AHEAD: ATTENTION HEADS FOR ADVERSARIAL DETECTION FOR GUI AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Graphical-user-interface (GUI) agents rely on grounding models to translate screenshots and element descriptions into interaction coordinates, making grounding a point of exposure to visual adversarial attacks. Existing detectors require adversarial fitting examples or repeated inference, motivating detection using signals available during grounding inference. We observe that visual attacks alter how attention is distributed over screenshot regions during coordinate prediction, producing measurable changes in attention entropy. Motivated by this observation, we propose AHEAD (Attention HEads for Adversarial Detection), which jointly measures head-wise entropy deviations from a clean reference. AHEAD extracts its features during a single grounding inference and uses Mahalanobis distance with a threshold calibrated on clean inputs, requiring no adversarial fitting examples or additional model inferences. Across four grounding models, two interface datasets, and three attack settings, AHEAD achieves a mean AUROC of 0.997 and a mean true positive rate of 99.0% at a matched 3% false positive rate (TPR@3%FPR), with the lowest overhead compared to the baselines. AHEAD also transfers from mobile to web interfaces and vice versa using only 50 clean inputs from the new domain, achieving a mean 0.907 TPR@3%FPR in both directions. Overall, AHEAD provides a lightweight safeguard between visual perception and action execution, enabling intervention before unintended interactions and supporting safer deployment of GUI agents.

1 INTRODUCTION

GUI agents built on large vision-language models (LVLMs) interpret screenshots and interact with applications through clicking and typing. Their *GUI grounding models* map a screenshot and a natural-language description of an interface element to its screen coordinates (Cheng et al., 2024; Gou et al., 2025; Team et al., 2026; Company, 2025). This mapping connects visual understanding to action, as the predicted coordinates determine the location of a subsequent interaction. The screenshots used for grounding may contain attacker-controlled content. Prior studies show that visual modifications can induce incorrect coordinate predictions or redirect agent actions toward unintended directions (Zhao et al., 2025; Aichberger et al., 2026; Zhang et al., 2025b; Wang et al., 2025; Ye et al., 2025). For example, an agent instructed to select a back button may instead predict a location inside an advertisement. Detecting attacked screenshots during grounding enables intervention before compromised predictions lead to unintended actions, providing a practical safeguard for deploying GUI agents in environments with untrusted visual content.

Because GUI agents repeatedly perform grounding to achieve their goals, adversarial detection should add little latency to each grounding call. It should also avoid relying on attack-specific supervision, as adversarial examples collected for detector fitting may not represent the attacks encountered at deployment. NEARSIDE projects hidden states onto a direction learned from adversarial pairs, requiring adversarial fitting examples (Huang et al., 2024). JailGuard avoids this supervision by comparing responses to mutated inputs, but requires repeated inference and adaptation of its response comparison to coordinate outputs (Zhang et al., 2025a). These requirements motivate a detection signal that is available within a single model inference and can be modeled using clean inputs alone.

We propose **AHEAD** (Attention HEads for Adversarial Detection), which detects visual attacks using *answer-to-image attention*: the attention from coordinate-answer tokens to the screenshot's image

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.

tokens. Our analysis shows that visual attacks change the entropy of this attention differently across heads, motivating the joint use of these head-specific signals. We further find that effective detection does not require carefully selected heads, as long as enough heads are used.

Following these observations, AHEAD represents each input as a vector of attention entropies from a fixed random subset of heads. It estimates the mean and covariance of these vectors from clean inputs, then measures deviations from this clean reference using Mahalanobis distance (Mahalanobis, 2018). This accounts for both the variation in each head's entropy and the correlations across heads. The detection threshold is calibrated on a separate clean set. AHEAD extracts its features during the same grounding call that produces the coordinate answer, without additional model inferences. It requires neither adversarial fitting examples nor a clean counterpart of the evaluated screenshot.

We evaluate AHEAD across four grounding models, the mobile and web datasets AMEX (Chai et al., 2025) and Mind2Web (Deng et al., 2023), and three attack settings. AHEAD achieves a mean AUROC of 0.997 and a mean TPR@3%FPR of 0.990. It consistently matches or exceeds the baselines' TPR@3%FPR, while adding mean latency overheads of approximately 7% on AMEX and 9% on Mind2Web. AHEAD also transfers from mobile to web interfaces and vice versa using only 50 clean inputs from the new domain, achieving a mean TPR@3%FPR of 0.907 in both directions. We also evaluate detector-aware attacks where the attacker has white-box access to the grounding model. Together, these results show that AHEAD enables accurate detection of visual attacks with low inference overhead and transfers between mobile and web interfaces using limited clean data.

Contributions. Our contributions are summarized as follows:

- We empirically show that visual attacks induce head-specific changes in answer-to-image attention entropy during GUI grounding, providing a signal for adversarial detection.
- We develop AHEAD, which uses head-wise entropy deviations from a clean reference, enabling detection without adversarial fitting examples or additional model generations.
- Across four models, two domains, and three attacks, AHEAD achieves strong detection at low FPR with low overhead and transfers across domains using only 50 clean target inputs.

2 Related Work

GUI Agents and Adversarial Attacks. Advances in GUI understanding and visual grounding, exemplified by CogAgent (Hong et al., 2024), SeeClick (Cheng et al., 2024), and UGround (Gou et al., 2025), support increasingly capable screenshot-based GUI interaction. However, screenshot-based grounding relies on rendered content that may be attacker-controlled. Prior studies have demonstrated attacks that exploit this dependence. Zhao et al. (2025) evaluate targeted and untargeted image attacks against GUI grounding models, while MIP (Aichberger et al., 2026) and Chameleon (Zhang et al., 2025b) exploit localized visual content to redirect agent behavior. WebInject (Wang et al., 2025) further studies perturbations to rendered webpage pixels, accounting for the color changes introduced by the display's color profile when these pixels are captured as screenshots observed by the agent. These studies show that visual manipulation can corrupt grounding predictions or redirect agent behavior. Because predicted coordinates determine where a GUI agent interacts, such attacks can turn manipulated observations into unintended actions. Detecting adversarial inputs during grounding is therefore important for enabling intervention before those actions are executed.

Adversarial Detection. Existing LVLM detectors exploit signals such as internal representations and response consistency. HiddenDetect (Jiang et al., 2025a) monitors hidden states for jailbreak detection. NEARSIDE (Huang et al., 2024) projects hidden states onto a direction derived from clean–adversarial pairs, enabling efficient inference but requiring adversarial examples to construct the direction. JailGuard (Zhang et al., 2025a) measures response divergence across mutated inputs, requiring multiple model inferences for each grounding decision. Its semantic response comparison also requires adaptation to coordinate outputs, where disagreement concerns predicted locations. AHEAD addresses these constraints using answer-to-image attention entropy over the generated coordinate tokens. Detector fitting and threshold calibration require only clean inputs, and the detection signal is obtained during the ordinary grounding call, without additional model inference.



Figure 1: Three attack settings defined from prior GUI-agent attack studies. A *patch* attack manipulates an attacker-controlled screen region, a *redirect* attack applies a full-image perturbation toward a target, and a *disrupt* attack induces an erroneous coordinate without specifying a target.

3 Threat Model

Setting and access. We study visual attacks at the grounding stage of GUI agents, which predict a coordinate string C from a screenshot I and an element description D. The attacker has white-box access to the grounding model, including its parameters and gradients, but can modify only I. We consider three attack settings with different editable regions and objectives (Figure 1).

Targeted attacks minimize $\mathcal{L}_{\text{coord}}(I, D; C^*)$, the cross-entropy loss for an attacker-specified coordinate string C^*. Pixel values lie in $[0, 1]$. For full-image attacks, the attacked screenshot is $\mathcal{A}(I; \delta) = I + \delta$, where $\delta \in \Delta_\epsilon(I) = \{\delta : \|\delta\|_\infty \le \epsilon,\ 0 \le I + \delta \le 1\}$.

`patch`: localized and targeted. The attacker optimizes a single banner to induce predictions at its center across different screenshot backgrounds and insertion locations (Aichberger et al., 2026; Zhang et al., 2025b). During optimization, the banner P is inserted at randomly sampled positions. A binary mask M marks the insertion region, and $C^*(M)$ denotes the coordinate string of its center. We denote the resulting screenshot by $\mathcal{A}(I; P, M)$, which leaves all pixels outside the region unchanged.

The banner is constrained relative to its original image P_0: $\mathcal{P}_\epsilon(P_0) = \{P : \|P - P_0\|_\infty \le \epsilon,\ 0 \le P \le 1\}$. The objective averages over inputs (I, D) from a separate optimization split $\mathcal{D}_{\text{opt}}$ and their randomly sampled placements M:

$$P^* = \arg \min_{P \in \mathcal{P}_\epsilon(P_0)} \mathbb{E}_{(I,D),M} \left[\mathcal{L}_{\text{coord}}(\mathcal{A}(I; P, M), D; C^*(M)) \right]. \tag{1}$$

`redirect`: full-image and targeted. The attacker targets C_R^*, the coordinate string for the center of another interface element (Zhao et al., 2025; Wang et al., 2025):

$$\delta_R^* = \arg \min_{\delta \in \Delta_\epsilon(I)} \mathcal{L}_{\text{coord}}(\mathcal{A}(I; \delta), D; C_R^*). \tag{2}$$

`disrupt`: full-image and untargeted. Let $\mathbf{z}_{\text{vis}}(I)$ denote the grounding model's visual-encoder feature vector for I. The attacker maximizes the distance between clean and perturbed visual features as a surrogate for grounding failure, without specifying a target coordinate (Zhao et al., 2025; Wang et al., 2025):

$$\delta_D^* = \arg \max_{\delta \in \Delta_\epsilon(I)} \left\| \mathbf{z}_{\text{vis}}(\mathcal{A}(I; \delta)) - \mathbf{z}_{\text{vis}}(I) \right\|_2^2. \tag{3}$$

Defender assumptions. The defender has access to a trusted set of clean grounding inputs. These clean inputs are used to fit the reference distribution and calibrate the detection threshold. The defender does not require prior knowledge of the attack type or access to adversarial examples.

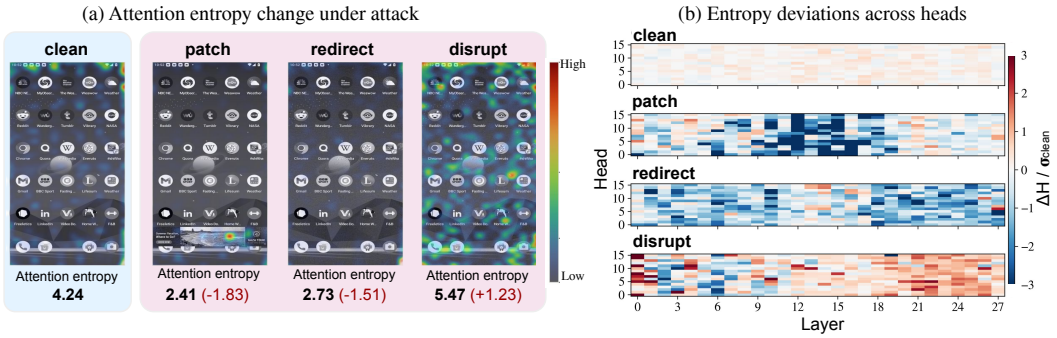

Figure 2: **Visual attacks produce entropy deviations across many heads.** (a) Attention maps and entropy values for head `L12H11` on one screenshot. (b) Each cell represents a fixed layer–head pair and shows the median of $(H_j(x) - c_j)/\sigma_j$ over the same 130 screenshots under each condition. Here, c_j and σ_j are the clean entropy median and standard deviation, estimated from 1,000 separate clean fitting inputs. Blue and red indicate lower and higher entropy relative to this reference, respectively.

4 Observations

Attention from generated tokens to textual or visual input tokens has been investigated in studies of language and vision-language models (Jiang et al., 2025b; Chuang et al., 2024). Motivated by these findings, we examine whether visual attacks alter how coordinate-answer tokens attend to screenshot tokens. We first examine how visual attacks change answer-to-image attention (Section 4.1), then analyze how these attention changes vary across attention heads (Section 4.2), and finally assess how the number and choices of attention heads for detection affect performance (Section 4.3). All analyses use UI-Venus-1.5-2B (Team et al., 2026) on the Samsung S10 subset of AMEX (Chai et al., 2025) with the three attacks introduced in Section 3.

Analysis target: answer-to-image attention. Given an input $x = (I, D)$ consisting of a screenshot I and an element description D, the GUI agent generates a coordinate answer C. We examine the model's self-attention from answer-token positions to the screenshot's image tokens, which we call *answer-to-image attention*. Each clean or attacked input is analyzed using its own generated answer. Let $j = (\ell, h)$ identify head h in layer ℓ, and let $A^j(x) \in \mathbb{R}^{T \times N_v}$ contain its attention weights from T answer positions to N_v image tokens. The entry $A^j_{t,i}(x)$ denotes the attention weight from answer position t to image token i. The answer span ends at the final numeric token of the coordinate answer.

4.1 Visual attacks change attention entropy

Figure 2(a) illustrates how visual attacks change the concentration of answer-to-image attention. We quantify this concentration using Shannon entropy. Averaging attention over answer positions and renormalizing over image tokens gives the normalized weights $p_{j,i}(x)$ and entropy $H_j(x)$:

$$p_{j,i}(x) = \frac{\sum_{t=1}^{T} A^j_{t,i}(x)}{\sum_{u=1}^{N_v} \sum_{t=1}^{T} A^j_{t,u}(x)}, \qquad H_j(x) = -\sum_{i=1}^{N_v} p_{j,i}(x) \log p_{j,i}(x). \tag{4}$$

Lower entropy indicates more concentrated attention, while higher entropy indicates more diffuse attention. For the illustrated head in Figure 2(a), `patch` and `redirect` attacks concentrate attention and reduce entropy, whereas `disrupt` disperses attention and increases entropy compared with the clean screenshot.

To assess whether these changes extend beyond a single attention head, Figure 2(b) summarizes entropy deviations across all 448 heads over the same 130 screenshots under each condition. For visualization, each head's entropy is centered at its clean median and scaled by its clean standard deviation. Across the three attacks, 143–217 heads exhibit absolute median deviations of at least one clean standard deviation, compared with none under the clean condition. These results show that visual attacks change attention entropy across many heads and inputs. These findings motivate our use of answer-to-image attention entropy as a detection signal in AHEAD.

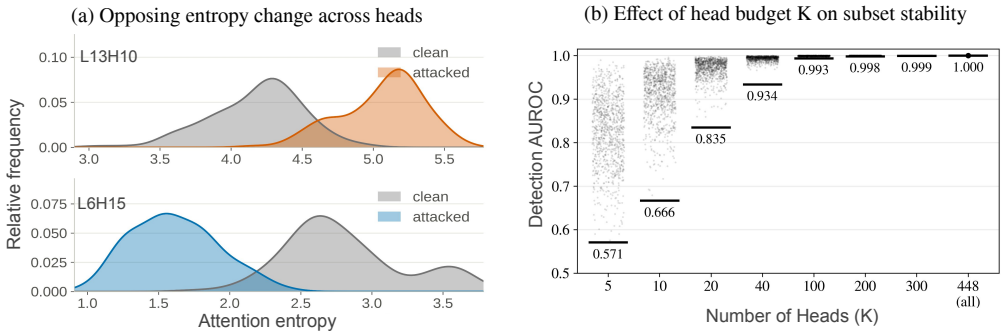

Figure 3: **Heads respond in opposite directions, while larger head sets reduce selection sensitivity.** (a) Under `disrupt`, the entropy distribution shifts upward for `L13H10` and downward for `L6H15`. (b) Each point is the AUROC averaged over three attacks for one random subset of K heads. We sample 1,000 subsets per K; $K = 448$ uses all heads and has one subset. Bars and numbers mark the minimum observed AUROC.

4.2 Entropy changes differ across attention heads

To use changes in answer-to-image attention entropy for detection, we next examine how the direction of entropy change varies across heads. Figure 2(b) shows that entropy can increase in some heads and decrease in others under the same attack. Under `disrupt`, for example, median entropy is higher than the clean reference in 99 heads and lower in 44 heads, with deviations of at least one clean standard deviation in both cases. Thus, substantial entropy changes occur in both directions within a single attack setting. Figure 3(a) examines these opposing responses in more detail by comparing the clean and attacked entropy distributions of two fixed heads. Under `disrupt`, the distribution shifts toward higher values for `L13H10` and toward lower values for `L6H15`.

The observation motivates using each head's entropy as a separate feature. AHEAD measures how far the vector of head entropies deviates from the distribution observed on clean inputs, allowing both increases and decreases to contribute to attack detection.

4.3 Larger head sets lead to more stable detection

With individual head entropies as features, we next ask how detection depends on the number and choice of heads. For each random subset of K heads, we form a vector of head entropies and estimate its mean and covariance from clean inputs. We use a regularized Mahalanobis distance (Mahalanobis, 2018) to measure how far each input's entropy vector deviates from the clean reference, accounting for variation and correlations among head entropies. The fitting and evaluation inputs are held fixed across subsets, so the comparison isolates the effect of head selection. Figure 3(b) reports the resulting AUROC averaged over the three attacks.

Small head subsets exhibit substantial variation: at $K = 10$, the mean attack-averaged AUROC across sampled subsets is 0.913, while the minimum is 0.666. As K increases, the spread narrows and the minimum observed AUROC rises. At $K = 300$, the minimum reaches 0.999. These results suggest that larger random head subsets achieve stronger and more stable detection without careful head selection. Since a single grounding inference provides attention weights for all heads, using more heads improves detection performance at little additional computational cost.

We summarize the main observations as follows:

1. *Visual attacks change answer-to-image attention entropy*: Entropy deviations from the clean reference appear across heads and inputs, providing a signal for attack detection.
2. *Entropy changes differ across attention heads*: The same attack can increase entropy in some heads and decrease entropy in others, motivating a separate entropy feature for each head.
3. *Larger head sets reduce sensitivity to head selection*: Increasing K raises the observed minimum AUROC while reducing its variance across sampled subsets.

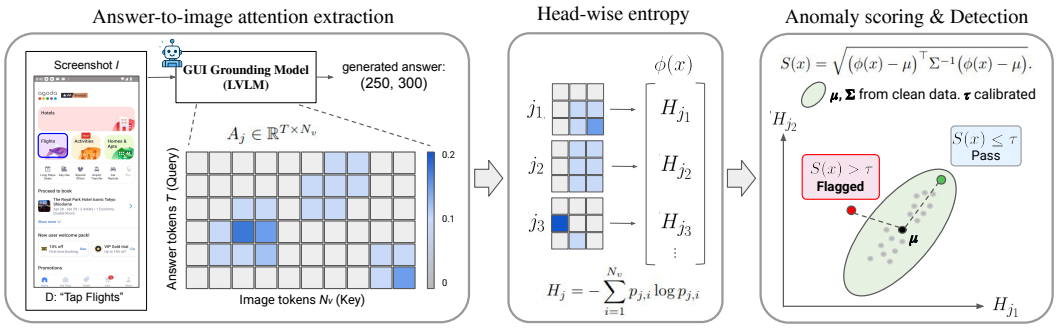

Figure 4: **Overview of AHEAD.** AHEAD extracts attention from generated answer tokens to image tokens (left), summarizes a randomly chosen subset of heads as an entropy vector (middle), and measures its deviation from a clean reference using Mahalanobis distance (right). The reference (μ, Σ) and threshold τ are estimated on clean fitting and calibration sets, respectively. Inputs with $S(x) > \tau$ are flagged as adversarial.

5 METHOD

Building on the three main observations summarized in Section 4, we introduce AHEAD, which detects visual attacks by jointly measuring head-wise entropy deviations from a clean reference. (Figure 4). AHEAD represents each input as a vector of answer-to-image attention entropies from a fixed random subset of K heads. It estimates a reference mean and covariance from clean entropy vectors and calibrates a detection threshold on a separate clean set. At deployment, AHEAD computes this vector from the GUI agent's generated coordinate answer and flags the input as adversarial when its Mahalanobis distance from the clean reference exceeds the threshold.

5.1 CLEAN REFERENCE CONSTRUCTION

We sample K attention heads uniformly without replacement and use the resulting subset $\mathcal{S}$ in a fixed order throughout fitting, calibration, and deployment. For an input $x = (I, D)$, we construct the entropy vector

$$\phi(x) = \big[H_j(x)\big]_{j \in \mathcal{S}} \in \mathbb{R}^K, \tag{5}$$

where $H_j(x)$ is the attention entropy defined in Equation 4. For each model–dataset pair, we compute the sample mean μ and regularized covariance Σ of these vectors on a clean fitting set $\mathcal{D}_{\text{fit}}$. The mean captures each head's typical entropy, while the covariance captures how the entropies vary individually and together across clean inputs.

5.2 ANOMALY SCORING AND DETECTION

Given the clean reference, we measure the deviation of an input's entropy vector using the regularized Mahalanobis distance:

$$S(x) = \sqrt{\big(\phi(x) - \mu\big)^\top \Sigma^{-1} \big(\phi(x) - \mu\big)}. \tag{6}$$

This distance accounts for both the variability of individual heads and the correlations between them. Unusual combinations of entropies can therefore receive high scores even when individual values remain within their typical ranges.

Before deployment, we calibrate the detection threshold on a clean set $\mathcal{D}_{\text{cal}}$ disjoint from $\mathcal{D}_{\text{fit}}$. For a target false positive rate α, we set

$$\tau = Q_{1-\alpha}(\{S(x) : x \in \mathcal{D}_{\text{cal}}\}), \tag{7}$$

where $Q_{1-\alpha}$ denotes the empirical $(1-\alpha)$-quantile. For a new input, AHEAD computes $\phi(x)$ from attention directed from the GUI agent's generated answer tokens to the image tokens. It evaluates $S(x)$ using the fixed reference and flags the input as adversarial when $S(x) > \tau$, allowing the GUI agent to take appropriate action before executing the predicted interaction.

Table 1: **Detection across agents and attacks.** Full-image attacks use $\epsilon = 16/255$. TPR uses matched 3% FPR on mixed clean inputs. Latency is normalized by the same agent's grounding latency without detection. Bold/underlined values mark the best/second-best results, including ties. Red., Dis., and Pat. denote `redirect`, `disrupt`, and `patch`.

		AMEX							*Mind2Web*						
		TPR@3%FPR			AUROC			Latency	TPR@3%FPR			AUROC			Latency
Model	Method	Red.	Dis.	Pat.	Red.	Dis.	Pat.	×agent	Red.	Dis.	Pat.	Red.	Dis.	Pat.	×agent
UI-Venus-1.5-2B	NEARSIDE†	0.990	0.217	0.633	0.997	0.924	0.945	1.39	0.973	0.243	0.930	0.993	0.958	0.965	1.46
	JailGuard	0.463	0.097	0.123	0.935	0.829	0.894	7.05	0.397	0.030	0.133	0.926	0.728	0.907	7.05
	AHEAD	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.11**	**1.000**	**0.993**	**1.000**	**1.000**	**0.999**	**1.000**	**1.10**
UGround-V1-2B	NEARSIDE†	0.710	0.987	0.663	0.949	0.996	0.905	1.10	0.890	**1.000**	0.307	0.983	**1.000**	0.744	1.11
	JailGuard	0.020	0.043	0.040	0.797	0.756	0.792	7.00	0.043	0.030	0.063	0.868	0.818	0.871	7.01
	AHEAD	**1.000**	**1.000**	**0.990**	**0.997**	**1.000**	**0.990**	**1.01**	**1.000**	**1.000**	**0.963**	**1.000**	**1.000**	**0.990**	**1.01**
Holo2-4B	NEARSIDE†	0.733	0.117	0.950	0.845	0.868	0.983	1.08	**1.000**	0.103	**1.000**	**1.000**	0.822	**1.000**	1.34
	JailGuard	0.467	0.137	0.117	0.868	0.693	0.845	6.78	0.540	0.070	0.240	0.929	0.749	0.913	7.06
	AHEAD	**1.000**	**0.997**	**1.000**	**1.000**	**0.996**	**1.000**	**1.02**	**1.000**	**0.987**	**1.000**	**1.000**	**0.995**	**1.000**	**1.12**
KV-Ground-4B	NEARSIDE†	0.983	0.923	0.950	**0.998**	0.988	0.991	1.23	0.957	0.507	0.280	0.992	0.802	0.911	1.24
	JailGuard	0.457	0.223	0.107	0.912	0.848	0.875	7.02	0.450	0.133	0.160	0.916	0.791	0.897	7.03
	AHEAD	**0.997**	**1.000**	**1.000**	0.995	**0.999**	**0.998**	**1.14**	**1.000**	**0.977**	**0.857**	**1.000**	**0.994**	**0.982**	**1.13**
Mean	NEARSIDE†	0.854	0.561	0.799	0.947	0.944	0.956	1.20	0.955	0.463	0.629	0.992	0.896	0.905	1.29
	JailGuard	0.352	0.125	0.097	0.878	0.782	0.852	6.96	0.357	0.066	0.149	0.910	0.771	0.897	7.04
	AHEAD	**0.999**	**0.999**	**0.998**	**0.998**	**0.999**	**0.997**	**1.07**	**1.000**	**0.989**	**0.955**	**1.000**	**0.997**	**0.993**	**1.09**

† NEARSIDE uses attack-specific supervision; AHEAD and JailGuard use no adversarial fitting data.

6 Experiments

6.1 Setup

Models, attacks, and data. We evaluate four GUI grounding models: UI-Venus-1.5-2B (Team et al., 2026), UGround-V1-2B (Gou et al., 2025), Holo2-4B (Company, 2025), and KV-Ground-4B (Kingsware & AI, 2026). Experiments use AMEX (Chai et al., 2025) for mobile interfaces and Mind2Web (Deng et al., 2023) for web interfaces. For each model–dataset pair, we evaluate the three attacks described in Section 3: `redirect`, `disrupt`, and `patch`, yielding 24 model–dataset–attack configurations. Full-image perturbations use $\epsilon = 16/255$. We evaluate 300 successful attacks per configuration. For each model–dataset pair, AHEAD uses disjoint sets of 1,000 clean inputs for fitting ($\mathcal{D}_{\text{fit}}$) and 250 for calibration ($\mathcal{D}_{\text{cal}}$). All detectors share the same 300 held-out clean evaluation inputs. Clean inputs are sampled without filtering by grounding correctness, retaining natural grounding failures (16–31%). AHEAD uses a fixed random subset of $K = 300$ attention heads.

Baselines and metrics. NEARSIDE (Huang et al., 2024) learns an attack-specific direction from the final-token hidden states of 150 clean and 150 adversarial inputs, and uses 150 additional clean inputs for calibration. JailGuard (Zhang et al., 2025a) compares predictions from six mutated screenshots and uses 250 clean inputs to calibrate its coordinate-similarity bandwidth and detection threshold. AHEAD and JailGuard require no adversarial fitting data. We report AUROC and the true positive rate at a matched 3% false positive rate (TPR@3%FPR). We also report TPR@τ and FPR@τ at a threshold τ fixed using independent clean calibration data. Appendix A details the protocol.

6.2 Detection performance

AHEAD achieves stronger detection with lower inference overhead. Without adversarial fitting examples, AHEAD achieves a mean AUROC of 0.997 and TPR@3%FPR of 0.990 across the 24 configurations (Table 1). NEARSIDE's performance varies across attack types, suggesting that its learned hidden-state direction does not provide equally strong separation for all attacks. JailGuard's moderate AUROC translates into substantially lower TPR@3%FPR, indicating that overall ranking performance does not ensure effective detection when false positives must remain low. The gap is particularly pronounced for `disrupt`, where both baselines lose sensitivity while AHEAD maintains high detection rates. These results show that AHEAD achieves consistently stronger separation at low FPR. The mean latency overhead is approximately 7% on AMEX and 9% on Mind2Web, the lowest among the evaluated detectors.

Detection with fixed thresholds. On the same evaluation inputs as Table 1, we apply each detector's threshold τ determined using separate clean calibration data. At their respective calibrated thresholds, AHEAD achieves the highest mean TPR: 0.997 on AMEX and 0.927 on Mind2Web, with mean FPRs of 0.037 and 0.017, respectively (Table 2). These results show that AHEAD maintains strong detection with thresholds fixed before evaluation.

Table 2: **Detection with thresholds set on clean data.**

	AMEX		Mind2Web	
Method	TPR@τ	FPR@τ	TPR@τ	FPR@τ
NEARSIDE	0.758	0.034	0.717	0.037
JailGuard	0.181	0.032	0.228	0.036
AHEAD	**0.997**	0.037	**0.927**	0.017

TPR averages over models and attacks; FPR averages over models and, for NEARSIDE, attack-specific detectors.

6.3 Cross-domain transfer

Table 3: **Cross-domain detection.** N counts clean target inputs used for adaptation and calibration. TPR is measured at a matched 3% FPR; FPR is measured at the clean-calibrated threshold τ.

		AMEX → Mind2Web			Mind2Web → AMEX		
Method	N	AUROC	TPR@3%FPR	FPR@τ	AUROC	TPR@3%FPR	FPR@τ
JailGuard	0	0.861	0.201	0.021	0.837	0.179	0.062
JailGuard	50	0.859	0.184	0.052	0.837	0.190	0.044
NEARSIDE†	0	0.759	0.425	0.151	0.804	0.447	0.221
NEARSIDE†	50	0.759	0.425	0.046	0.804	0.447	0.049
AHEAD	0	0.892	0.652	1.000	0.689	0.432	1.000
AHEAD	**50**	**0.990**	**0.907**	0.020	**0.988**	**0.907**	0.032

†NEARSIDE retains attack-specific directions learned from source clean–adversarial pairs.

Transfer setup. We evaluate transfer between AMEX and Mind2Web in both directions, comparing source-only detectors with adaptation using a total budget of $N = 50$ clean target inputs per method. AHEAD keeps its source head subset fixed and updates its mean, covariance, and threshold, using the source covariance for regularization and cross-fitting for calibration. NEARSIDE retains its source-trained attack direction and recalibrates its threshold; JailGuard re-estimates its bandwidth and threshold. All methods share the same target evaluation inputs. Results average over four GUI agents and 100 target-sample draws. AUROC and TPR average over three attacks. Appendix E details the method-specific sampling pools and adaptation procedures.

AHEAD recovers detection with few clean target inputs. The contrast in Table 3 reflects what each detector models. NEARSIDE's attack directions shift across domains, and relearning them under its original formulation requires target adversarial examples. Clean-only threshold recalibration therefore leaves its score rankings unchanged. JailGuard's response-divergence scores overlap between clean and attacked inputs, even among correctly grounded clean examples; bandwidth adaptation does little to improve separation at low FPR. AHEAD's clean attention-entropy statistics also change across domains, but target clean inputs directly support updating its mean and covariance.

6.4 Adaptive attacks

Adaptive attack setup. We evaluate on AMEX using KV-Ground-4B, which has the highest clean grounding accuracy among our evaluated models. The attacker has white-box access to the grounding model and knows AHEAD's algorithm, but not the fixed head subset chosen at deployment. This models a setting where model weights are public but the deployed detector configuration remains private. Without knowledge of the deployed subset, the attacker minimizes the expected detection penalty, which we approximate via expectation-over-transformation, averaging over $M = 4$ surrogate AHEAD detectors, each using $K = 300$ randomly sampled heads. For an attacked input x', the attacker minimizes

$$\mathcal{L}_{\text{adaptive}} = \mathcal{L}_{\text{attack}} + \frac{\lambda}{M} \sum_{m=1}^{M} \left[\max\left(0, \frac{s_m(x')}{\tau_m} - 1\right) \right]^2, \tag{8}$$

where s_m and τ_m are the surrogate's score and threshold. For each attack, we report the evaluated λ that maximizes end-to-end ASR. Details are in Appendix F.

Table 4: **Adaptive attack performance** AS: attack success; Ev.: detector evasion; ASR: successful attack and evasion. All rates are percentages. Ev. and ASR are medians across 20 detector draws.

Attack	Setting	AS	Ev.	**ASR**
`redirect`	Non-adaptive	96.0	1.0	0.0
	Adaptive	91.0	13.5	**11.0**
`disrupt`	Non-adaptive	22.0	0.0	0.0
	Adaptive	5.0	32.5	**1.5**
`patch`	Non-adaptive	82.0	1.0	0.0
	Adaptive	56.0	67.0	**27.5**

Attack success and detector evasion. At the evaluated weights that maximize median end-to-end ASR, adaptive optimization increases evasion while reducing attack success relative to the non-adaptive controls (Table 4). For `patch`, evasion rises from 1.0% to 67.0%, while attack success falls from 82.0% to 56.0%. Median end-to-end ASR reaches 11.0%, 1.5%, and 27.5% for `redirect`, `disrupt`, and `patch`, respectively. This trade-off holds at the reported settings, but not uniformly across the full weight sweep (Appendix F).

6.5 Ablation

Table 5: **Feature ablation.** All variants use 300 dimensions and full Mahalanobis scoring. Entropy and attention mass use identical head subsets. Hidden states are averaged over answer tokens before randomly selecting 300 dimensions. TPR@3%FPR is averaged over 24 configurations (mean $\pm$ SD across 10 draws).

Feature	TPR@3%FPR
Hidden states (last layer)	0.702 ± 0.008
Hidden states (all layers)	0.896 ± 0.009
Attention mass	0.948 ± 0.003
Attention entropy (AHEAD)	$\mathbf{0.989} \pm 0.005$

Table 6: **Scoring ablation.** All variants use entropies from the same 300 heads. Mean entropy is scored by its absolute deviation from the clean mean, divided by the clean standard deviation. TPR@3%FPR is averaged over 24 configurations (mean $\pm$ SD across 10 draws).

Scoring rule	TPR@3%FPR
Mean entropy	0.350 ± 0.016
Euclidean	0.673 ± 0.022
Diagonal Mahalanobis	0.760 ± 0.015
Full Mahalanobis (AHEAD)	$\mathbf{0.989} \pm 0.005$

Feature and scoring ablation. Head-wise attention entropy achieves a mean TPR@3%FPR of 0.989, compared with 0.948 for attention mass and 0.702/0.896 for last-layer/all-layer hidden states (Table 5). Attention mass measures the total attention weight assigned to image tokens, averaged over answer tokens. These results motivate AHEAD's use of head-wise attention entropy. Preserving individual head entropies and applying Euclidean distance raises TPR@3%FPR from 0.350 for mean entropy to 0.673; diagonal and full Mahalanobis scoring further increase it to 0.760 and 0.989, respectively (Table 6). These results support AHEAD's joint scoring of head-wise entropy deviations using the full clean covariance. Additional comparisons are provided in Appendix G.

7 Limitations

AHEAD has three limitations. First, it requires access to internal attention weights, limiting its direct applicability to black-box APIs that expose only model outputs. Second, our evaluation focuses on individual grounding decisions rather than complete agent trajectories. Its effects on end-to-end task completion and the accumulation of false positives over repeated interactions therefore remain unexplored. Third, AHEAD provides a detection signal but does not include a post-detection intervention or recovery policy. Future work should integrate AHEAD with intervention and recovery mechanisms and evaluate their effects on both security and task utility over complete agent trajectories.

8 Conclusion

We presented AHEAD, a visual adversarial detector for GUI agents that uses attention that can be extracted during model inference. By jointly evaluating head-wise attention entropies against clean behavior, AHEAD provides a detection signal without adversarial supervision or additional model inference. Experiments across mobile and web interfaces demonstrate strong detection performance, modest runtime overhead, and adaptation using limited clean target data. These findings show that the grounding process itself provides useful evidence of visual manipulation, offering a practical means of screening inputs before an agent executes an action.

AI USE STATEMENT

In this work, we used generative AI tools to provide feedback on experimental design and assist with the interpretation of results. Additionally, we used ChatGPT for manuscript drafting, language editing, and LaTeX formatting. We have reviewed all AI-assisted work, checking the resulting text for consistency with our methodology and experimental results. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work aims to improve the security of GUI agents by detecting adversarially manipulated screenshots before the predicted interactions are executed. Our experiments use the existing AMEX (Chai et al., 2025) and Mind2Web (Deng et al., 2023) datasets and pretrained GUI grounding models (Gou et al., 2025; Team et al., 2026; Kingsware & AI, 2026; Company, 2025). Although the attacks studied in this work could be misused to manipulate agent behavior, our objective is to understand these vulnerabilities and evaluate a detection mechanism. Practical deployment should account for both undetected attacks and false positives that may interrupt legitimate tasks.

REPRODUCIBILITY STATEMENT

We describe AHEAD's feature extraction, clean reference construction, and detection rule in Section 5, and the experimental setup and evaluation metrics in Section 6. Appendix A specifies the model checkpoints, input preprocessing, data splits, attack configurations, and baseline implementations. Appendices B and C provide additional observation and detection results. Appendices D, E, F, and G describe the latency measurements, cross-domain adaptation procedure, adaptive attack configurations, and feature and scoring ablations, respectively. These details document the procedures used to obtain the reported results.

REFERENCES

Lukas Aichberger, Alasdair Paren, Guohao Li, Philip Torr, Yarin Gal, and Adel Bibi. Mip against agent: Malicious image patches hijacking multimodal os agents. *Advances in Neural Information Processing Systems*, 38:18536–18575, 2026.

Yuxiang Chai, Siyuan Huang, Yazhe Niu, Han Xiao, Liang Liu, Guozhi Wang, Dingyu Zhang, Shuai Ren, and Hongsheng Li. Amex: Android multi-annotation expo dataset for mobile gui agents. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 2138–2156, 2025.

Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Li YanTao, Jianbing Zhang, and Zhiyong Wu. Seeclick: Harnessing gui grounding for advanced visual gui agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9313–9332, 2024.

Yung-Sung Chuang, Linlu Qiu, Cheng-Yu Hsieh, Ranjay Krishna, Yoon Kim, and James Glass. Lookback lens: Detecting and mitigating contextual hallucinations in large language models using only attention maps. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 1419–1436, 2024.

H Company. Holo2 - open foundation models for navigation and computer use agents, 2025. URL `https://huggingface.co/collections/Hcompany/holo2`.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36:28091–28114, 2023.

Boyu Gou, Demi Ruohan Wang, Boyuan Zheng, Yanan Xie, Cheng Chang, Yiheng Shu, Huan Sun, and Yu Su. Navigating the digital world as humans do: Universal visual grounding for gui agents. In *International Conference on Learning Representations*, volume 2025, pp. 30851–30883, 2025.

Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxiao Dong, Ming Ding, et al. Cogagent: A visual language model for gui agents. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14281–14290. IEEE, 2024.

Youcheng Huang, Fengbin Zhu, Jingkun Tang, Pan Zhou, Wenqiang Lei, Jiancheng Lv, and Tat-Seng Chua. Effective and efficient adversarial detection for vision-language models via a single vector. *arXiv preprint arXiv:2410.22888*, 2024.

Yilei Jiang, Xinyan Gao, Tianshuo Peng, Yingshui Tan, Xiaoyong Zhu, Bo Zheng, and Xiangyu Yue. Hiddendetect: Detecting jailbreak attacks against multimodal large language models via monitoring hidden states. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14880–14893, 2025a.

Zhangqi Jiang, Junkai Chen, Beier Zhu, Tingjin Luo, Yankun Shen, and Xu Yang. Devils in middle layers of large vision-language models: Interpreting, detecting and mitigating object hallucinations via attention lens. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 25004–25014. IEEE, 2025b.

Kingsware and Vocaela AI. Kv-ground: Small gui grounding models for high-resolution images, 2026. URL `https://huggingface.co/vocaela/KV-Ground-4B-BaseGuiOwl1.5-0228`.

Olivier Ledoit and Michael Wolf. A well-conditioned estimator for large-dimensional covariance matrices. *Journal of multivariate analysis*, 88(2):365–411, 2004.

Prasanta Chandra Mahalanobis. On the generalized distance in statistics. *Sankhyā: The Indian Journal of Statistics, Series A (2008-)*, 80:S1–S7, 2018.

Venus Team, Changlong Gao, Zhangxuan Gu, Yulin Liu, Xinyu Qiu, Shuheng Shen, Yue Wen, Tianyu Xia, Zhenyu Xu, Zhengwen Zeng, et al. Ui-venus-1.5 technical report. *arXiv preprint arXiv:2602.09082*, 2026.

Xilong Wang, John Bloch, Zedian Shao, Yuepeng Hu, Shuyan Zhou, and Neil Zhenqiang Gong. Webinject: Prompt injection attack to web agents. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 2010–2030, 2025.

Ziang Ye, Yang Zhang, Wentao Shi, Xiaoyu You, Fuli Feng, and Tat-Seng Chua. Visualtrap: A stealthy backdoor attack on gui agents via visual grounding manipulation. *arXiv preprint arXiv:2507.06899*, 2025.

Xiaoyu Zhang, Cen Zhang, Tianlin Li, Yihao Huang, Xiaojun Jia, Ming Hu, Jie Zhang, Yang Liu, Shiqing Ma, and Chao Shen. Jailguard: A universal detection framework for prompt-based attacks on llm systems. *ACM Transactions on Software Engineering and Methodology*, 35(1):1–40, 2025a.

Yitong Zhang, Ximo Li, Liyi Cai, and Jia Li. Environmental injection attacks against gui agents in realistic dynamic environments. *arXiv preprint arXiv:2509.11250*, 2025b.

Haoren Zhao, Tianyi Chen, and Zhen Wang. On the robustness of gui grounding models against image attacks. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 1609–1614. IEEE, 2025.

APPENDIX CONTENTS

A Experimental protocol and implementation details 13

B Observation details 15

C Additional detection results 16

D Latency measurement details 18

E Cross-domain transfer details 19

F Adaptive attack details 22

G Feature and scoring ablation details 24

A EXPERIMENTAL PROTOCOL AND IMPLEMENTATION DETAILS

Models and datasets. Table 7 lists the model checkpoints, attention head counts, and input resolutions used in our evaluation. We use a common budget of $K = 300$ randomly sampled heads for all models. For AMEX (Chai et al., 2025), we use the Samsung S10 subset with screenshots at 1440×3040 pixels. For Mind2Web (Deng et al., 2023), we use the multimodal release and crop each screenshot to the top 1280×1080 browser viewport. Each model processes these screenshots at the input resolution reported in Table 7, which determines the number of image tokens N_v for each model–dataset pair. The answer span starts after the assistant header or reasoning trace, or at the coordinate marker for tool-call outputs, depending on the model's output format. It ends at the final numeric token of the predicted coordinate.

Table 7: **Models, checkpoints, and input resolutions.** Head counts are total/usable across decoder layers; usable heads are available in every record. Resolutions are width×height in pixels, with N_v in parentheses. Each checkpoint appears below its model.

Model	Heads	AMEX (N_v)	Mind2Web (N_v)
UI-Venus-1.5-2B `inclusionAI/UI-Venus-1.5-2B`	448/448	608×1344 (798)	992×832 (806)
UGround-V1-2B `osunlp/UGround-V1-2B`	336/333	644×1400 (1150)	1036×868 (1147)
Holo2-4B `Hcompany/Holo2-4B`	1152/1152	608×1344 (798)	992×832 (806)
KV-Ground-4B `vocaela/KV-Ground-4B-BaseGuiOwl1.5-0228`	1152/1152	608×1344 (798)	992×832 (806)

Data splits. For each model–dataset pair, the clean fitting, calibration, and evaluation sets contain 1,000, 250, and 300 inputs, respectively. These sets are pairwise disjoint and do not overlap with the attack-generation pool. Clean inputs are sampled without filtering by grounding correctness; Table 8 reports the resulting grounding failures. The same clean evaluation inputs are used across all three attacks and detectors. Within each model–dataset–attack configuration, all detectors also share 300 successful attacks, selected from 1,000 attack attempts on initially correct inputs. Detection rates are therefore conditional on attack success and should not be interpreted as attack success rates.

Table 8: **Natural grounding failures in the clean sets.** Entries count incorrectly grounded inputs. Fit, Cal., and Eval. contain 1,000, 250, and 300 inputs, respectively.

	AMEX			Mind2Web		
Model	Fit	Cal.	Eval.	Fit	Cal.	Eval.
UI-Venus-1.5-2B	188	47	57	160	40	48
UGround-V1-2B	307	77	92	287	71	86
Holo2-4B	195	49	59	175	44	52
KV-Ground-4B	160	40	48	157	39	47

Attack implementation. Table 9 summarizes the optimization settings and success criteria. Full-image attacks apply uint8 quantization at every optimization step, using a straight-through estimator for differentiation. For `redirect`, the target C_R^* is fixed within each model–dataset pair at the center of a top-left square covering 0.04% of the input frame. The universal `patch` is a full-width banner initialized from a natural image. At evaluation, banners are placed without overlapping the intended interface element.

Baseline implementations. **NEARSIDE** (Huang et al., 2024) projects the last-layer hidden state of the final answer token onto a direction defined by the difference between the mean attacked and clean hidden states. We fit a separate direction for each attack using 150 attacked and 150 clean inputs,

Table 9: **Attack settings at $\epsilon = 16/255$.** Pixel values lie in $[0, 1]$, and coordinates are evaluated in $[0, 1000]^2$. The ℓ_∞ budget is measured relative to the clean screenshot for full-image attacks and to the initial natural image for `patch`.

Setting	`redirect`	`disrupt`	`patch`
Region	Full screenshot	Full screenshot	Full-width banner
Optimizer	I-FGSM	I-FGSM	MI-FGSM; momentum 1.0
Step size	$1/255$	$1/255$	Cosine decay: $0.1 \to 0.01$
Iterations	100	100	2,500
Optimization data	Per screenshot	Per screenshot	400 inputs; 4 per step; random positions
Target	Fixed attacker-specified coordinate C_R^*	None	Banner center
Success	Within 9 units of C_R^*	Outside the ground-truth target box	Inside the banner

giving the baseline attack-specific supervision. The threshold is calibrated at the 97th percentile of scores on 150 additional clean inputs. Direction fitting, calibration, and evaluation use disjoint inputs.

JailGuard (Zhang et al., 2025a) uses the maximum pairwise response divergence across six mutated screenshots. We retain its divergence-based scoring procedure and adapt response similarity to predicted coordinates using $s_{ij} = \exp(-\|c_i - c_j\|/\sigma)$ in the normalized coordinate space. The mutations are Gaussian blur, grayscale, brightness jitter, solarization, posterization, and random black occlusion. Geometric transformations are excluded to preserve the target coordinates. Using 250 clean calibration inputs, we set σ to the 95th percentile of pairwise coordinate distances and the detection threshold τ to the 97th percentile of divergence scores. JailGuard therefore requires no adversarial fitting examples.

B OBSERVATION DETAILS

Head-specific entropy shifts. Table 10 provides additional statistics for Figure 2(b), using UI-Venus-1.5-2B on the same 130 AMEX screenshots under each condition. For each head j, we compute the median standardized entropy deviation $d_j = \mathrm{median}_x[(H_j(x) - c_j)/\sigma_j]$, where c_j and σ_j are the entropy median and standard deviation estimated from 1,000 separate clean fitting inputs. Attacks produce deviations with $|d_j| \geq 1$ in 143–217 of the 448 heads, whereas no head meets this criterion under the clean condition. The `patch` and `redirect` attacks predominantly decrease entropy, while `disrupt` produces substantial shifts in both directions. These statistics summarize median shifts across screenshots; individual clean inputs can still exhibit head-wise entropy deviations. The differing responses across heads support retaining each head's entropy as a separate detection feature.

Table 10: **Head-wise entropy shifts.** UI-Venus-1.5-2B on AMEX, with 448 heads and the same 130 screenshots under each condition. Thresholds apply to the median standardized entropy deviation d_j. Lower and Higher count negative and positive deviations, respectively.

	$\lvert d_j\rvert \geq 1$			$\lvert d_j\rvert \geq 2$
Condition	Lower	Higher	Total	Total
Clean	0	0	0	0
`patch`	143	9	152	81
`redirect`	206	11	217	56
`disrupt`	44	99	143	34

Sensitivity to head selection. Table 11 summarizes the random-subset experiment in Figure 3(b). For each evaluated subset size $K < 448$, we sample 1,000 head subsets and fit each detector on the same 1,000 clean inputs. Evaluation uses 300 held-out clean inputs and 300 attacked inputs per attack, with AUROC averaged over the three attacks. Larger subsets yield more consistent detection: at $K = 10$, the minimum and median AUROC are 0.666 and 0.924, respectively, whereas every sampled subset at the evaluated sizes $K \geq 100$ achieves AUROC above 0.99. At $K = 300$, the minimum AUROC is 0.999, compared with 1.000 for the full-head reference. This reduced sensitivity supports using a fixed random subset of 300 heads in this setting.

Table 11: **Detection across random head subsets.** UI-Venus-1.5-2B on AMEX. Statistics summarize attack-averaged AUROC over 1,000 subsets per $K < 448$; Minimum is the lowest observed value. The final column reports the percentage of subsets below 0.99. The last row reports the full-head reference ($K = 448$).

K	Minimum	5th percentile	Median	Below 0.99 (%)
5	0.571	0.688	0.834	100.0
10	0.666	0.807	0.924	99.4
20	0.835	0.930	0.977	86.5
40	0.934	0.983	0.995	20.1
100	0.993	0.998	0.999	0.0
200	0.998	0.999	1.000	0.0
300	0.999	1.000	1.000	0.0
448	1.000	–	–	–

C ADDITIONAL DETECTION RESULTS

Table 12: **Detection at $\epsilon = 8/255$.** Entries are AUROC / TPR@3%FPR, using clean negatives that include natural grounding failures and 300 successful attacks per configuration.

Dataset	Model	`redirect`	`disrupt`	`patch`
AMEX	UI-Venus-1.5-2B	1.000 / 1.000	0.999 / 1.000	1.000 / 1.000
	UGround-V1-2B	0.999 / 1.000	1.000 / 1.000	0.992 / 0.997
	Holo2-4B	1.000 / 1.000	0.992 / 0.963	1.000 / 1.000
	KV-Ground-4B	0.995 / 1.000	0.999 / 1.000	1.000 / 1.000
	Mean	0.999 / 1.000	0.997 / 0.991	0.998 / 0.999
Mind2Web	UI-Venus-1.5-2B	1.000 / 1.000	0.998 / 0.990	0.998 / 0.993
	UGround-V1-2B	1.000 / 1.000	1.000 / 1.000	0.999 / 1.000
	Holo2-4B	1.000 / 1.000	0.990 / 0.953	0.995 / 1.000
	KV-Ground-4B	1.000 / 1.000	0.992 / 0.960	0.996 / 0.997
	Mean	1.000 / 1.000	0.995 / 0.976	0.997 / 0.998

Table 13: **False positives by clean grounding correctness.** FPR at the calibrated threshold on 300 correctly and 300 incorrectly grounded clean inputs per model–dataset pair, averaged over models. †NEARSIDE additionally averages its three attack-specific detectors.

	AMEX		Mind2Web	
Method	Correct	Incorrect	Correct	Incorrect
NEARSIDE†	0.030	0.042	0.031	0.062
JailGuard	0.025	0.101	0.028	0.132
AHEAD	0.032	0.048	0.017	0.044

Table 14: **Benign modifications of clean inputs.** Accuracy and FPR are averaged over four models; thresholds are unchanged. Brightness and noise conditions use a separate set of 100 initially correctly grounded clean inputs per model–dataset pair. Banner conditions use the clean counterparts of 100 attacked inputs at matched positions and sizes. Clean reports the unmodified reference.

	AMEX		Mind2Web	
Condition	Accuracy	FPR	Accuracy	FPR
Clean reference	1.000	0.032	1.000	0.017
Brightness $\times 1.1$	0.975	0.033	0.962	0.028
Uniform noise $\pm 16/255$	0.970	0.035	0.968	0.035
Natural-image banner	0.960	0.030	0.980	0.015
Gray banner	0.978	0.033	0.975	0.007

Table 15: **Matched banner controls.** Benign columns report FPR; the optimized-patch column reports detection rate. Each comparison uses the same 100 inputs.

		Benign FPR		Detection rate
Dataset	Model	Natural image	Gray	Optimized `patch`
AMEX	UI-Venus	0.040	0.040	1.000
	UGround	0.040	0.040	0.970
	Holo2	0.020	0.020	1.000
	KV-Ground	0.020	0.030	1.000
Mind2Web	UI-Venus	0.010	0.010	1.000
	UGround	0.030	0.010	0.610
	Holo2	0.010	0.010	1.000
	KV-Ground	0.010	0.000	0.830

Detection at a reduced perturbation budget. Table 12 evaluates AHEAD at $\epsilon = 8/255$ using the same clean splits and fitted detectors as Table 1. Full-image attacks use 300 iterations with a step size of $0.25/255$; patch optimization halves both the perturbation budget and the step-size schedule of the standard setting. Across the 24 configurations, AHEAD achieves a mean AUROC of 0.998 and a mean TPR@3%FPR of 0.994, with minimum values of 0.990 and 0.953, respectively. At the unchanged calibrated thresholds, TPR@τ is 0.998 on AMEX and 0.962 on Mind2Web. FPR remains 0.037 and 0.017, respectively, because the clean evaluation inputs and thresholds are unchanged. These results extend the detection findings to successful attacks at the smaller budget; the two budgets use different optimization schedules.

Variation across random head subsets. We repeat the main evaluation with $K = 300$ using ten head-sampling seeds (0–9). The mean TPR@3%FPR over the 24 configurations ranges from 0.977 to 0.994. The main result uses seed 0, with a mean AUROC of 0.997 and a mean TPR@3%FPR of 0.990. This range indicates limited variation in aggregate detection performance across the evaluated head subsets.

False positives on natural grounding failures. Table 13 applies each detector's calibrated threshold to separate sets of 300 correctly and 300 incorrectly grounded clean inputs per model–dataset pair. AHEAD's FPR increases by 1.6 percentage points on AMEX and 2.7 on Mind2Web for incorrectly grounded inputs, compared with increases of 7.6 and 10.4 points for JailGuard. Replacing mixed clean negatives with correctly grounded inputs changes AHEAD's mean TPR@3%FPR by less than 0.005, whereas NEARSIDE and JailGuard improve by 0.031 and 0.065, respectively. AHEAD therefore maintains similar detection performance under both clean-negative evaluation protocols.

Benign modifications and matched banner controls. We evaluate benign image modifications without refitting AHEAD or recalibrating its threshold (Table 14). Mean FPR is at most 0.035 across the tested modifications. Table 15 further compares optimized patches with natural-image and gray banners on the same 100 inputs, using matched positions and sizes within each model–dataset pair. Benign banners produce FPRs of at most 4% in every configuration, whereas optimized patches are detected on 61%–100% of inputs. Thus, the elevated detection rates for optimized patches are not reproduced by these matched benign overlays.

D LATENCY MEASUREMENT DETAILS

Measurement protocol. We measure end-to-end latency from a decoded screenshot and instruction to the parsed coordinate and, when applicable, the detection decision on the CPU. Measurements include preprocessing, coordinate generation, and all detector-specific operations, but exclude model loading and offline detector fitting. All methods run on a single RTX 3090 with batch size 1 and bfloat16 precision. For each model, we time 100 clean evaluation inputs on AMEX and 50 on Mind2Web, following 10 and 5 warm-up inputs, respectively.

Latency overhead. Table 16 reports absolute latency. AHEAD adds approximately 7% on AMEX and 9% on Mind2Web, averaged across models, consistent with Table 1. It extracts attention during coordinate generation without an additional forward pass. JailGuard requires six additional generations and increases latency by approximately a factor of seven.

Table 16: **Per-input end-to-end latency (seconds).** Means over clean inputs; all methods within each model–dataset configuration are timed in the same run. †NEARSIDE uses attack-specific directions.

Dataset	Model	Grounding only	+AHEAD	+NEARSIDE†	+JailGuard
AMEX	UI-Venus	0.489	0.542	0.680	3.447
	UGround	3.155	3.167	3.480	22.079
	Holo2	3.480	3.559	3.751	23.609
	KV-Ground	1.432	1.635	1.766	10.058
Mind2Web	UI-Venus	0.438	0.483	0.642	3.092
	UGround	3.247	3.256	3.595	22.760
	Holo2	0.886	0.991	1.183	6.252
	KV-Ground	1.418	1.610	1.763	9.971

E CROSS-DOMAIN TRANSFER DETAILS

Transfer protocol. Table 17 details the 50-input adaptation protocol used in Table 3. This budget covers both parameter estimation and threshold calibration. Adaptation uses no target attacks or grounding-correctness labels, and all methods share the same target evaluation inputs. We repeat sampling 100 times from each method's target pool, which is disjoint from the target evaluation set. AHEAD results additionally average over ten source head subsets.

Table 17: **Transfer protocol by method.** Each method adapts using 50 clean inputs from the indicated target pool. †NEARSIDE uses a separate direction for each attack.

Method	Kept from source	Target updates	Target pool
AHEAD	Head subset; covariance Σ_s for whitening	Mean; covariance correction; threshold	Fitting (1,000)
NEARSIDE†	Attack direction	Threshold	Clean (300)
JailGuard	Mutations; divergence rule	Bandwidth; threshold	Calibration (250)

Clean-only reference adaptation. AHEAD retains its fixed subset of $K = 300$ source heads. Let $\phi(x) \in \mathbb{R}^K$ denote their entropy features (Equation 5), and let Σ_s denote the regularized covariance fitted on clean source inputs (Section 5.1). We write L_s for the lower-triangular Cholesky factor of Σ_s, satisfying $\Sigma_s = L_s L_s^\top$, and transform the features into source-whitened coordinates: $z(x) = L_s^{-1}\phi(x)$. Given n clean target inputs $\{x_i\}_{i=1}^n$, we compute their mean $\hat{\mu}_z = n^{-1}\sum_i z(x_i)$ and sample covariance $\hat{S} = (n-1)^{-1}\sum_i v_i v_i^\top$, where $v_i = z(x_i) - \hat{\mu}_z$. To regularize estimation from this small sample, we use a Ledoit–Wolf-inspired covariance weight a (Ledoit & Wolf, 2004).[1] Let (λ_j, u_j) be the retained eigenpairs of $\hat{S}$, where $\lambda_j > 10^{-10}$ and the eigenvectors u_j are orthonormal. The adapted score is

$$S(x)^2 = \max\left\{0,\ \|v(x)\|_2^2 - \sum_j w_j\left(u_j^\top v(x)\right)^2\right\}, \qquad w_j = \frac{a\lambda_j}{1-a+a\lambda_j}, \qquad v(x) = z(x) - \hat{\mu}_z. \tag{9}$$

The correction reduces the contribution of directions in which clean target inputs vary, while preserving the source metric outside the retained target subspace. Mean-only adaptation uses $a = 0$. Both variants use only clean target inputs.

Threshold calibration. We use five-fold cross-fitting to estimate the threshold within the same target budget. For each fold, the mean, covariance, and weight a are re-estimated from the other four folds, and the held-out inputs are scored using that reference. Pooling these out-of-fold scores into S_{oof}, we set

$$\tau = \sqrt{\frac{1+1/n}{1+1/n'}}\, Q_{0.97}(S_{\text{oof}}), \qquad n' = n - \lceil n/5 \rceil, \tag{10}$$

where $Q_{0.97}$ is the empirical 97th percentile and n' is the smallest training-fold size under a nearly equal five-fold split. The prefactor approximately adjusts for the difference in mean-estimation sample size; it is 0.998 at $n = 50$. The deployed detector is then refitted on all n target inputs. Mean-only adaptation follows the same calibration procedure with $a = 0$.

[1] The implemented weight is $a = 1 - \min(\bar{\beta}^2, \delta^2)/\max(\delta^2, 10^{-12})$, where $\delta^2 = \|\hat{S} - \bar{\lambda}I\|_F^2/K$, $\bar{\lambda} = \mathrm{tr}(\hat{S})/K$, and $\bar{\beta}^2 = (n^2K)^{-1}\sum_i \|v_i v_i^\top - \hat{S}\|_F^2$. Here, I is the $K \times K$ identity matrix and $\|\cdot\|_F$ denotes the Frobenius norm. For fewer than three fitting inputs, we set $a = 0$. The coefficient is computed relative to $\bar{\lambda}I$, while the scoring rule uses the fixed identity reference inherited from source whitening.

Updating the clean reference improves discrimination. Table 18 isolates the contribution of the covariance correction to the recovery reported in Table 3. At $n = 50$, mean-only adaptation raises TPR@3%FPR from the source-only values of 0.652 and 0.432 to 0.859 and 0.747 for AMEX $\rightarrow$ Mind2Web and the reverse direction, respectively. Adding the covariance correction further raises both values to 0.907. At the calibrated thresholds, the corresponding achieved FPRs are 0.020 and 0.032.

The gains at a matched FPR show that updating the clean reference improves the separation of clean and attacked inputs. Larger target samples provide further discrimination gains with the covariance correction, while mean-only adaptation largely plateaus beyond $n = 50$. The achieved FPRs also become more conservative at larger budgets, particularly for AMEX $\rightarrow$ Mind2Web, consistent with the approximate nature of the calibration procedure.

Table 18: **Detection across target sample budgets.** TPR is measured at a matched 3% FPR; FPR is measured at the calibrated threshold τ. Results average over four GUI agents and ten head subsets; AUROC and TPR additionally average over three attacks. Adaptation results further average over 100 target-sample draws. Each n is the total target budget.

	AMEX → Mind2Web			Mind2Web → AMEX		
n	AUROC	TPR	FPR	AUROC	TPR	FPR
Mean-only adaptation						
20	0.981	0.855	0.063	0.969	0.748	0.067
50	0.982	0.859	0.040	0.969	0.747	0.047
100	0.982	0.863	0.033	0.969	0.746	0.042
300	0.982	0.863	0.027	0.971	0.750	0.034
Mean and covariance adaptation						
20	0.987	0.883	0.046	0.981	0.831	0.052
50	**0.990**	**0.907**	**0.020**	**0.988**	**0.907**	**0.032**
100	0.992	0.921	0.009	0.992	0.949	0.024
300	0.994	0.948	0.002	0.996	0.993	0.010
Source-only reference ($n = 0$)						
0	0.892	0.652	1.000	0.689	0.432	1.000
Target-fitted reference (1,000 fitting + 250 calibration)						
1,250	0.997	0.972	0.030	0.998	0.999	0.032

Consistent gains across GUI agents. Table 19 compares the two adaptation variants at $n = 50$. The covariance correction increases TPR@3%FPR and lowers FPR@τ for every evaluated agent–direction pair. The largest TPR gains occur when transferring UGround and Holo2 from Mind2Web to AMEX, from 0.668 to 0.900 and from 0.606 to 0.839, respectively.

Table 19: **Adaptation with 50 clean target inputs by GUI agent.** TPR is measured at a matched 3% FPR; FPR is measured at τ. Results average over ten head subsets and 100 adaptation draws; TPR additionally averages over three attacks.

	Mean only		Mean and covariance	
GUI agent	TPR	FPR	TPR	FPR
AMEX → Mind2Web				
UI-Venus	0.992	0.046	0.994	0.026
UGround	0.814	0.027	0.899	0.015
Holo2	0.891	0.035	0.932	0.015
KV-Ground	0.740	0.050	0.803	0.026
Mind2Web → AMEX				
UI-Venus	0.975	0.051	0.986	0.031
UGround	0.668	0.036	0.900	0.028
Holo2	0.606	0.056	0.839	0.037
KV-Ground	0.740	0.046	0.904	0.030

Why clean-only adaptation differs across methods. For NEARSIDE, the cosine similarity between source and target attack directions averages 0.50 (standard deviation 0.38) across the 24 agent–direction–attack configurations. Target attack examples are used for this diagnostic, not for adaptation. With the source direction fixed, threshold recalibration leaves score rankings unchanged. JailGuard's bandwidth update changes mean TPR@3%FPR from 0.201 to 0.184 and from 0.179 to 0.190 in the two directions (Table 3), providing little improvement at low FPR. AHEAD uses clean target inputs to update the reference mean and covariance correction, changing the anomaly score itself and producing the discrimination gains quantified above.

F ADAPTIVE ATTACK DETAILS

Evaluation protocol. We evaluate the attacks of Section 6.4 on the same 100 AMEX screenshots using KV-Ground-4B. Inputs have clean predictions and ground-truth boxes and are selected before attack generation, disjoint from detector fitting, calibration, and evaluation sets and from the 400 patch-training inputs. All 100 attempts enter the reported rates, without conditioning on attack success. The attacker has white-box access to the GUI agent and four fixed surrogate AHEAD detectors, each with an independently sampled subset of $K = 300$ heads. At every optimization step, the evasion penalty averages over all four surrogates. Evaluation uses 20 independently fitted detectors with fresh head subsets that are unavailable to the attacker during optimization.

For each evaluation detector, evasion is the fraction of inputs that pass its threshold, and end-to-end ASR is the fraction that both achieve the attack objective and pass the detector. We report the median of each rate separately across the 20 detectors. End-to-end ASR is computed from the joint outcome on each input, rather than the product of the marginal rates.

Attack configuration. Table 20 lists the objectives and success criteria. Other attack settings follow Appendix A. The control $\lambda = 0$ uses the same settings without an evasion penalty; the evaluated weights are listed in Table 21. For `redirect`, optimization retains the surrogate-accepted iterate with the lowest attack loss.

Table 20: **Adaptive attack settings.** All attacks use $\epsilon = 16/255$. Coordinate distances are measured in the normalized $[0, 1000]^2$ space.

Setting	`redirect`	`disrupt`	`patch`
Objective	Cross-entropy to C_R^*	Negative visual-feature distance	Cross-entropy to banner center
Optimizer	I-FGSM	I-FGSM	MI-FGSM
Iterations	100	100	2,500
Success	Within 9 units of C_R^*	Outside the ground-truth box	Inside the banner

Table 21: **Full penalty-weight sweep.** All rates are percentages over 100 inputs. Evasion and end-to-end ASR are separate medians across 20 unseen detectors; the range gives minimum and maximum evasion. Bold weights and ASRs identify the adaptive settings reported in Table 4.

			Evasion		
Attack	λ	Attack success	Median	Range	End-to-end ASR
`redirect`	0	96.0	1.0	[0.0, 4.0]	0.0
	1	58.0	33.0	[21.0, 53.0]	4.0
	3	77.0	21.5	[7.0, 42.0]	8.0
	10	91.0	13.5	[2.0, 33.0]	**11.0**
	30	93.0	13.5	[1.0, 30.0]	9.0
	50	93.0	10.0	[0.0, 35.0]	9.0
`disrupt`	0	22.0	0.0	[0.0, 0.0]	0.0
	300	5.0	32.5	[20.0, 63.0]	**1.5**
	3,000	11.0	3.0	[0.0, 40.0]	0.0
	10,000	8.0	6.5	[0.0, 45.0]	0.0
`patch`	0	82.0	1.0	[0.0, 8.0]	0.0
	1	92.0	9.5	[2.0, 50.0]	6.5
	5	63.0	60.5	[39.0, 81.0]	27.0
	10	56.0	67.0	[46.0, 81.0]	**27.5**

G FEATURE AND SCORING ABLATION DETAILS

Matched comparisons. Tables 5 and 6 use identical fitting, calibration, and evaluation inputs. Feature comparisons use 300 dimensions and full Mahalanobis scoring with ridge $10^{-6}I$. Entropy and attention mass share head subsets across ten paired draws (seeds 0–9). Hidden states are averaged over the answer span, using the last layer or concatenated layers, before selecting 300 random coordinates. Scoring comparisons hold the entropy head subsets fixed. Each threshold is the 97th percentile of the variant's clean calibration scores.

Hidden-state alternatives. Random coordinate selection outperforms PCA for both hidden-state representations (Table 22). With all available features and Ledoit–Wolf covariance estimation, entropy achieves TPR@3%FPR of 0.994, compared with 0.954 for all-layer hidden states. Its advantage therefore also holds without dimensionality reduction.

Table 22: **Alternative hidden-state representations.** Mean TPR@3%FPR across 24 configurations. Random-coordinate results show mean ± SD over ten draws. PCA uses standardized features and is fitted only on clean fitting inputs. The final row uses standardized full features and changes both dimensionality and covariance estimator.

Representation	Last layer	All layers
Random 300 coordinates	0.702 ± 0.008	0.896 ± 0.009
PCA-300	0.592	0.765
Full dimension, Ledoit–Wolf	0.731	0.954

Feature performance by dataset and attack. Table 23 shows that entropy's advantage over attention mass is concentrated in `patch`, particularly on Mind2Web (0.953 versus 0.752). Both features perform strongly on `redirect` and `disrupt`. Entropy achieves higher mean TPR@3%FPR across the 24 configurations in all ten paired draws. These results support summarizing how attention is distributed within the image, rather than only its total mass.

Contribution of the scoring rule. The mean-entropy baseline uses absolute standardized deviation from the clean mean. Retaining individual head entropies, accounting for their variances, and then modeling their correlations successively improve TPR@3%FPR in every dataset–attack condition (Table 23). Each step improves mean performance in all ten paired draws, supporting full-covariance scoring of head-wise entropy deviations.

Table 23: **Ablations by dataset and attack.** TPR@3%FPR averaged over GUI agents; entropy and mass results additionally average ten paired head draws. Feature rows use full Mahalanobis scoring; scoring rows use entropy features. Red., Dis., and Pat. denote `redirect`, `disrupt`, and `patch`.

	AMEX			Mind2Web		
Variant	Red.	Dis.	Pat.	Red.	Dis.	Pat.
Feature comparison						
Attention entropy	1.000	0.998	0.997	1.000	0.984	0.953
Attention mass	1.000	0.999	0.950	1.000	0.988	0.752
Scoring comparison						
Mean entropy	0.376	0.301	0.300	0.331	0.408	0.381
Euclidean	0.704	0.671	0.582	0.894	0.688	0.499
Diagonal Mahalanobis	0.779	0.824	0.690	0.919	0.728	0.623
Full Mahalanobis	1.000	0.998	0.997	1.000	0.984	0.953