place Dot is instead a specific weakness of **CaptchaAgent**. The policy submits on 95.3% of rollouts but reaches only 16.1 Pass@1, while GPT-5.4 reaches 88.5 on the same puzzles. Five attempts lift both types, to 27.5 and 43.2, but both remain far below their human references. Place Dot therefore exposes a remaining weakness in precise grounding.

Unstable execution. Some types reach valid solutions across repeated attempts but do so unreliably on the first rollout. The policy rises from 70.5 at Pass@1 to 86.0 at Pass@5 (App. E), a

15.5-point gap. The gap is not uniform. It exceeds 30 points in Slide Puzzle (42.2 to 91.0), Bingo (56.8 to 92.5) and Pick Area (59.8 to 93.0). However, it stays under 10 points in the 8 types already above 90. These tasks therefore expose execution variance rather than uniformly missing capability.

6 Conclusion

We introduced **CaptchaArena**, a large-scale, fine-grained computer-use training dataset containing 50K CAPTCHA puzzles across 20 types. Unlike static image–answer datasets, **CaptchaArena** represents each puzzle as an executable browser task. Every answer is a browser program, admitted only after its replay is accepted by the site's own verifier. The same process produces 50K screenshot-action trajectories, including 46K with judge-filtered reasoning annotations. For irregular targets, **CaptchaArena** additionally provides pixel-mask supervision, allowing clicks to be graded against the target shape rather than a bounding box.

Using **CaptchaArena**, we further train **CaptchaAgent**, a single 9B policy across all 20 types. It reaches 70.5 Pass@1 after supervised fine-tuning and 71.7 after reinforcement learning using the same environment verifier as the reward. This compares with 11.4 for the untrained backbone, 35.2 for the strongest open-weight GUI agent we evaluate, 69.2 for the strongest closed-source model and 94.1 for humans. Scaling the Qwen3.5 backbone does not close the gap, since a 35B-A3B model also scores 11.4, while the closed-source models span 49.4 to 69.2. These results point to training data and interaction supervision, rather than scale alone, as a key factor in CAPTCHA performance. The remaining gap to humans is concentrated in a small number of types and reflects three main failure modes: non-submission, weak grounding, and unstable execution. Because the environment deterministically verifies the final page state, reinforcement learning requires neither a learned reward model nor human reward labels.

Limitations. Reasoning annotations depend on commercial models, although they explain only replay-verified actions. The vision encoder stays frozen, and our largest gap to the human reference shows on a task requiring precise grounding. Actions are discrete, so trajectories cannot reproduce mouse-trajectory signals used by behavioral detectors (Acien et al., 2022).

Ethics Statement

CaptchaArena runs entirely in an environment we host ourselves. The external suites in App. I are also served locally. Halligan contains puzzles derived from CAPTCHA types deployed on real websites, but no experiment interacts with a live CAPTCHA service or deployed anti-bot system, and we neither collect nor replay traffic from one. **CaptchaArena** imagery is generated or openly licensed, with no proprietary assets redistributed (App. C). The work studies CAPTCHA solving as a computer-use capability rather than bypassing protections on services we do not own. Because this capability is dual use, we release the dataset to support both agent evaluation and the study of CAPTCHA robustness.

Reproducibility Statement

Each released puzzle includes an executable browser-action solution that can be replayed against the environment's own verifier. A third party can therefore confirm that a released solution solves its puzzle without access to our generation pipeline (§3). We do not claim that rerunning the stochastic generation process reconstructs the dataset exactly. §5.1 specifies the evaluation protocol and Pass@k estimator.

Use of Large Language Models

Large language models and other AI tools were used for four purposes. First, OpenAI models were used to generate images for most **CaptchaArena** task types. Second, as part of research execution, GPT-5.4-mini generated reasoning annotations, while Gemini-2.5-Flash and GPT-5.5 were used to verify them. Third, an AI assistant was used to polish the writing. Finally, it was used to draft sections and passages that were subsequently reviewed and revised by the authors. All claims, numbers, and citations were verified by the authors, who take responsibility for the final content.

REFERENCES

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://doi.org/10.1016/j.patcog.2022.108643.

Deepak Akkil, Mowafak Allaham, Amal Raj, Tamer Abuelsaad, and Ravi Kokku. Emergence WebVoyager: Toward consistent and transparent evaluation of (web) agents in the wild. In *ICLR Workshop on Agents in the Wild*, 2026. URL https://arxiv.org/abs/2603.29020.

Anthropic. Introducing Claude 4. https://www.anthropic.com/news/claude-4, May 2025.

Ahmed Awadallah, Sahil Gupta, Yash Lara, Yadong Lu, Hussein Mozannar, Akshay Nambi, Zach Nussbaum, et al. Fara-1.5: Scalable learning environments for computer use agents, 2026. URL https://arxiv.org/abs/2606.20785.

ByteDance Seed. UI-TARS-1.5. https://seed-tars.com/1.5, 2025.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021. URL https://arxiv.org/abs/2107.03374.

Yuxi Chen, Haoyu Zhai, Chenkai Wang, Rui Yang, Lingming Zhang, Gang Wang, and Huan Zhang. CAPTCHA solving for native GUI agents: Automated reasoning-action data generation and self-corrective training. In *International Conference on Machine Learning*, 2026. URL https://openreview.net/pdf?id=HWTRHS26ZF.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities, 2025. URL https://arxiv.org/abs/2507.06261.

Gelei Deng, Haoran Ou, Yi Liu, Jie Zhang, Tianwei Zhang, and Yang Liu. Oedipus: LLM-enchanced reasoning CAPTCHA solver. In *ACM SIGSAC Conference on Computer and Communications Security*, pp. 6–20. ACM, 2025. URL https://doi.org/10.1145/3719027.3744872.

Google DeepMind. Gemini 3.5 Flash. https://deepmind.google/models/model-cards/gemini-3-5-flash/, May 2026.

H Company. Holo3.1: Fast & local computer use agents, 2026. URL https://huggingface.co/Hcompany/Holo-3.1-9B.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Russ Salakhutdinov, and Daniel Fried. VisualWebArena: Evaluating multimodal agents on realistic visual web tasks. In *Annual Meeting of the Association for Computational Linguistics*, pp. 881–905. Association for Computational Linguistics, 2024. URL https://aclanthology.org/2024.acl-long.50/.

Yaxin Luo, Zhaoyi Li, Jiacheng Liu, Jiacheng Cui, Xiaohan Zhao, and Zhiqiang Shen. Open CaptchaWorld: A comprehensive web-based platform for testing and benchmarking multimodal LLM agents. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://openreview.net/forum?id=dhWkIa3NFu.

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

OpenAI. GPT-5.4 Thinking system card. https://deploymentsafety.openai.com/gpt-5-4-thinking, March 2026a.

OpenAI. GPT-5.5 system card. https://deploymentsafety.openai.com/gpt-5-5, April 2026b.

OpenAI. Introducing ChatGPT images 2.0. https://openai.com/index/introducing-chatgpt-images-2-0/, April 2026c.

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. arXiv preprint arXiv:2501.12326, 2025. URL https://arxiv.org/abs/2501.12326.

Qwen Team. Qwen3.5: Towards native multimodal agents, 2026. URL https://qwen.ai/blog?id=qwen3.5.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://doi.org/10.48550/arXiv.2402.03300.

Python Song, Luke Tenyi Chang, Yun-Yun Tsai, Penghui Li, and Junfeng Yang. Reasoning under vision: Understanding visual-spatial cognition in vision-language models for CAPTCHA. arXiv preprint arXiv:2510.06067, 2025. URL https://arxiv.org/abs/2510.06067.

Xinhao Song, Su Su, Sirui Song, Hongliang Wu, Wen Shen, Zhihua Wei, Gongshen Liu, Linfeng Zhang, and Dongrui Liu. HLL: Can agents cross humanity's last line of verification? arXiv preprint arXiv:2606.02449, 2026. URL https://arxiv.org/abs/2606.02449.

Xiwen Teoh, Yun Lin, Siqi Li, Ruofan Liu, Avi Sollomoni, Yaniv Harel, and Jin Song Dong. Are CAPTCHAs still bot-hard? Generalized visual CAPTCHA solving with agentic vision language model. In *USENIX Security Symposium*, pp. 3747–3766. USENIX Association, 2025. URL https://www.usenix.org/conference/usenixsecurity25/presentation/teoh.

Venus Team, Zhuohan Cai, Haoxing Chen, Jiaxuan Chen, Weizhi Chen, Changlong Gao, Zhangxuan Gu, Yuan Guo, Yusong Hu, Jianrong Jiang, Jianguo Li, Runze Li, Jinzhen Lin, Zhenyu Ma, Changhua Meng, Han Peng, Xinyu Qiu, Shuheng Shen, Zhongyi Shui, Weiqiang Wang, Ming Wen, Zhuoer Xu, Hang Yan, Kaiwen Yang, Ruilin Yao, Nanjun Yu, Zhengwen Zeng, Lianrui Zhang, Yunzhu Zhang, Zhe Zhao, and Beitong Zhou. UI-Venus-2 technical report, 2026. URL https://arxiv.org/abs/2609.00028.

Pengcheng Wang, Haoxiang Liu, Yang Dai, Xiangxiang Zeng, Guanhua Chen, Baotian Hu, Longyue Wang, and Weihua Luo. CaptchaMind: Training captcha solvers via reinforcement learning with explicit reasoning supervision. arXiv preprint arXiv:2605.19538, 2026. URL https://arxiv.org/abs/2605.19538.

Xiangyu Wu, Yuwei Hu, Tianyu Cui, Yueying Tian, Qing-Guo Chen, Zhao Xu, Weihua Luo, Kaifu Zhang, Yang Yang, and Jianfeng Lu. MirrorCAPTCHA: Wild CAPTCHA, wild distribution, wild web-based platform meet multimodal LLM agents. In *Annual Meeting of the Association for Computational Linguistics*, pp. 31001–31017, San Diego, California, United States, 2026a.

Association for Computational Linguistics. URL https://aclanthology.org/2026.acl-long.1431/.

Zonglin Wu, Yule Xue, Yaoyao Feng, Xiaolong Wang, and Yiren Song. MCA-Bench: A multimodal benchmark for evaluating CAPTCHA robustness against VLM-based attacks. In *AAAI Conference on Artificial Intelligence*, volume 40, pp. 38039–38047. AAAI Press, 2026b. URL https://doi.org/10.1609/aaai.v40i44.41141.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In *Advances in Neural Information Processing Systems*, volume 37, pp. 52040–52094, 2024. URL https://osworld-v1.xlang.ai/.

Haiyang Xu, Xi Zhang, Haowei Liu, Junyang Wang, Zhaozai Zhu, Shengjie Zhou, Xuhao Hu, Feiyu Gao, Junjie Cao, Zihua Wang, Zhiyuan Chen, Jitong Liao, Qi Zheng, Jiahui Zeng, Ze Xu, Shuai Bai, Junyang Lin, Jingren Zhou, and Ming Yan. Mobile-agent-v3.5: Multi-platform fundamental GUI agents. *arXiv preprint arXiv:2602.16855*, 2026. URL https://arxiv.org/abs/2602.16855.

Taofeng Xue, Chong Peng, Mianqiu Huang, Linsen Guo, Tiancheng Han, Haozhe Wang, Jianing Wang, et al. EvoCUA: Evolving computer use agents via learning from scalable synthetic experience, 2026. URL https://arxiv.org/abs/2601.15876.

Tianci Xue, Weijian Qi, Tianneng Shi, Chan Hee Song, Boyu Gou, Dawn Song, Huan Sun, and Yu Su. An illusion of progress? Assessing the current state of web agents. In *Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=6jZi4HSs6o.

Jianwei Yang, Hao Zhang, Feng Li, Xueyan Zou, Chunyuan Li, and Jianfeng Gao. Set-of-Mark prompting unleashes extraordinary visual grounding in GPT-4V, 2023. URL https://arxiv.org/abs/2310.11441.

Jianyi Zhang, Ziyin Zhou, Xu Ji, Shizhao Liu, and Zhangchi Zhao. CAPTURE: A benchmark and evaluation for LVLMs in CAPTCHA resolving, 2025. URL https://arxiv.org/abs/2512.11323.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, 2024. URL https://iclr.cc/virtual/2024/poster/17826.

APPENDIX

A ACTION SPACE AND COORDINATE FRAMES

Action protocol. All agents use the same five-tool interface provided by the evaluation client: screenshot, click(x, y), type_text(s), drag(x_0, y_0, x_1, y_1), and hold$(x, y, \Delta t)$. An episode ends when the page is submitted or when the 15-turn limit is reached. Submission is explicit for most tasks. Geometry Click, Misleading Click, and Pick Area submit automatically after the solving click, while Hold Button submits when its timer expires.

Hold timing. Hold Button requires the agent to hold a button for $\Delta t \sim \mathrm{Unif}\{2, \ldots, 6\}$ seconds. The page checks the elapsed hold time every 100 ms and submits automatically once the required duration is reached. Its reference solution therefore contains a single hold$(x, y, \Delta t)$ action and no explicit submit action.

Table 4: The 20 released **CaptchaArena** tasks grouped by interaction mode. *Mode*: S single-click, M multi-click, A arrow-cycle, R real-time, T text-entry. *Skills*: G grounding, L logic, M math. *Action steps* gives the mean number of agent actions in the reference trajectory, rounded to one star per action and capped at five.

#	Task	Mode	Interaction	Skills	Action steps	Grading
1	Geometry Click	S	single-click	G	★	mask hit
2	Misleading Click	S	single-click		★	mask *miss*
3	Pick Area	S	single-click		★	mask hit
4	Place Dot	S	single-click	G	★★	distance tol.
5	Select Animal	S	single-click, 2×3 grid		★★	target cell
6	Bingo	M	multi-click, 3×3 grid	L	★★★	any valid swap
7	Click Order	M	multi-click, ordered	GL	★★★★★	click order
8	Image Recognition	M	multi-click, 3×3 grid		★★★★	$A = A^\star$
9	Patch Select	M	multi-click, $N \times N$ grid		★★★★★	$A = A^\star$
10	Unusual Detection	M	multi-click, 2×3 grid		★★★★	$A = A^\star$
11	Connect Icon	A	arrow-cycle, N=6		★★★★	$\hat{k} = k^\star$
12	Coordinates	A	arrow-cycle, N=5	L	★★★	$\hat{k} = k^\star$
13	Dart Count	A	arrow-cycle, N=5	M	★★★★	$\hat{k} = k^\star$
14	Image Matching	A	arrow-cycle, N=5		★★★	$\hat{k} = k^\star$
15	Object Match	A	arrow-cycle, N=5		★★★	$\hat{k} = k^\star$
16	Path Finder	A	arrow-cycle, N=5	L	★★★★	$\hat{k} = k^\star$
17	Rotation Match	A	arrow-cycle, 8 angles	L	★★★★	$\hat{k} = k^\star$
18	Hold Button	R	real-time hold		★	duration interval
19	Slide Puzzle	R	real-time slider		★★★	distance tol.
20	Dice Count	T	text entry	M	★★	integer equality

Coordinate frames. Ground truth and masks are stored in natural-image coordinates, with the origin at the image's top-left. Executable clicks are mostly in absolute pixels of the fixed 1280×1080 viewport. A point in natural-image coordinates is mapped to the viewport using the rendered element's bounding box (x_0, y_0, w, h) and natural size (W_n, H_n) as

$$c_{\text{view}} = \left(x_0 + \frac{w}{W_n} c_{\text{nat},x},\ y_0 + \frac{h}{H_n} c_{\text{nat},y} \right).$$

This allows the same annotation to be used consistently across render scales. The coordinate frame is stored per record rather than fixed by task type. Pick Area and Place Dot store targets in natural-image coordinates but executable clicks in viewport coordinates. Geometry Click stores both in natural-image coordinates and converts them at run time. Tasks built around fixed HTML controls use viewport coordinates only. The execution and evaluation code therefore reads the coordinate frame from each record.

B Task Catalogue

Table 4 gives each task's interaction mode, the skills it needs beyond visual recognition, the length of its reference trajectory and its grading predicate. The paragraphs below group the tasks into 4 families by page layout rather than by interaction mode. App. D gives the exact acceptance predicates of the localization tasks.

B.1 Family A: Arrow-Cycle Single Choice

The seven arrow-cycle tasks show a reference on the left and one candidate on the right. The agent uses arrow controls to cycle through candidates and submits the matching one.

Image Matching shows a type exemplar and five candidate images. Exactly one candidate belongs to the same class as the exemplar. The candidates share a similar scene and visual style, so the agent must identify the matching semantic type.

Object Match shows a counting card, for example "3 anchors" and 5 scenes containing different numbers of the target object. The agent must count the objects and submit the scene matching the requested count.

Coordinates shows a seating chart in which rows are labeled with letters and columns with numbers. The reference specifies an address such as c11, and the agent must find the candidate chart that places Jerry at that location.

Connect Icon shows a reference pair of icons connected by a dashed line. Each candidate contains the same set of icons but connects a different pair. The agent cycles through the candidates and submits the one whose connection matches the reference.

Path Finder shows a 2D path schematic with a marked location and several rendered views of the same scene, each placing a marker at a different position. The agent must transfer the marked location from the schematic to the rendered scene and select the matching candidate.

Rotation Match shows a hand pointing in one of 8 directions. The arrow control rotates the subject image by $45°$ at each step. The agent submits when the subject orientation matches the reference direction.

Dart Count shows a target sum and 5 candidate dartboards. Each dartboard contains numbered regions, and only regions hit by a dart contribute to the total. The agent must identify the hit regions, sum their values, and submit the board matching the target sum.

B.2 Family B: Single-Point Localization

The four Family B tasks are solved through a single image click. The page shows a single image and no candidate list. In three tasks, the click immediately submits the puzzle; Place Dot instead requires an explicit submit action. The grading rule depends on the task. Pick Area and Geometry Click accept clicks inside a target pixel mask, while Misleading Click rejects clicks inside a forbidden-region mask. Place Dot stores a target point and grades by distance. Pixel masks are used for irregular regions that cannot be represented accurately by bounding boxes.

Pick Area shows a scene divided into regions by drawn curves. The task is to click anywhere inside the largest region. Because the target region has an irregular shape, grading uses a pixel mask rather than a bounding box.

Geometry Click shows a rendered 3D scene containing multiple objects. The prompt identifies one target by color and identity, for example "click yellow four". The agent must localize that object and click it. Grading uses a pixel mask because many targets, such as digits or curved shapes, are poorly represented by bounding boxes.

Misleading Click shows a character labeled "DON'T CLICK ME". The task is to click anywhere except the character itself. Grading uses a forbidden-region mask so that only clicks on the character are rejected, while background inside its bounding box remains valid.

Place Dot shows a city map with a path drawn across it. A car is parked at one endpoint, and the task is to click the other endpoint. The stored target is a point, and grading is based on distance to that point.

B.3 Family C: Grid Multi-Select

The four Family C tasks show a grid of tiles and a prompt specifying what to select. The agent selects one or more tiles and then presses submit. Grading uses exact set equality over the selected cells, so selecting an extra tile or missing a required tile causes failure.

Image Recognition shows a 3×3 grid of photographs and names a class, for example "select crosswalks". The task is to select every tile showing that class.

Patch Select overlays an $N \times N$ grid, with $N \in \{4, 5\}$, on a photograph and names an object in the image. The task is to select every cell occupied by that object. Because objects need not align with grid boundaries, the agent must localize them precisely and judge borderline cells.

Select Animal shows a 2×3 grid of everyday objects and names one target, for example "click the camera." The task is to select the corresponding tile. Each puzzle has exactly one correct cell.

Unusual Detection shows a 2×3 grid of animals and asks the agent to select the unusual ones. An unusual tile is a chimera, such as the head of one animal combined with the body of another, while the remaining tiles show ordinary animals.

B.4 Family D: Structured Singletons

The five Family D tasks each use a different interaction pattern rather than selecting among candidates. Four end when the agent presses submit, while Hold Button ends when the page's own timer completes.

Bingo shows a 3×3 board of icons and asks the agent to swap two tiles to complete a line of three identical icons. A swap consists of two clicks, one for each tile. A puzzle may admit multiple valid swaps, and any valid one is accepted.

Click Order shows a reference sequence of glyphs and an image containing scattered copies of those glyphs together with distractors. The task is to click the target glyphs in the specified order. The scattered glyphs may be colored, warped, rotated, or outlined, requiring the agent to recognize their identities rather than match their appearance directly.

Slide Puzzle shows an image with a missing piece and a horizontal slider. Dragging the slider moves the piece across the image. The agent must align the piece with the hole and then press submit.

Hold Button shows a button that must be held until a progress indicator completes. The required duration is $\Delta t \sim \mathrm{Unif}\{2, \ldots, 6\}$ seconds, and the page submits automatically once the threshold is reached (App. A).

Dice Count shows $K \in \{1, 2, 4, 6\}$ images containing multiple dice and asks for the total number of pips across all images. The agent computes the sum, enters it into a text field, and submits.

C Generation, Compositing, and Quality Control

Asset provenance. Most generated images are produced with ChatGPT Images 2.0 (OpenAI, 2026c). Connect Icon and Click Order additionally use an openly licensed SVG icon collection.[2] We release the composited puzzle image, not the underlying icon collection. Image Recognition uses a public Google reCAPTCHA V2 image dataset released under CC0 1.0.[3] Geometry Click and Path Finder use Blender renders. Each CAPTCHA type maintains its own asset pool, although types requiring similar visual content may draw from overlapping source images.

Hand-annotated answers. Four types require answers to be read from the finished image rather than taken directly from the generator: Dice Count, Dart Count, Patch Select, and Place Dot. These correspond to the pip total, dart sum, occupied grid cells, and path endpoint, respectively.

Rendered-puzzle quality control. After checking all generated images, we randomly sample 200 rendered puzzles per CAPTCHA type for manual inspection of formatting and rendering quality. Two authors independently check that the instruction, interaction elements, and reference answer are rendered correctly. Invalid or duplicate samples are discarded and regenerated. Disagreements are resolved by the first author.

Answers known by construction. For procedurally constructed tasks, some answers are available directly from the rendering process. The irregular-target localization tasks obtain their pixel masks from the generated target layer or renderer, rather than manual tracing. Grid-based tasks place

[2] https://www.svgrepo.com/

[3] https://www.kaggle.com/datasets/mikhailma/test-dataset

known source tiles into known cells, so their answer sets are also determined during construction. Exact grading rules are given in App. D.

Fixed layout and screenshot capture. Page layouts are fixed within each CAPTCHA type rather than randomly repositioning controls across puzzles. Before each screenshot, the capture routine moves the cursor to $(5, 5)$, away from all controls, preventing hover effects from a previous action from appearing as part of the next observation.

Answer balancing. Where applicable, answer positions are balanced across puzzles so that a fixed positional guess provides no systematic advantage. Single-choice and arrow-cycle tasks use near-uniform answer positions, while the number of correct items in multi-select tasks is also balanced. Localization tasks are excluded because their answers are determined by the target location itself.

D Certificate Schema and Acceptance Regions

Released schema. Each puzzle carries a raw semantic label and an executable certificate over the actions click, drag, type_text, and hold. Bingo and the arrow-cycle tasks store a nested list of action sequences, where each member represents a valid reference solution. Only Bingo contains multiple sequences because a puzzle may admit several valid swaps; the arrow-cycle tasks contain a single sequence. The remaining tasks store a flat action sequence. A parser therefore reads both the nesting structure and each tool call's action field.

Acceptance predicates. The four localization tasks use different grading rules. Three use a stored pixel mask. A click c is mapped to integer natural-image coordinates and tested as

$$\mathrm{hit}(c, M) = \mathbf{1}\big[M(\lfloor c \rfloor) > 127\big]. \tag{1}$$

No morphology is applied at grading time.

Geometry Click accepts iff $\mathrm{hit}(c, M_{\mathrm{geo}}) = 1$. The generator dilates the target with a 10 px elliptical kernel before storing the mask, so the tolerance is encoded in the mask rather than added by the grader. The reference click is chosen as the deepest interior point of the undilated target.

Pick Area accepts iff $\mathrm{hit}(c, M_{\mathrm{pick}}) = 1$. Its mask is the generated target region without dilation.

Misleading Click uses the complementary rule. Its mask marks the forbidden character, and any click outside that region is accepted.

Place Dot has a point target rather than a region and therefore does not use a mask. A click is accepted iff $\|c - c^\star\|_2 < 12.5$ in natural-image pixels.

Why masks over rectangles. A bounding box can include background outside an irregular target, especially for concave or elongated shapes. A fixed-radius region can instead be too strict for large targets and too permissive for small ones. Pixel masks follow the target footprint directly and therefore support irregular shapes without these errors. Place Dot retains a distance-based rule because its target is a single point rather than a region. App. J.1 measures the over-acceptance introduced by bounding boxes on evaluated clicks.

E Full Pass@k Breakdown

Tables 3 and 6 report Pass@1 for every system. Here we report the full Pass@1 through Pass@5 curves for both of our policies on the same test split Table 5. Values are computed using the unbiased estimator in §5.1 from five rollouts per puzzle. Other systems are omitted because they are evaluated with only one rollout per puzzle, for which only Pass@1 is available.

The gap between the two ends of a row is the quantity of interest. Where Pass@1 is low but Pass@5 is high, the policy can find the right answer within five attempts but cannot do so reliably on the first attempt. This indicates a variance problem rather than a failure of comprehension. When both Pass@1 and Pass@5 are low, additional attempts provide little benefit, indicating that the main limitation is what the policy can reliably discriminate. §5 discusses these patterns by task type.

Table 5: Per-type Pass@k breakdown for the SFT and RL policies on the test split.

Type	@1	@2	@3	@4	@5
Geometry Click	68.2	82.9	87.9	90.4	92.0
Misleading Click	99.1	100.0	100.0	100.0	100.0
Pick Area	59.8	78.1	86.1	90.3	93.0
Place Dot	16.1	26.8	34.1	39.3	43.2
Select Animal	99.3	100.0	100.0	100.0	100.0
Bingo	56.8	77.8	86.4	90.5	92.5
Click Order	93.2	98.4	99.2	99.6	100.0
Image Recognition	56.8	69.0	74.3	77.2	79.0
Patch Select	9.6	16.4	21.2	24.8	27.5
Unusual Detection	85.1	92.7	94.8	95.8	96.5
Connect Icon	71.7	89.1	95.0	97.4	98.5
Coordinates	96.6	99.1	99.4	99.5	99.5
Dart Count	99.0	99.5	99.5	99.5	99.5
Image Matching	98.2	98.9	99.0	99.0	99.0
Object Match	89.1	93.3	95.0	95.7	96.0
Path Finder	71.9	84.5	89.3	91.9	93.5
Rotation Match	91.5	97.3	99.0	99.7	100.0
Hold Button	99.9	100.0	100.0	100.0	100.0
Slide Puzzle	42.2	65.0	78.1	86.0	91.0
Dice Count	5.7	10.2	13.9	17.1	20.0
Average	**70.5**	**79.0**	**82.6**	**84.7**	**86.0**

(a) Final SFT checkpoint (§4.2)

Type	@1	@2	@3	@4	@5
Geometry Click	68.2	78.9	83.3	86.0	88.0
Misleading Click	98.9	100.0	100.0	100.0	100.0
Pick Area	60.6	78.0	86.2	90.7	93.5
Place Dot	20.1	31.7	39.2	44.7	49.2
Select Animal	99.4	100.0	100.0	100.0	100.0
Bingo	56.3	79.2	89.3	94.0	96.5
Click Order	90.1	97.6	98.8	99.0	99.0
Image Recognition	60.3	71.9	76.8	79.8	82.0
Patch Select	10.4	17.5	23.0	27.4	31.0
Unusual Detection	85.7	93.6	95.8	96.9	97.5
Connect Icon	76.8	90.7	95.0	96.9	98.0
Coordinates	99.9	100.0	100.0	100.0	100.0
Dart Count	99.2	99.5	99.5	99.5	99.5
Image Matching	98.5	99.0	99.3	99.4	99.5
Object Match	91.5	95.0	96.1	96.6	97.0
Path Finder	72.9	85.8	90.8	93.2	94.5
Rotation Match	94.5	98.6	99.7	100.0	100.0
Hold Button	100.0	100.0	100.0	100.0	100.0
Slide Puzzle	43.9	66.2	78.0	84.2	87.4
Dice Count	7.3	12.5	16.5	19.7	22.5
Average	**71.7**	**79.8**	**83.4**	**85.4**	**86.8**

(b) Best GRPO checkpoint (§4.3)

F Submit Rates

Table 3 reports accuracy. Table 6 extends the comparison to the two larger untrained backbones and all six open-weight GUI agents, while Tables 7 and 8 report submit rates.

The untrained columns make the failure concrete. Outside the four types that submit automatically, the untrained models reach a graded submission on 0 to 3% of episodes. They act on the page and then stop without ever committing to an answer, so almost none of their attempts are graded. Place Dot is a clear example. The 9B model submits on only 0.5% of episodes, compared with 95.3% for the supervised policy on the same puzzles.

G Human Reference

Two annotators solved all 4,000 test puzzles over 10 days, each completing 200 randomly assigned puzzles per day, so every puzzle was solved exactly once. They used the same rendered page, fixed viewport, and interaction interface as the agents, with no answer key, hints, or second attempt. A puzzle counts as solved only when the page's verifier accepts the first submission. Neither annotator was familiar with the task formats beforehand. We report the median time per puzzle because occasional idle periods can substantially inflate the mean.

The gap to the supervised policy is concentrated in a few task types. Place Dot (-83.9), Dice Count (-77.3), Patch Select (-63.9), and Slide Puzzle (-57.3) together account for 14.1 points of the 23.6-point average gap. The policy matches or exceeds the annotators on Select Animal, Unusual Detection, Dart Count, and Hold Button. Reinforcement learning reduces the average gap from 23.6 to 22.4 points but leaves the same four largest deficits. A low human score has a different interpretation from a low policy score. Image Recognition has the lowest human accuracy (65.5), suggesting genuine ambiguity about which cells contain the target. In contrast, annotators solve Place Dot perfectly and Slide Puzzle in 199 of 200 cases, showing that the large policy deficits on these tasks cannot be explained by task ambiguity.

H Source-Asset Reuse Across Splits

Where a type's asset pool is large enough, splits are constructed by partitioning source assets before rendering. Types built from smaller pools cannot always be partitioned exactly, so we audited

Table 6: Per-type Pass@1 for open-weight systems on the full test split. The systems are Qwen3.5 (Qwen Team, 2026), UI-TARS-1.5-7B (Qin et al., 2025; ByteDance Seed, 2025), GUI-Owl-1.5-8B (Xu et al., 2026), UI-Venus-2-9B (Venus Team et al., 2026), Holo-3.1-9B (H Company, 2026), EvoCUA-8B (Xue et al., 2026) and Fara1.5-9B (Awadallah et al., 2026).

	Qwen3.5			Open-weight GUI agents					
Type	9B	27B	35B	UI-TARS	GUI-Owl	Venus2	Holo-3.1	EvoCUA	Fara1.5
Geometry Click	0.0	0.0	1.0	85.5	89.5	**93.5**	78.5	92.5	90.0
Misleading Click	**93.5**	**93.5**	91.5	74.0	82.0	25.5	58.5	39.5	66.5
Pick Area	35.5	**38.5**	34.5	33.0	33.0	23.0	27.5	29.5	30.0
Place Dot	0.0	0.0	0.0	8.5	22.5	9.5	5.0	**23.0**	16.5
Select Animal	0.0	0.0	0.0	93.0	**99.5**	28.5	80.0	**99.5**	**99.5**
Bingo	0.0	0.0	0.0	3.5	4.5	0.5	4.5	9.5	**14.0**
Click Order	0.0	0.0	0.0	1.5	**62.5**	0.0	27.5	43.5	42.5
Image Recognition	0.0	0.0	0.0	15.5	**54.0**	10.5	38.0	35.5	34.0
Patch Select	0.0	0.0	0.0	0.5	**5.0**	1.0	1.0	1.5	4.0
Unusual Detection	0.0	0.0	0.0	2.5	5.5	2.0	**15.0**	6.5	7.5
Connect Icon	0.0	0.0	0.0	1.5	13.5	**48.5**	11.0	15.5	7.5
Coordinates	0.0	0.0	0.0	5.5	29.5	**62.0**	30.0	27.5	18.5
Dart Count	0.0	0.0	0.0	3.0	28.0	**99.0**	61.0	22.5	19.0
Image Matching	0.0	0.0	0.0	10.0	**95.0**	50.0	62.0	91.5	63.0
Object Match	0.5	0.0	0.0	15.5	28.0	**73.5**	28.0	34.0	29.5
Path Finder	0.0	0.0	0.0	12.5	14.0	**66.5**	25.5	22.0	30.5
Rotation Match	0.0	0.0	0.0	3.5	10.0	3.5	6.5	10.5	**19.0**
Hold Button	97.5	**100.0**	**100.0**	0.0	0.0	**100.0**	76.0	0.0	0.0
Slide Puzzle	0.0	0.0	0.0	**2.5**	2.0	1.0	0.5	0.0	1.0
Dice Count	0.0	0.0	0.0	0.0	1.0	**6.5**	5.5	1.5	2.0
Average	11.4	11.6	11.4	18.6	34.0	**35.2**	32.1	30.3	29.7

Table 7: Submit rate (%) for open-weight systems. A rollout that never submits is scored as a failure, so the submit rate places an upper bound on Pass@1. Pass@1 for the same systems is in Table 6.

	Qwen3.5			Open-weight GUI agents					
Type	9B	27B	35B	UI-TARS	GUI-Owl	Venus2	Holo-3.1	EvoCUA	Fara1.5
Geometry Click	100.0	99.5	99.0	100.0	100.0	100.0	99.5	99.0	100.0
Misleading Click	99.5	100.0	100.0	100.0	100.0	100.0	97.0	100.0	100.0
Pick Area	100.0	100.0	100.0	99.5	100.0	100.0	94.0	100.0	100.0
Place Dot	0.5	0.0	0.0	67.5	100.0	52.5	56.5	100.0	98.0
Select Animal	0.0	0.0	0.0	95.0	99.5	29.0	80.0	100.0	100.0
Bingo	2.5	0.0	0.0	50.0	73.5	1.0	32.0	84.5	92.5
Click Order	0.0	0.0	0.0	45.5	94.5	1.5	56.5	81.5	66.5
Image Recognition	1.0	0.0	0.5	92.0	100.0	19.5	94.5	100.0	97.0
Patch Select	0.5	0.0	0.0	59.5	76.5	6.5	46.5	96.0	92.0
Unusual Detection	0.5	0.0	0.0	81.5	98.5	10.0	63.0	100.0	96.5
Connect Icon	3.0	0.0	0.5	17.5	78.0	97.0	63.5	77.5	32.5
Coordinates	1.0	0.0	0.0	34.0	81.5	84.5	63.5	96.5	60.0
Dart Count	1.0	0.0	0.0	19.5	69.5	100.0	73.0	52.5	66.0
Image Matching	1.5	0.0	0.0	22.5	99.0	64.5	73.0	98.5	80.0
Object Match	1.5	0.0	0.0	48.0	88.5	91.0	59.5	94.5	85.0
Path Finder	0.0	0.0	0.0	58.5	98.0	94.0	70.0	92.0	97.0
Rotation Match	0.0	0.0	0.0	59.0	90.5	29.0	35.0	91.0	91.5
Hold Button	97.5	100.0	100.0	0.0	0.0	100.0	76.0	0.0	0.0
Slide Puzzle	0.0	0.0	0.0	14.0	82.5	11.0	62.5	19.5	34.5
Dice Count	0.0	0.0	0.0	48.5	33.5	51.5	72.5	51.0	92.5
Average	**20.5**	**20.0**	**20.0**	**55.6**	**83.2**	**57.1**	**68.4**	**81.7**	**79.1**

source-image overlap between training and test using content hashes rather than filenames. This distinction matters: in Unusual Detection, all 200 test filenames also occur in training, while the content overlap is exactly zero. Under content hashing, 12 of the 20 types share no source image between training and test. Six types share source images: Image Matching (276 of 282 test images), Coordinates (113 of 192), Dart Count (101 of 293), Misleading Click (95 of 333), Patch Select (48 of 200), and Rotation Match (32 of 232). Image Recognition stores its cells in per-puzzle subdirectories not covered by the scan, so we report no overlap figure for it. We also omit Hold Button, since

Table 8: Submit rate (%) for the closed-source models and **CaptchaAgent**. Closed-source models use 1 rollout per puzzle, while *SFT* and *RL* use 5.

	Closed-source models			**CaptchaAgent**	
Type	GPT-5.4	Gemini	Claude	SFT	RL
Geometry Click	100.0	100.0	100.0	99.9	99.9
Misleading Click	100.0	99.0	100.0	99.7	99.6
Pick Area	99.0	98.0	100.0	100.0	100.0
Place Dot	100.0	67.5	100.0	95.3	99.8
Select Animal	100.0	93.0	100.0	99.9	99.8
Bingo	100.0	94.0	99.5	100.0	100.0
Click Order	100.0	71.5	95.0	96.9	97.3
Image Recognition	100.0	96.5	100.0	99.8	100.0
Patch Select	100.0	65.0	100.0	98.5	89.9
Unusual Detection	100.0	88.0	100.0	99.9	99.8
Connect Icon	100.0	91.0	97.5	99.8	99.9
Coordinates	100.0	78.0	91.0	97.4	100.0
Dart Count	100.0	85.0	97.0	99.9	99.8
Image Matching	100.0	90.5	100.0	99.2	99.3
Object Match	100.0	74.0	96.5	98.0	98.2
Path Finder	100.0	86.0	100.0	99.9	100.0
Rotation Match	100.0	79.5	100.0	98.2	100.0
Hold Button	100.0	100.0	100.0	99.9	100.0
Slide Puzzle	99.0	96.5	98.5	99.8	99.0
Dice Count	100.0	72.0	100.0	54.4	76.6
Average	**99.9**	**86.3**	**98.8**	**96.8**	**97.9**

the reused asset is only the button image, while the task itself is determined by the required hold duration.

Source-image reuse does not imply puzzle duplication. The target, distractors, layout, and answer are sampled per puzzle, so a reused image can appear in a different task instance with a different answer. We report these measurements so that readers can account for source reuse when interpreting generalization results.

I Generalization Analysis

We evaluate on two benchmarks built independently of **CaptchaArena**. Open CaptchaWorld (Luo et al., 2025) contains 463 puzzles across the same 20 task types that we adopt, but uses its own renderers, layouts, and source assets. Halligan (Teoh et al., 2025) contains 2,600 puzzles across 26 CAPTCHA types derived from real websites. Some share similar high-level task formats with our benchmark, but differ in layout, visual content, and task instantiation. Neither benchmark contains data from our generation pipeline or validation split.

Protocol. We serve both benchmarks locally and evaluate them through the same screenshot-based agent interface, 15-turn cap, and 1280×1080 viewport used throughout this paper. Open CaptchaWorld was originally evaluated through browser-use, so our conversion replaces structured element-level actions with pixel coordinates that the policy must localize itself. Published scores are therefore not directly comparable with ours because the interaction and evaluation protocols differ. Every system is evaluated with one rollout per puzzle.

Results. Reinforcement learning improves performance on both external benchmarks, by $+3.8$ Pass@1 on Open CaptchaWorld and $+6.4$ on Halligan, compared with $+1.2$ on our own test split. Submit rates also increase while the mean number of executed actions decreases on both benchmarks (Table 10).

Table 9: Human reference on 20 types at 200 puzzles each, against both policies.

Mode	Type	Human		Pass@1		Δ	
		Acc.	Time (s)	SFT	RL	SFT	RL
Single-click	Geometry Click	96.5	6.1	68.2	68.2	−28.3	−28.3
	Misleading Click	100.0	3.3	99.1	98.9	−0.9	−1.1
	Pick Area	97.5	2.4	59.8	60.6	−37.7	−36.9
	Place Dot	100.0	4.8	16.1	20.1	−83.9	−79.9
	Select Animal	98.0	5.7	99.3	99.4	+1.3	+1.4
Multi-click	Bingo	100.0	4.9	56.8	56.3	−43.2	−43.7
	Click Order	97.5	11.0	93.2	90.1	−4.3	−7.4
	Image Recognition	65.5	7.4	56.8	60.3	−8.7	−5.2
	Patch Select	73.5	5.3	9.6	10.4	−63.9	−63.1
	Unusual Detection	83.5	7.5	85.1	85.7	+1.6	+2.2
Arrow-cycle	Connect Icon	98.5	9.3	71.7	76.8	−26.8	−21.7
	Coordinates	98.5	5.1	96.6	99.9	−1.9	+1.4
	Dart Count	98.0	19.5	99.0	99.2	+1.0	+1.2
	Image Matching	99.0	18.9	98.2	98.5	−0.8	−0.5
	Object Match	95.5	5.0	89.1	91.5	−6.4	−4.0
	Path Finder	100.0	4.9	71.9	72.9	−28.1	−27.1
	Rotation Match	100.0	5.4	91.5	94.5	−8.5	−5.5
Real-time	Hold Button	98.0	3.9	99.9	100.0	+1.9	+2.0
	Slide Puzzle	99.5	9.2	42.2	43.9	−57.3	−55.6
Text entry	Dice Count	83.0	30.8	5.7	7.3	−77.3	−75.7
Average		**94.1**	**8.5**	**70.5**	**71.7**	**−23.6**	**−22.4**

Time: the median seconds per puzzle. Δ: a policy's Pass@1 minus the human accuracy, thus a positive value means the policy is ahead.

Table 10: Generalization to two external evaluation benchmarks. Best results for each metric are shown in bold.

Model	Open CaptchaWorld			Halligan		
	Pass@1	Submit	Steps	Pass@1	Submit	Steps
Qwen3.5-9B	9.8	21.1	9.5	0.9	12.6	12.7
UI-TARS-1.5-7B	14.0	53.8	9.1	0.9	13.2	12.7
SFT (§4.2)	47.2	90.6	4.2	13.6	53.4	8.7
RL (§4.3)	**51.0**	**94.0**	**3.9**	**20.0**	**66.9**	**7.2**

One rollout per puzzle. *Submit*: submission rate (%). *Steps*: mean number of executed actions per puzzle. Bold marks the highest Pass@1 in each benchmark.

On Open CaptchaWorld, RL improves on 9 types, declines on 5, and leaves 6 unchanged (Table 11a). The largest gains are Path Finder (+20.0) and Coordinates (+16.7), while the largest drops are Dart Count (−10.0) and Rotation Match (−8.3).

On Halligan, RL improves on 14 of 26 types, declines on 8, and leaves 4 unchanged (Table 11b). About 70% of the aggregate improvement comes from yandex/text (+67.0), botdetect (+29.0), and mtcaptcha (+25.0). Absolute transfer remains limited at 20.0; none of Halligan's 26 types appears in our training corpus.

Two UI-TARS results reflect interface limitations rather than CAPTCHA recognition. On Halligan, it repeatedly clicks the same coordinate on the Arkose and FunCAPTCHA start screens and reaches the puzzle in only 11 of 1,200 episodes, producing 0.0 on all 12 corresponding types. On Open CaptchaWorld, it scores 0.0 on Hold Button because its action space does not support a timed hold.

Table 11: Per-type Pass@1 on the two external evaluation benchmarks.

Type	n	Qwen	TARS	SFT	RL
Bingo	25	0.0	0.0	44.0	40.0
Click Order	20	0.0	5.0	45.0	45.0
Connect Icon	20	0.0	0.0	60.0	60.0
Coordinates	18	0.0	0.0	44.4	61.1
Dart Count	20	0.0	5.0	75.0	65.0
Dice Count	20	0.0	10.0	0.0	15.0
Geometry Click	20	0.0	15.0	5.0	20.0
Hold Button	10	100.0	0.0	100.0	100.0
Image Matching	19	0.0	15.8	94.7	94.7
Image Recognition	20	0.0	30.0	65.0	70.0
Misleading Click	20	75.0	30.0	90.0	85.0
Object Match	20	0.0	15.0	60.0	55.0
Patch Select	20	0.0	0.0	0.0	10.0
Path Finder	10	0.0	10.0	30.0	50.0
Pick Area	30	20.0	20.0	46.7	46.7
Place Dot	32	0.0	0.0	0.0	3.1
Rotation Match	48	0.0	16.7	52.1	43.8
Select Animal	30	0.0	93.3	86.7	100.0
Slide Puzzle	31	0.0	0.0	29.0	38.7
Unusual Detection	30	0.0	13.3	16.7	16.7
Average	– –	**9.8**	**14.0**	**47.2**	**51.0**

(a) Open CaptchaWorld

Type	Qwen	TARS	SFT	RL
amazon/waf	0.0	0.0	8.0	6.0
arkose/3d_rollball	0.0	0.0	2.0	2.0
arkose/dice_match	0.0	0.0	21.0	20.0
arkose/numbermatch	0.0	0.0	2.0	6.0
arkose/orbit_match	0.0	0.0	23.0	21.0
arkose/rockstack	0.0	0.0	3.0	4.0
baidu/rotate	0.0	7.0	2.0	8.0
botdetect	1.0	11.0	1.0	30.0
funcaptcha/card	1.0	0.0	19.0	11.0
funcaptcha/counting	3.0	0.0	13.0	17.0
funcaptcha/dice_pair	4.0	0.0	11.0	20.0
funcaptcha/galaxies	0.0	0.0	74.0	71.0
funcaptcha/hand_number	2.0	0.0	10.0	16.0
funcaptcha/rotated	3.0	0.0	29.0	39.0
funcaptcha/square_icon	3.0	0.0	35.0	42.0
geetest/gobang	1.0	0.0	0.0	0.0
geetest/icon	0.0	0.0	31.0	25.0
geetest/iconcrush	4.0	1.0	9.0	16.0
geetest/slide	0.0	2.0	4.0	19.0
hcaptcha	0.0	0.0	6.0	5.0
lemin	0.0	0.0	1.0	0.0
mtcaptcha	0.0	0.0	14.0	39.0
recaptchav2	0.0	0.0	1.0	2.0
tencent/vtt	0.0	1.0	30.0	30.0
yandex/kaleidoscope	0.0	1.0	1.0	1.0
yandex/text	1.0	0.0	3.0	70.0
Average	**0.9**	**0.9**	**13.6**	**20.0**

(b) Halligan

One rollout per puzzle. *Qwen*: untrained Qwen3.5-9B. *TARS*: UI-TARS-1.5-7B. *Average*: unweighted mean over task types. Open CaptchaWorld contains 463 puzzles with uneven type sizes; Halligan contains 100 puzzles per type.

J Case Studies

Two claims in the main text are examined here on individual rollouts. First, grading a click against a bounding rectangle can accept clicks that miss an irregular target (§1). App. D explains this geometrically, and App. J.1 measures the effect on clicks made during evaluation. Second, §5 identifies procedural failures that are not explained by visual recognition alone. App. J.2 illustrates these failures through examples from UI-TARS-1.5-7B and **CaptchaAgent**.

Every rollout below comes from the evaluation runs behind Tables 3 and 6. We use the same UI-TARS rollouts reported there.

J.1 Rectangle Grading Inflates Click Accuracy

Set-up. Geometry Click and Pick Area are the two types whose targets are irregular regions graded by a stored mask (App. D). Misleading Click, the third irregular-target type in §3, is excluded because its mask marks the region a click must avoid, so a bounding rectangle could only reject more clicks rather than accept more. For every evaluated click, we map the coordinate to natural-image pixels exactly as the front-end does and grade it twice: once against the stored mask and once against the tight bounding rectangle of the same mask. The tight rectangle is the most conservative axis-aligned bounding-box rule, because it is the smallest rectangle that contains the full target mask. Mask regrading reproduces the recorded server verdict on 99.5–100.0% of rollouts in every cell; the remaining differences come from one-pixel rounding at the mask boundary. The comparison therefore changes only the grading rule while holding the clicks fixed.

Reading. Table 12 shows why the two types behave differently. A Geometry Click target is a small glyph whose bounding rectangle covers only 1.1% of the image, so rectangle grading mainly admits near misses. A Pick Area target is a much larger irregular region, and its rectangle covers 65.1% of the image, allowing substantially more background to be accepted.

Table 13 shows the effect on measured accuracy. On Geometry Click, rectangle grading increases accuracy by 5.0–8.0 points across UI-TARS and the two trained policies, while the untrained back-

Table 12: How much of a target's bounding rectangle belongs to the target. Mask and rectangle areas are averaged over the 200 test puzzles of each type. The final column gives the accuracy of a click drawn uniformly inside the rectangle under the mask verifier.

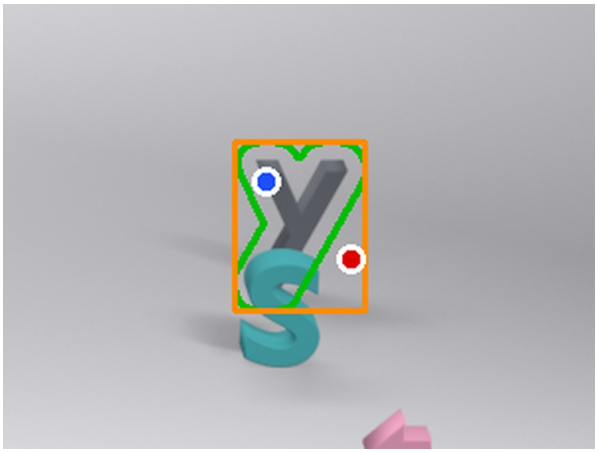
Type	mask/image	rect/image	mask/rect (mean)	mask/rect (min–max)	uniform-in-rect
Geometry Click	0.8%	1.1%	0.78	0.57–0.97	78.3
Pick Area	38.8%	65.1%	0.62	0.37–0.89	61.6

Table 13: The same clicks evaluated under mask and bounding-rectangle grading.

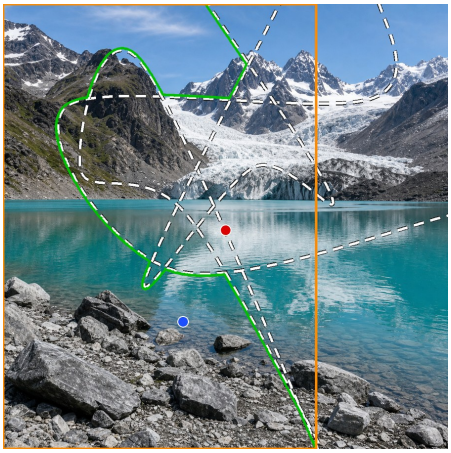
		Geometry Click			Pick Area		
System	n	Verifier	Rect	Δ	Verifier	Rect	Δ
Qwen3.5-9B (untrained)	200	0.0	0.0	+0.0	35.5	71.5	+36.0
UI-TARS-1.5-7B	200	85.5	90.5	+5.0	33.0	77.5	+44.5
SFT	1000	68.2	76.2	+8.0	59.8	81.6	+21.8
RL	1000	68.2	73.4	+5.2	60.6	82.3	+21.7
uniform click in rect		78.3	100.0		61.6	100.0	

Verifier: grading against the stored pixel mask. *Rect*: grading the same clicks against the tight bounding rectangle. Δ: the resulting accuracy inflation. n: number of graded rollouts.

bone scores zero under both rules. On Pick Area, the increase is 21.7–21.8 points for the trained policies, 36.0 points for the untrained backbone, and 44.5 points for UI-TARS. In particular, UI-TARS rises from 33.0 under the verifier to 77.5 under rectangle grading, even though its verifier score is below the 61.6 obtained by a click sampled uniformly inside the rectangle. Bounding-box grading therefore changes the apparent quality of grounding rather than simply adding a constant offset.



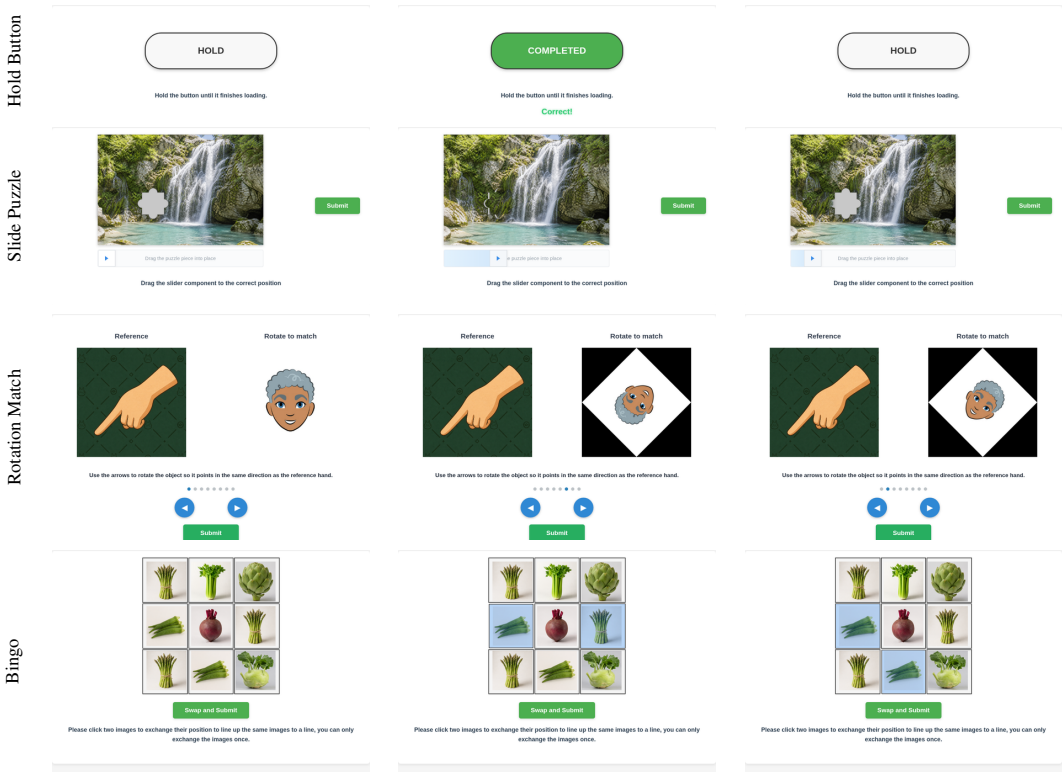
(a) Geometry Click image196, "click on the letter y". UI-TARS lands inside the rectangle, 11 px off the mask.



(b) Pick Area image2307. UI-TARS lands inside the rectangle but outside the target region.

Figure 3: Two UI-TARS clicks accepted by bounding-rectangle grading but rejected by the pixel-mask verifier. **CaptchaAgent** (RL) click on the same puzzle is shown in blue. Green denotes the verifier mask and orange its tight bounding rectangle.

Figure 3 shows two examples. The Geometry Click miss (Fig. 3a) is a genuine near miss: the UI-TARS click lands a few pixels away from the target glyph but remains inside its bounding rectangle. Rectangle grading would therefore count the click as correct even though it misses the target under the page's verifier. The Pick Area example (Fig. 3b) shows a larger grounding error: the click lands inside the bounding rectangle but outside the target region. These examples illustrate how rectangle grading can accept both small localization errors and clicks on background enclosed by an irregular target's bounding box.



(a) Initial page (b) **CaptchaAgent** (c) UI-TARS-1.5-7B

Figure 4: Four examples spanning real-time, arrow-cycle, and multi-click interactions. Column (a) shows the initial page; columns (b) and (c) show the last state reached by the **CaptchaAgent** and UI-TARS before submitting or stopping. Action traces and outcomes are listed in Table 14.

Table 14: Action traces behind Figure 4. Coordinates are viewport pixels; $\times k$ denotes repetition of the same action. *Step cap* means that the episode reaches the step limit without submission.

Puzzle	System	Steps	Actions	Outcome
Hold Button	**CaptchaAgent**	1	hold(640,408,Δt)	correct
	UI-TARS	15	14×click on and around the button; 1×drag	step cap
Slide Puzzle	**CaptchaAgent**	3	drag(325→394); drag(394→463) at y=666; submit	correct
	UI-TARS	12	3×click on the handle; 9×drag of 6–18 px	no submission
Rotation Match	**CaptchaAgent**	6	click(720,847)×5 (right arrow); submit	correct
	UI-TARS	7	left, left, right, left, right, right, right	no submission
Bingo	**CaptchaAgent**	3	click(779,505); click(501,505); submit	correct
	UI-TARS	13	12×click toggling tiles across the grid; submit	wrong answer

J.2 Where **CaptchaAgent** Succeeds and UI-TARS Fails

Selection. We examine four types with different interaction requirements: Hold Button and Slide Puzzle from the real-time interaction mode, Rotation Match from arrow-cycle, and Bingo from multi-click. **CaptchaAgent** reaches 100.0, 43.9, 94.5, and 56.3 Pass@1 on these four types, respectively, while UI-TARS reaches 0.0, 2.5, 3.5, and 3.5. Hold Button additionally exposes an action-space limitation because UI-TARS does not provide a timed hold action.

For each type, we show one test puzzle on which the **CaptchaAgent** succeeds and UI-TARS fails. The examples are Hold Button image2301, Slide Puzzle slide2304, Rotation Match puzzle_rotation_105_225, and Bingo bingo2303. Figure 4 shows the initial page and the final state reached by each system before submission or stopping, and Table 14 lists the corresponding actions.

What the agents do. The four failures take different forms. On **Hold Button**, UI-TARS repeatedly clicks on and around the button but cannot issue the required hold action and reaches the step limit. On **Slide Puzzle**, it uses repeated short drags of only 6–18 pixels and stops before the piece reaches its slot, while **CaptchaAgent** completes the movement in two larger drags and submits. On **Rotation Match**, UI-TARS alternates left and right actions, stops one step from the target orientation, and terminates without submitting. The **CaptchaAgent** instead advances to the correct orientation and submits. On **Bingo**, UI-TARS repeatedly toggles tiles and ultimately submits an invalid swap, while the **CaptchaAgent** identifies and executes a valid two-tile swap.

These examples expose procedural and interface limitations rather than failures of localization alone: the agent must hold for a duration, drag toward a target, track changes across successive states, and complete a valid multi-step interaction before submission. These behaviors are directly represented in the replay-verified trajectories, where each intermediate screenshot and action is preserved. The comparison is in-distribution for **CaptchaAgent** and zero-shot for UI-TARS, so it shows the behavior of an unadapted general GUI agent rather than what UI-TARS could achieve after task-specific training. The external evaluations in App. I provide the corresponding out-of-distribution comparison.

K Supervised Fine-Tuning Configuration

Table 15 gives the full configuration for §4.2. LoRA adapters are applied only to the language model, while the vision encoder and vision–language aligner remain frozen. Visual inputs are therefore encoded using the pretrained vision components throughout training. We report the final checkpoint after the full training schedule rather than selecting the best epoch on the validation set.

Per-turn expansion converts the 12,000 training puzzles into 37,621 training samples, corresponding to an average of 3.1 supervised turns per puzzle.

Table 15: Supervised fine-tuning configuration.

Configuration	Value
Model	
Base model	Qwen3.5-9B
Adapters	rank-64 LoRA, $\alpha = 128$, dropout 0.05
LoRA target modules	q, k, v, o, gate, up, down
Frozen	vision encoder, vision–language aligner
Data	
Training puzzles	12,000 (600 per type)
Training samples	37,621
Maximum context length	32,768 tokens
Optimization	
Optimizer	AdamW
Learning rate	10^{-4}, cosine decay, 5% warmup
Weight decay	0.01
Gradient clipping	1.0
Epochs	3 (1,764 steps)
Effective batch size	64
Precision	bf16, ZeRO-3 with optimizer offload
Hardware	4× A100 40GB

L Reinforcement Learning Details

Task pool. We construct the GRPO training pool at the puzzle level rather than the task level. Candidate puzzles are evaluated with 5 rollouts under the supervised policy, and puzzles solved in some but not all rollouts are retained because they are more likely to produce within-group reward variance. Puzzles that are always solved or always failed provide little or no GRPO learning signal. Puzzle-level mining is important because a task-level average can hide both easy and difficult

instances within the same type. We therefore retain trainable puzzles from all 20 types rather than pruning entire types based on their mean success rate.

Coverage across all task types also limits drift. The KL penalty is evaluated only on sampled trajectories, so a type absent from the pool receives no direct KL constraint on its own trajectories while the shared model parameters are updated. The resulting GRPO training pool contains 2,643 puzzles across all 20 task types.

Reward. Let

$$p = \mathrm{clip}(\mathrm{progress}, 0, 1), \qquad f = \mathrm{clip}\left(\frac{\mathrm{turns}}{\mathrm{max\ turns}}, 0, 1\right),$$

where p is task-specific partial credit and f is the fraction of the turn budget used. The final reward is

$$r = \max\left(0,\ r_{\mathrm{raw}} - w_{\mathrm{turn}} f\right),$$

with

$$r_{\mathrm{raw}} = \begin{cases} 1, & \text{if submitted and correct,} \\ w_{\mathrm{prog}} p^{\gamma} + w_{\mathrm{sub}} \mathbf{1}[\mathrm{acted} \wedge p \geq \tau], & \text{if submitted and incorrect,} \\ w_{\mathrm{prog}} p^{\gamma}(1 - \lambda f), & \text{if not submitted.} \end{cases}$$

We use $w_{\mathrm{prog}} = 0.3$, $w_{\mathrm{sub}} = 0.05$, $\gamma = 1$, $\tau = 0.5$, $\lambda = 0.5$, and $w_{\mathrm{turn}} = 0.03$.

The reward addresses two shortcuts observed during training. First, an ungated submission bonus can encourage a cheap action followed immediately by submission. We therefore require sufficient progress before granting the submission bonus. Second, without a cost for long unsuccessful trajectories, the policy can continue probing until the turn limit instead of committing to an answer. The turn-dependent terms penalize this behavior. Flooring the final reward at zero prevents differences in episode length among zero-progress rollouts from creating artificial within-group variance.

Optimization and compute. Training is strictly on-policy. Each optimization step samples 8 puzzles with 16 rollouts per puzzle, giving 128 trajectories, and the same batch is used for the update. A step takes approximately 6–10 hours, dominated by multi-turn interaction with the browser environment. Table 16 gives the full configuration.

Checkpoint evaluation. After each training step, a quick evaluation on the validation split monitors training and shortlists checkpoints. Shortlisted checkpoints are then evaluated on all 200 validation puzzles per type with 5 rollouts per puzzle, using the same agent interface and verifier as the main evaluation. We select the checkpoint with the highest validation Pass@1 and evaluate it once on the test split. All reported RL test results come from this single test evaluation.

Table 16: Reinforcement learning configuration.

Configuration	**Value**
Model	
Base model	supervised checkpoint from §4.2
Trainable	full language-model weights
Frozen	vision encoder, vision–language aligner
Precision	bf16, FSDP with parameter and optimizer offload
Algorithm	
Advantage estimator	GRPO
Group size	16 rollouts per puzzle
Batch size	8 puzzles per step
Optimizer	AdamW
Learning rate	10^{-6}, constant, no warmup
Entropy coefficient	0
Clip ratio (high)	0.2
KL	0.005 against the frozen supervised reference
Epochs	1
Checkpoints	one per step
Rollout	
Engine	SGLang, asynchronous
Maximum prompt length	3,072 tokens
Maximum response length	12,288 tokens
Maximum context length	32,768 tokens
Temperature	1.0
Turn cap	15
Parallel calls	1
Memory fraction	0.38
Hardware	1×H100 and 3×RTX PRO 6000 (96 GB)