GEM: Gaussian Embedding Modeling for Out-of-Distribution Detection in GUI Agents

Zheng Wu, Pengzhou Cheng, Zongru Wu, Lingzhong Dong, Zhuosheng Zhang*

School of Computer Science, Shanghai Jiao Tong University
{wzh815918208, zhangzs}@sjtu.edu.cn

Abstract

Graphical user interface (GUI) agents have recently emerged as an intriguing paradigm for human-computer interaction, capable of automatically executing user instructions to operate intelligent terminal devices. However, when encountering out-of-distribution (OOD) instructions that violate environmental constraints or exceed the current capabilities of agents, GUI agents may suffer task breakdowns or even pose security threats. Therefore, effective OOD detection for GUI agents is essential. Traditional OOD detection methods perform suboptimally in this domain due to the complex embedding space and evolving GUI environments. In this work, we observe that the in-distribution input semantic space of GUI agents exhibits a clustering pattern with respect to the distance from the centroid. Based on the finding, we propose GEM, a novel method based on fitting a Gaussian mixture model over input embedding distances extracted from the GUI agent that reflect its capability boundary. Evaluated on 8 datasets spanning smartphones, computers, and web browsers, our method achieves an average accuracy improvement of 23.70% over the best-performing baseline while only increasing training time by 4.9% and testing time by 6.5%. We also experimentally demonstrate that GEM can improve the step-wise success rate by 9.40% by requesting assistance from the cloud model when encountering OOD samples. Analysis verifies the generalization ability of our method through experiments on nine different backbones.

Code —
https://github.com/Wuzheng02/GEM-OODforGUIagents

Introduction

Recently, graphical user interface (GUI) agents (Zhang et al. 2024a; Tang et al. 2025b) have emerged as an intriguing paradigm to human-computer interaction, capable of autonomously executing user instructions and performing human-like control on intelligent terminal devices such as smartphones, computers, and web browsers. The common approach to building GUI agents involves post-training multimodal large language models (MLLMs) using high-quality trajectory data to enhance key task capabilities such as perception (Li et al. 2023; Wang et al. 2023b), reasoning (Yao et al. 2023; Wei et al. 2022), and reflection (Liu et al. 2025b; Hu et al. 2025). Despite notable advances in instruction following, GUI agents remain vulnerable to out-of-distribution (OOD) risks—executing instructions that violate environmental constraints (e.g., non-existent functions or applications) or exceed the agent's current capabilities. These failures can lead to task breakdowns or even pose security threats. In real-world applications, OOD risks for GUI agents come with two primary forms: (i) **Internalization-OOD**, where the agent operates in domain-specific environments (e.g., a particular type of smartphones or vehicle cabins) but incorrectly assumes the presence of unsupported capabilities; (ii) **Extrapolation-OOD**, where the agent encounters instructions tied to dynamic or evolving environments (e.g., new third-party applications).

As illustrated in Figure 1, following OOD instructions and executing an incorrect action path (e.g., unintentionally resetting a smartphone) can result in critical failures. Therefore, it is essential for GUI agents to incorporate OOD detection mechanisms that identify tasks beyond their supported scope. This not only mitigates potential operational risks but also enables targeted enhancements to agent capabilities through continued research and development.

Popular OOD detection methods can be broadly categorized (Malinin and Gales 2018) into two types: embedding-based approaches (Lee et al. 2018) and model uncertainty-based approaches (Hendrycks and Gimpel 2016). These methods have demonstrated effectiveness in traditional (M)LLM tasks such as summarization (Ren et al. 2022), visual question answering (Kervadec et al. 2021), mathematical reasoning (Wang et al. 2024), and text generation (Wu et al. 2022). Directly applying these existing OOD detection techniques to the GUI agent domain results in suboptimal performance. Notably, even with the classification threshold calibrated to correctly identify 95% of in-distribution (ID) tasks, the best-performing method still misclassifies between 31.75% and 48.92% of OOD tasks as ID.

Compared with traditional (M)LLM-based OOD detection tasks, OOD detection for GUI agents presents two critical challenges:

(i) Complex Embedding Space: The inputs to GUI agents are inherently complex (Ma, Zhang, and Zhao 2024). Concretely, GUI screens typically contain densely populated UI components (Zhang et al. 2024a), leading to higher infor-

*Corresponding author.

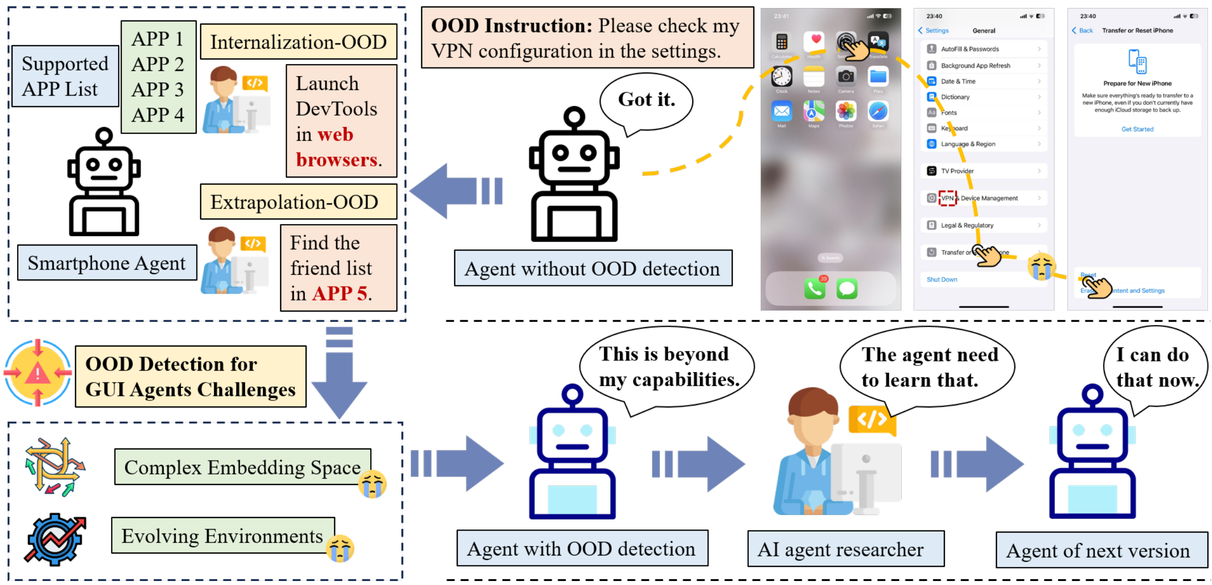

Figure 1: Comparison between an agent w/ and w/o OOD detection when facing an OOD instruction. Also illustrated are the OOD scenarios and the challenges of OOD detection for GUI agents.

mation density than traditional MLLM tasks. Besides, the diversity of user instructions further complicates the inference of user intent. This complexity substantially increases the difficulty of effective OOD detection for GUI agents.

(ii) Evolving Environments: GUI environments frequently change due to system upgrades or the installation of new third-party applications. As a result, GUI agents must continually adapt their capabilities (Wang et al. 2025; Liu et al. 2025a). This evolving nature complicates OOD detection by demanding both accurate capability assessment and temporal adaptability. Consequently, reliance on static external classifiers becomes increasingly impractical.

To address the challenges above, we propose **GEM**, a novel method based on fitting a Gaussian mixture model (GMM) (Reynolds et al. 2009) over input embedding distances extracted from the GUI Agent that reflect its capability boundary. Specifically, we first extract the input embeddings from the encoder layer of the GUI agent on its training data, and fit them into a high-dimensional hypersphere. We then compute the L2-norm distances of all embeddings relative to the centroid of the hypersphere. Under bayesian information criterion (BIC) (Neath and Cavanaugh 2012) supervision, we fit a GMM to these distances and define the OOD detection boundary as a configurable number of standard deviations away from each GMM cluster center. Experiments across eight GUI agent datasets spanning smartphones, computers, and web browsers platforms show that our method consistently outperforms all traditional OOD detection methods. And we provide a possible method to use GEM to improve the end-to-end results for GUI agents.

In summary, we make three key contributions:

(i) We present the first systematic analysis of OOD detection for GUI agents and compare various widely adopted OOD detection methods in this domain.

(ii) We propose a novel GMM-based approach that leverages input embedding distances from MLLMs for OOD detection. Our method achieves an average accuracy improvement of 23.70% over the best performing baseline across eight datasets spanning three platform types while only increasing training time by 4.9% and testing time by 6.5%.

(iii) We offer some new insights into this emerging research area and provide a possible method to use GEM to improve the end-to-end results for GUI agents.

Related Work

MLLM-based GUI Agents

With the advancement of MLLMs (Liu et al. 2023; Achiam et al. 2023), numerous GUI agent foundation models (Hong et al. 2024; Wu et al. 2024; Qin et al. 2025) have emerged, capable of executing user instructions across smartphones, computers, and web browsers. For the technical framework, prompt-based approaches (Wang et al. 2025; Jiang et al. 2025; Wu et al. 2025c) have been developed, leveraging closed-source models to construct agent systems that fulfill user instructions. Additionally, researchers have also explored pre-training (Ma, Zhang, and Zhao 2024; Wu et al. 2025a), supervised fine-tuning (SFT) (Zhang and Zhang 2024; Wu et al. 2025b), and reinforcement learning (RL) (Zhou et al. 2024; Xia and Luo 2025; Tang et al. 2025a) techniques to further enhance the ability of GUI agents to complete user instructions. Regardless of their advancements, GUI agents inevitably face capability limitations. In complex and dynamic real-world environments (Cheng et al. 2025a; Guo et al. 2025), GUI agents are prone to encountering OOD situations.

Method	Smartphone		Computer		Web browser	
	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓
Embedding-based methods						
TV score	54.26	98.02	60.13	85.37	60.27	89.48
Last layer embedding	50.94	76.48	64.31	68.73	67.51	74.58
Best layer embedding	**76.14**	**48.92**	**89.72**	**45.60**	**89.77**	**31.75**
Uncertainty-based methods						
Top-k confidence	67.07	85.10	57.51	93.59	57.40	94.41
Output entropy	67.07	92.69	57.51	86.22	57.40	86.73

Table 1: Results of the pilot study. Existing popular OOD detection methods perform poorly in the GUI agent domain.

OOD Detection in MLLMs

For OOD detection in MLLMs, it is necessary to jointly consider both visual and textual modalities, in contrast to traditional computer vision OOD detection, which only requires modeling the visual modality (Liu et al. 2020; Ren et al. 2019), or LLMs OOD detection, which focuses on textual modality (Wang et al. 2024; Ren et al. 2022). In the context of MLLMs, some researchers have proposed using maximum concept matching (Ming et al. 2022) to characterize OOD uncertainty, while others have approached the problem by training models on ID datasets and classifying whether an input image belongs to an unknown category (Wang et al. 2023a). However, OOD detection for MLLMs remains a challenging research area (Dong et al. 2024; Lu et al. 2024). Furthermore, for MLLM-based GUI agents, task scenarios are significantly more complex and dynamic (Shi et al. 2025) compared to traditional MLLM tasks, making OOD detection even more difficult.

Investigating the Challenge of OOD Detection for GUI Agents

In this section, we conduct pilot experiments to evaluate the effectiveness of popular OOD detection methods for GUI agents and understand the challenges. We will provide the problem formulation and present the experimental settings followed by the key results and analysis of the pilot study.

Problem Formulation

A GUI agent $\mathcal{F}$ is initially trained on an ID dataset $\mathcal{D}_{\text{ID}} = \{(s_i, x_i)\}_{i=1}^{k}$, consisting of k pairs of screenshots s_i and user instructions x_i. $\mathcal{F}$ learns knowledge of operating systems through methods such as SFT and RL on $\mathcal{D}_{\text{ID}}$.

After deployment on the device, $\mathcal{F}$ receives an instruction x and captures the current device screenshot s_t. $\mathcal{F}$ then constructs a prompt that combines s_t and x, which is subsequently used to predict an action a_t. Formally, at each time step t ($0 \leq t \leq T$), the process can be expressed as: $a_t = \mathcal{F}(s_t, x)$ where s_t is the screenshot at time step t, x is the instruction, and a_t is the action predicted by the agent.

Then a_t is executed, leading to a new screenshot s_{t+1}. $\mathcal{F}$ evaluates whether the instruction x has been completed. If x is not completed, the agent repeats this process until either x is completed or the maximum step limit T is reached. Throughout the execution process, it is possible that some pairs of screenshots and instructions (s_t, x) may deviate significantly from the distribution of $\mathcal{D}_{\text{ID}}$. Such pairs are classified as OOD samples. When these OOD samples are encountered, the agent's action predictions are prone to errors, which can lead to undesirable execution results.

The objective of OOD detection for GUI agents, therefore, is to identify whether the pair (s_t, x) at each time step t deviates from the distribution of $\mathcal{D}_{\text{ID}}$. If OOD is detected, the agent should immediately terminate the execution of the instruction and alert the user; otherwise, it continues with the execution. To formally define the OOD detection mechanism, we introduce the following OOD detection function f_{OOD}, which operates on each pair (s_t, x). If (s_t, x) is OOD, $f_{\text{OOD}}(s_t, x)$ returns a value of 1. Otherwise, the agent will predict the appropriate action a_t and proceed to execute it.

Pilot Study with Popular OOD Detection Methods

Existing popular OOD detection methods (Malinin and Gales 2018; Yang et al. 2024) can be broadly categorized into two main types: embedding-based methods (Lee et al. 2018) and uncertainty-based methods (Gal and Ghahramani 2016). Based on existing research (Dong et al. 2024; Lu et al. 2024) in OOD detection for (M)LLMs, we used three embedding-based methods and two uncertainty-based methods for experimentation.

We evaluated the methods used as baselines across eight datasets spanning three platforms. Following standard OOD detection method evaluation criteria (Luan et al. 2021; Cui and Wang 2022), we plotted the ROC curve and reported the area under the receiver operating characteristic curve (AUROC) (Metz 1978) and false positive rate at 95% true positive rate (FPR95).

The pilot experiment results are shown in Table 1. The AUROC metric reflects the separability between OOD and ID datasets achieved by different methods. Even the best baseline performances only range from 76.14% to 89.77%, barely reaching the acceptable level expected for OOD detection. The FPR95 metric measures the false positive rate when the true positive rate reaches 95%. Pilot experiments show that even the best baselines result in an FPR95 of 31.75% to 48.92%, indicating that while maintaining 95% task success for the GUI agent, 31.75% to 48.92% of OOD

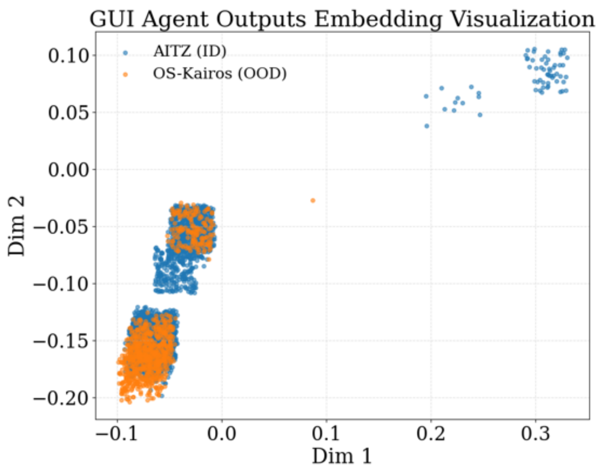

Figure 2: The output embeddings of the OS-Kairos and AITZ are visualized. Most of the samples are confused.

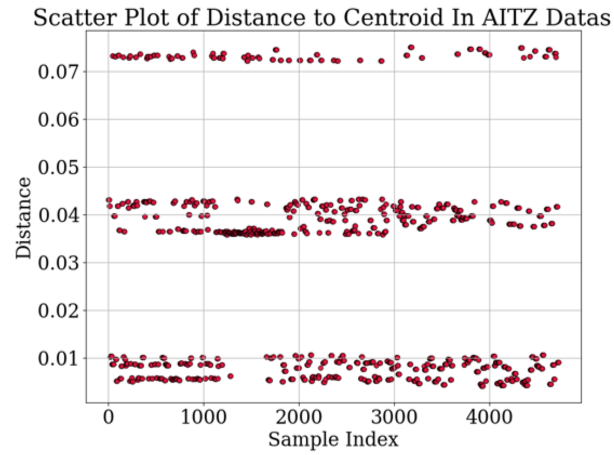

Figure 3: The multimodal embeddings of the AITZ dataset show a clustered distribution pattern around the centroid.

tasks would be mistakenly executed, posing significant risks.

Why do these OOD detection methods perform suboptimally?

For embedding-based methods, due to the relatively uniform reasoning patterns of GUI agents—TV score, which rely on differences in layer embeddings to infer reasoning path divergence, perform suboptimally. Other methods based on single-layer embeddings have achieved better performance; however, since GUI agents tend to lose some information related to input understanding during the reasoning process, there is still room to improve.

For uncertainty-based methods, the separability between the output spaces of ID and OOD samples in the GUI agent domain is extremely low (as shown in Figure 2). As a result, GUI agents struggle to estimate the uncertainty of their outputs, making uncertainty-based methods nearly ineffective to distinguish between ID and OOD samples.

These OOD detection methods rely on finding a decision boundary by identifying the Youden Index (Fluss, Faraggi, and Reiser 2005) after obtaining some scoring metric, that is Youden Index $= \arg\max_t (\text{TPR}(t) - \text{FPR}(t))$, where t is the threshold, and TPR and FPR represent the true positive rate and false positive rate.

However, popular GUI agents (Wu et al. 2024; Qin et al. 2025; Xu et al. 2024) have access to diverse training data. In the embedding space, the ID dataset representations for these GUI agents naturally form clusters according to their different data sources. Moreover, we observe that even for datasets originating from a single data source such as AITZ (Zhang et al. 2024b), as shown in Figure 3, there can still be noticeable clustering phenomena. We also observe that samples farther from the centroid tend to yield higher success rates. For such non-linearly separable data, the Youden Index approach becomes inadequate.

GEM Method

Pilot experiments show that popular OOD detection methods perform suboptimally in the GUI agent domain. However, we observe that the input embeddings generated by GUI agents naturally lend themselves to modeling the ID representation of $\mathcal{D}_{\text{ID}}$.

Given a GUI agent $\mathcal{F}$ and an ID dataset $\mathcal{D}_{\text{ID}} = \{(s_i, x_i)\}_{i=1}^k$, we first obtain an encoder layer l_e from $\mathcal{F}$. The encoder l_e maps each input pair (s_i, x_i) to an embedding vector $e_i \in R^n$. Thus, we construct an ID embedding dataset $\mathcal{D}_{\text{embedding}} = \{e_i\}_{i=1}^k$. Each e_i is a point in the n-dimensional embedding space.

To model this distribution, we first compute the centroid μ of $\mathcal{D}_{\text{embedding}}$: $\mu = \frac{1}{k}\sum_{i=1}^k e_i$. Next, we calculate the Euclidean distance between each embedding e_i and the centroid μ: $d_i = \|e_i - \mu\|_2, \quad i = 1, \ldots, k$, resulting in a distance dataset $\mathcal{D}_{\text{distance}} = \{d_i\}_{i=1}^k$.

The distribution of $\mathcal{D}_{\text{distance}}$ is typically multi-centered and may contain multiple modes. Therefore, instead of fitting a simple Gaussian or applying heuristic thresholds, we model it using a GMM. Specifically, we assume that the distances are generated from a mixture of m univariate Gaussian components. The GMM models the probability density function as: $p(d) = \sum_{j=1}^m \pi_j \mathcal{N}(d \mid \mu_j, \sigma_j^2)$, π_j is the mixing coefficient of the j-th component, satisfying $\sum_{j=1}^m \pi_j = 1$ and $\pi_j \geq 0$, $\mathcal{N}(d \mid \mu_j, \sigma_j^2)$ denotes the density of a univariate Gaussian:

$$\mathcal{N}(d \mid \mu_j, \sigma_j^2) = \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left(-\frac{(d-\mu_j)^2}{2\sigma_j^2}\right). \quad (1)$$

Given the dataset $\mathcal{D}_{\text{distance}}$, the log-likelihood of the data under the GMM is:

$$\log \mathcal{L}_m = \sum_{i=1}^k \log\left(\sum_{j=1}^m \pi_j \mathcal{N}(d_i \mid \mu_j, \sigma_j^2)\right). \quad (2)$$

Method	AndroidControl Acc.(%)↑	Prec.(%)↑	Rec.(%)↑	F1(%)↑	OS-Kairos Acc.(%)↑	Prec.(%)↑	Rec.(%)↑	F1(%)↑
TV score	55.37	42.48	68.95	52.57	77.36	90.20	82.20	86.01
Top-k confidence	68.29	77.30	71.58	74.33	63.71	26.04	74.47	38.59
Output entropy	68.29	55.13	62.43	58.55	63.71	93.05	61.77	74.25
Last layer embed.	70.08	69.02	96.77	80.57	32.66	17.45	91.10	29.29
Best layer embed.	78.20	77.02	94.07	84.69	74.35	33.29	67.33	44.56
GEM (ours)	**99.39**	**98.33**	**100.0**	**99.16**	**100.0**	**100.0**	**100.0**	**100.0**

Method	Meta-GUI Acc.(%)↑	Prec.(%)↑	Rec.(%)↑	F1(%)↑	ScreenSpot-Mobile Acc.(%)↑	Prec.(%)↑	Rec.(%)↑	F1(%)↑
TV score	68.56	71.91	91.51	80.53	65.54	92.01	67.76	78.04
Top-k confidence	54.60	34.54	63.60	44.77	52.93	11.91	60.96	19.92
Output entropy	54.60	77.46	50.93	61.46	52.93	92.62	52.07	66.67
Last layer embed.	59.38	39.47	75.72	51.89	30.71	12.05	98.61	21.47
Best layer embed.	76.59	55.50	96.31	70.42	62.72	19.60	92.83	32.36
GEM (ours)	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**

Method	Omniact-Desktop Acc.(%)↑	Prec.(%)↑	Rec.(%)↑	F1(%)↑	ScreenSpot-Desktop Acc.(%)↑	Prec.(%)↑	Rec.(%)↑	F1(%)↑
TV score	47.50	29.04	82.36	42.94	23.92	7.66	95.21	14.18
Top-k confidence	44.54	84.05	33.36	47.76	31.00	95.43	27.43	42.62
Output entropy	44.54	27.47	79.95	40.88	31.00	7.35	81.44	13.49
Last layer embed.	60.05	35.80	83.84	50.17	46.54	10.05	89.22	18.06
Best layer embed.	**91.15**	83.73	78.34	80.94	43.46	10.18	96.71	18.43
GEM (ours)	89.53	**87.89**	**100.0**	**93.55**	**96.86**	**96.74**	**100.0**	**98.34**

Method	Omniact-Web Acc.(%)↑	Prec.(%)↑	Rec.(%)↑	F1(%)↑	ScreenSpot-Web Acc.(%)↑	Prec.(%)↑	Rec.(%)↑	F1(%)↑
TV score	48.63	13.07	72.45	22.15	45.35	11.27	79.59	19.75
Top-k confidence	61.36	91.17	63.15	74.61	42.97	94.91	39.84	56.12
Output entropy	61.36	12.16	45.47	19.19	42.97	10.54	76.83	18.54
Last layer embed.	61.82	18.52	81.89	30.20	47.77	12.64	87.61	22.09
Best layer embed.	86.03	40.34	80.38	53.72	74.77	24.77	97.48	39.50
GEM (ours)	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**

Table 2: Comparison of OOD Detection Results Across Different Datasets.

The parameters $\{\pi_j, \mu_j, \sigma_j^2\}$ are estimated by maximizing the log-likelihood $\log \mathcal{L}_m$ via the expectation-maximization (EM) algorithm. Each iteration of EM consists of E-step and M-step. E-step estimate the posterior probability that sample d_i belongs to component j:

$$\gamma_{ij} = \frac{\pi_j \mathcal{N}(d_i \mid \mu_j, \sigma_j^2)}{\sum_{l=1}^{m} \pi_l \mathcal{N}(d_i \mid \mu_l, \sigma_l^2)}. \tag{3}$$

M-step update the parameters:

$$\pi_j^{\text{new}} = \frac{1}{k}\sum_{i=1}^{k} \gamma_{ij},\ \mu_j^{\text{new}} = \frac{\sum_{i=1}^{k} \gamma_{ij} d_i}{\sum_{i=1}^{k} \gamma_{ij}}, \tag{4}$$

$$\sigma_j^{2\,\text{new}} = \frac{\sum_{i=1}^{k} \gamma_{ij}(d_i - \mu_j^{\text{new}})^2}{\sum_{i=1}^{k} \gamma_{ij}}. \tag{5}$$

The E-step and M-step are alternated until convergence to a local maximum of the likelihood.

To determine the optimal number of components m, we employ the BIC, defined as:

$$\text{BIC}(m) = -2\log \mathcal{L}_m + m \log k, \tag{6}$$

The optimal number of components m^* is thus selected as $m^* = \arg\min_m \text{BIC}(m)$. Once the GMM is fitted with m^* components, each Gaussian component $\mathcal{N}(\mu_j, \sigma_j^2)$ provides a center μ_j and a standard deviation σ_j. We define the ID boundary as the interval $[\mu_j - n\sigma_j, \mu_j + n\sigma_j]$.

At inference time, given a new input pair (s_t, x), the GUI agent computes the embedding $e_t = l_e(s_t, x)$ and its distance to the centroid $d_t = \|e_t - \mu\|_2$.

To determine whether (s_t, x) is ID, we check whether d_t falls within any of the ID boundaries defined by the fitted

Model	Smartphone		Computer		Web browser	
	Accuracy(%)↑	F1 Score(%)↑	Accuracy(%)↑	F1 Score(%)↑	Accuracy(%)↑	F1 Score(%)↑
UI-TARS-7B	97.94	96.55	83.22	89.58	98.28	98.97
Qwen2-VL-2B	98.64	97.68	88.41	92.56	**100.0**	**100.0**
Qwen2-VL-7B	99.51	99.16	87.63	92.10	**100.0**	**100.0**
Qwen2.5-VL-3B	93.77	90.22	88.44	92.58	**100.0**	**100.0**
Qwen2.5-VL-7B	**99.78**	**99.61**	**89.97**	**93.50**	**100.0**	**100.0**
OS-Atlas-Base-7B	33.08	46.19	74.29	84.87	83.02	90.72
OS-Atlas-Pro-7B	33.08	46.19	74.29	84.87	83.02	90.72
LLaVA1.5	28.83	44.66	72.44	83.96	83.11	90.77
Blip + BERT	30.04	45.06	77.81	86.66	88.38	93.45

Table 3: The performance of GEM with different encoder structures.

GMM components. Formally, the OOD detection function $f_{\text{OOD}}(s_t, x)$ is given by:

$$f_{\text{OOD}}(s_t, x) = \begin{cases} 0, & \text{if } \exists j \text{ s.t. } d_t \in [\mu_j - n\sigma_j, \mu_j + n\sigma_j], \\ 1, & \text{otherwise.} \end{cases} \tag{7}$$

If $f_{\text{OOD}}(s_t, x) = 1$, the input is classified as OOD, and the agent immediately terminates execution and notifies the user. Otherwise, the agent proceeds with its normal action prediction and execution.

Experiments

In this section, we first introduce the experimental setup, followed by a presentation and analysis of the performance of GEM on OOD detection for GUI agents.

Experiments Setup

Datasets. For the ID dataset, we use the AITZ (Zhang et al. 2024b) dataset, which contains GUI agent data covering more than 70 Android app scenarios. For the OOD datasets, we select eight GUI agent datasets spanning three platforms: smartphone, computer, and web browser. The smartphone platform includes AndroidControl (Li et al. 2024), OS-Kairos (Cheng et al. 2025b), Meta-GUI (Sun et al. 2022), and ScreenSpot-Mobile (Li et al. 2025), while the computer platform includes Omniact-Desktop (Kapoor et al. 2024) and ScreenSpot-Desktop, and the web browser platform includes Omniact-Web and ScreenSpot-Web.

Implementation. Popular GUI Agents (Wu et al. 2024; Qin et al. 2025; Xu et al. 2024) are developed based on Qwen2-VL-7B. To simulate the construction process of a popular GUI Agent, we perform SFT on Qwen2-VL-7B using AITZ train dataset, and then evaluate its OOD detection performance on each sample from OOD datasets and AITZ test dataset (ID).

Main Results

Table 2 shows the main results. GEM consistently outperforms existing methods across almost all datasets. Although on the Omniact-Desktop dataset the accuracy is 1.62% lower than the best-performing baseline, our method achieves substantial improvements in other metrics. This shows a better balance between rejecting OOD samples and retaining ID samples, which is critical for reliable OOD detection. On other seven datasets, GEM outperforms all baselines across all evaluation metrics.

Because GUI agents exhibit relatively simple reasoning paths, limiting the effectiveness of methods like the TV score, which rely on modeling diverse reasoning paths. And the complex semantics of multimodal inputs weakens the performance of other embedding-based approaches. Furthermore, due to the semantic similarity in the output space, uncertainty-based OOD detection methods for GUI agents are unreliable. In contrast, GEM effectively captures subtle high-dimensional differences between ID and OOD data, leading to strong and robust detection.

Further Analysis

In this section, we present several interesting observations and propose a potential method for using GEM to enhance end-to-end tasks in GUI agents.

Generalization Experiment

To demonstrate the generalization ability of GEM, we evaluate its performance across nine different GUI agents or MLLMs. As shown in Table 3, GEM consistently achieves high accuracy across five models spanning three platforms. For the results where GEM performs suboptimally, OS-Atlas were exposed to most of the smartphone datasets used in our experiment during their GUI grounding pretraining phase. Meanwhile, LLaVA-1.5 and BLIP capture visual information at a coarser granularity. This highlights their limitations in handling the complex embedding space specific to GUI agent tasks. This also indicates that extrapolation-OOD scenarios are more challenging for OOD detection than internalization-OOD scenarios.

Layer-Level OOD Detection Analysis

We evaluated the effectiveness of representations extracted from each of the twenty-eight layers of Qwen2-VL-7B for OOD detection using an embedding-based approach. As shown in Figure 4, in most datasets, the AUROC first increases as the layer depth increases, then decreases. However, the AUROC rises again in the last two layers. Interest-

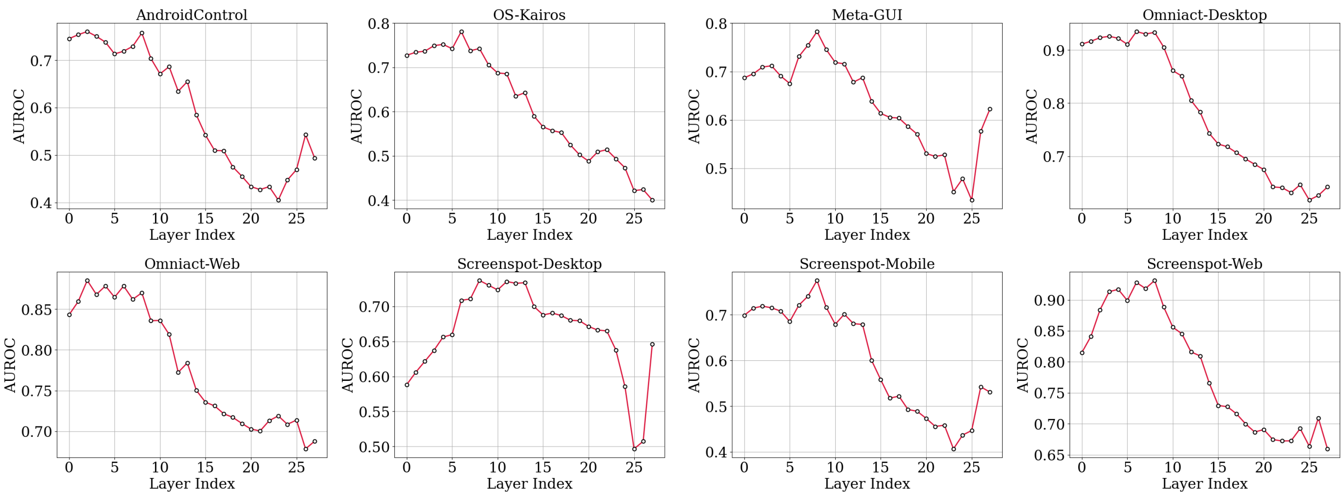

Figure 4: AUROC for OOD detection using embeddings from different layers of the GUI agent.

Mode	SCROLL(%)↑	PRESS(%)↑	STOP(%)↑	CLICK(%)↑	TYPE(%)↑	TOTAL(%)↑
Zero-Shot	19.24	32.05	0.00	30.00	44.59	26.94
SFT	63.11	32.60	65.35	45.14	44.59	49.38
SFT + GEM	63.11$_{0.00\uparrow}$	35.00$_{2.40\uparrow}$	73.20$_{28.06\uparrow}$	55.40$_{10.26\uparrow}$	52.59$_{8.00\uparrow}$	58.78$_{9.40\uparrow}$

Table 4: Analysis of experiments to use GEM to improve the end-to-end results. We report the single-step action accuracy of GUI agents for different action types and the total.

ingly, in 4 out of 8 datasets, the best OOD detection results are achieved at the ninth layer. This might be because, as the layers get deeper, the importance of specific task-related features increases, while the contribution of general visual or textual features decreases. Around the ninth layer, the GUI agent likely finds a best balance. And the GUI agent has developed a kind of confidence estimation in its final output in the final two layeers, allowing it to better distinguish between OOD and ID samples.

How to use GEM to improve the end-to-end results

GEM not only enhances the security of GUI agents but also improves the end-to-end results of GUI agents by combining it with other modules. For example, we trained Qwen2-VL-7B using the AITZ dataset and conducted tests using a mixed test set of AITZ and OS-Kairos to simulate a real GUI agent testing environment with multiple knowledge sources. When GEM determines that the current sample is an OOD sample, it requests assistance from the cloud-based GPT-4o to execute tasks. We report the step-wise success rates for different types of GUI agents as well as the total success rate. The experimental results are shown in Table 4, indicating that GEM can enhance the step-wise success rate of GUI agents by 9.4%, demonstrating the potential of GEM to improve the end-to-end results.

Time Cost of GEM

Although GEM calculates the distances in the embedding space multiple times, it does not actually incur a burden on the time overhead. We conducted experiments on the time cost of GEM, and the experimental results are shown in Table 5. The experiments indicate that GEM only increases training time by 4.9% and testing time by 6.5%.

	Training time	Testing time
Original time	3.63 h	0.77 s
Time with GEM	0.18 h	0.05 s
Additional time (%)	4.9	6.5

Table 5: Time Comparison of GUI Agent w/ and w/o GEM.

Conclusion

We present the first systematic analysis of OOD detection for GUI agents and show that traditional OOD detection methods perform suboptimally in this domain. Then we propose GEM, a method that uses input embedding information to fit a GMM to detect OOD samples for GUI agents. Experiments show that GEM achieves an average accuracy improvement of 23.70% over the best-performing baseline while only increasing training time by 4.9% and testing time by 6.5%. And we present several interesting observations and propose a potential method for using GEM to enhance end-to-end tasks in GUI agents.

Acknowledgments

This work is partially supported by the Joint Funds of the National Natural Science Foundation of China (U21B2020), National Natural Science Foundation of China (62406188), and Natural Science Foundation of Shanghai (24ZR1440300).

References

Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F. L.; Almeida, D.; Altenschmidt, J.; Altman, S.; Anadkat, S.; et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Cheng, P.; Wu, Z.; Wu, Z.; Zhang, A.; Zhang, Z.; and Liu, G. 2025a. OS-Kairos: Adaptive Interaction for MLLM-Powered GUI Agents. *arXiv preprint arXiv:2503.16465*.

Cheng, P.; Wu, Z.; Wu, Z.; Zhang, A.; Zhang, Z.; and Liu, G. 2025b. OS-Kairos: Adaptive Interaction for MLLM-Powered GUI Agents. *arXiv preprint arXiv:2503.16465*.

Cui, P.; and Wang, J. 2022. Out-of-distribution (OOD) detection based on deep learning: A review. *Electronics*, 11(21): 3500.

Dong, H.; Zhao, Y.; Chatzi, E.; and Fink, O. 2024. Multiood: Scaling out-of-distribution detection for multiple modalities. *arXiv preprint arXiv:2405.17419*.

Fluss, R.; Faraggi, D.; and Reiser, B. 2005. Estimation of the Youden Index and its associated cutoff point. *Biometrical Journal: Journal of Mathematical Methods in Biosciences*, 47(4): 458–472.

Gal, Y.; and Ghahramani, Z. 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, 1050–1059. PMLR.

Guo, Y.; Miao, T.; Wu, Z.; Cheng, P.; Zhou, M.; and Zhang, Z. 2025. Atomic-to-Compositional Generalization for Mobile Agents with A New Benchmark and Scheduling System. *arXiv preprint arXiv:2506.08972*.

Hendrycks, D.; and Gimpel, K. 2016. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*.

Hong, W.; Wang, W.; Lv, Q.; Xu, J.; Yu, W.; Ji, J.; Wang, Y.; Wang, Z.; Dong, Y.; Ding, M.; et al. 2024. Cogagent: A visual language model for gui agents. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 14281–14290.

Hu, Z.; Xiong, S.; Zhang, Y.; Ng, S.-K.; Luu, A. T.; An, B.; Yan, S.; and Hooi, B. 2025. Guiding VLM Agents with Process Rewards at Inference Time for GUI Navigation. *arXiv preprint arXiv:2504.16073*.

Jiang, W.; Zhuang, Y.; Song, C.; Yang, X.; Zhou, J. T.; and Zhang, C. 2025. AppAgentX: Evolving GUI Agents as Proficient Smartphone Users. *arXiv preprint arXiv:2503.02268*.

Kapoor, R.; Butala, Y. P.; Russak, M.; Koh, J. Y.; Kamble, K.; AlShikh, W.; and Salakhutdinov, R. 2024. Omniact: A dataset and benchmark for enabling multimodal generalist autonomous agents for desktop and web. In *European Conference on Computer Vision*, 161–178. Springer.

Kervadec, C.; Antipov, G.; Baccouche, M.; and Wolf, C. 2021. Roses are red, violets are blue... but should VQA expect them to? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2776–2785.

Lee, K.; Lee, K.; Lee, H.; and Shin, J. 2018. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. *Advances in neural information processing systems*, 31.

Li, J.; Li, D.; Savarese, S.; and Hoi, S. 2023. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, 19730–19742. PMLR.

Li, K.; Meng, Z.; Lin, H.; Luo, Z.; Tian, Y.; Ma, J.; Huang, Z.; and Chua, T.-S. 2025. Screenspot-pro: Gui grounding for professional high-resolution computer use. *arXiv preprint arXiv:2504.07981*.

Li, W.; Bishop, W.; Li, A.; Rawles, C.; Campbell-Ajala, F.; Tyamagundlu, D.; and Riva, O. 2024. On the Effects of Data Scale on Computer Control Agents. *arXiv preprint arXiv:2406.03679*.

Liu, G.; Zhao, P.; Liu, L.; Chen, Z.; Chai, Y.; Ren, S.; Wang, H.; He, S.; and Meng, W. 2025a. LearnAct: Few-Shot Mobile GUI Agent with a Unified Demonstration Benchmark. *arXiv preprint arXiv:2504.13805*.

Liu, H.; Li, C.; Wu, Q.; and Lee, Y. J. 2023. Visual instruction tuning. *Advances in neural information processing systems*, 36: 34892–34916.

Liu, W.; Wang, X.; Owens, J.; and Li, Y. 2020. Energy-based out-of-distribution detection. *Advances in neural information processing systems*, 33: 21464–21475.

Liu, Y.; Li, P.; Wei, Z.; Xie, C.; Hu, X.; Xu, X.; Zhang, S.; Han, X.; Yang, H.; and Wu, F. 2025b. InfiGUIAgent: A Multimodal Generalist GUI Agent with Native Reasoning and Reflection. *arXiv preprint arXiv:2501.04575*.

Lu, S.; Wang, Y.; Sheng, L.; Zheng, A.; He, L.; and Liang, J. 2024. Recent Advances in OOD Detection: Problems and Approaches. *arXiv preprint arXiv:2409.11884*.

Luan, S.; Gu, Z.; Freidovich, L. B.; Jiang, L.; and Zhao, Q. 2021. Out-of-distribution detection for deep neural networks with isolation forest and local outlier factor. *IEEE Access*, 9: 132980–132989.

Ma, X.; Zhang, Z.; and Zhao, H. 2024. CoCo-Agent: A Comprehensive Cognitive MLLM Agent for Smartphone GUI Automation. In Ku, L.-W.; Martins, A.; and Srikumar, V., eds., *Findings of the Association for Computational Linguistics: ACL 2024*, 9097–9110. Bangkok, Thailand: Association for Computational Linguistics.

Malinin, A.; and Gales, M. 2018. Predictive uncertainty estimation via prior networks. *Advances in neural information processing systems*, 31.

Metz, C. E. 1978. Basic principles of ROC analysis. *Seminars in Nuclear Medicine*, 8(4): 283–298.

Ming, Y.; Cai, Z.; Gu, J.; Sun, Y.; Li, W.; and Li, Y. 2022. Delving into out-of-distribution detection with vision-language representations. *Advances in neural information processing systems*, 35: 35087–35102.

Neath, A. A.; and Cavanaugh, J. E. 2012. The Bayesian information criterion: background, derivation, and applications. *Wiley Interdisciplinary Reviews: Computational Statistics*, 4(2): 199–203.

Qin, Y.; Ye, Y.; Fang, J.; Wang, H.; Liang, S.; Tian, S.; Zhang, J.; Li, J.; Li, Y.; Huang, S.; et al. 2025. UI-TARS: Pioneering Automated GUI Interaction with Native Agents. *arXiv preprint arXiv:2501.12326*.

Ren, J.; Liu, P. J.; Fertig, E.; Snoek, J.; Poplin, R.; Depristo, M.; Dillon, J.; and Lakshminarayanan, B. 2019. Likelihood ratios for out-of-distribution detection. *Advances in neural information processing systems*, 32.

Ren, J.; Luo, J.; Zhao, Y.; Krishna, K.; Saleh, M.; Lakshminarayanan, B.; and Liu, P. J. 2022. Out-of-distribution detection and selective generation for conditional language models. *arXiv preprint arXiv:2209.15558*.

Reynolds, D. A.; et al. 2009. Gaussian mixture models. *Encyclopedia of biometrics*, 741(659-663): 3.

Shi, Y.; Yu, W.; Yao, W.; Chen, W.; and Liu, N. 2025. Towards trustworthy gui agents: A survey. *arXiv preprint arXiv:2503.23434*.

Sun, L.; Chen, X.; Chen, L.; Dai, T.; Zhu, Z.; and Yu, K. 2022. META-GUI: Towards Multi-modal Conversational Agents on Mobile GUI. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, 6699–6712.

Tang, F.; Gu, Z.; Lu, Z.; Liu, X.; Shen, S.; Meng, C.; Wang, W.; Zhang, W.; Shen, Y.; Lu, W.; Xiao, J.; and Zhuang, Y. 2025a. GUI-G^2: Gaussian Reward Modeling for GUI Grounding. arXiv:2507.15846.

Tang, F.; Xu, H.; Zhang, H.; Chen, S.; Wu, X.; Shen, Y.; Zhang, W.; Hou, G.; Tan, Z.; Yan, Y.; Song, K.; Shao, J.; Lu, W.; Xiao, J.; and Zhuang, Y. 2025b. A Survey on (M)LLM-Based GUI Agents. arXiv:2504.13865.

Wang, H.; Li, Y.; Yao, H.; and Li, X. 2023a. Clipn for zero-shot ood detection: Teaching clip to say no. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 1802–1812.

Wang, W.; Bao, H.; Dong, L.; Bjorck, J.; Peng, Z.; Liu, Q.; Aggarwal, K.; Mohammed, O. K.; Singhal, S.; Som, S.; et al. 2023b. Image as a foreign language: Beit pretraining for vision and vision-language tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 19175–19186.

Wang, Y.; Zhang, P.; Yang, B.; Wong, D.; Zhang, Z.; and Wang, R. 2024. Embedding trajectory for out-of-distribution detection in mathematical reasoning. *Advances in Neural Information Processing Systems*, 37: 42965–42999.

Wang, Z.; Xu, H.; Wang, J.; Zhang, X.; Yan, M.; Zhang, J.; Huang, F.; and Ji, H. 2025. Mobile-Agent-E: Self-Evolving Mobile Assistant for Complex Tasks. *arXiv preprint arXiv:2501.11733*.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Xia, F.; Chi, E.; Le, Q. V.; Zhou, D.; et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35: 24824–24837.

Wu, Q.; Jiang, H.; Yin, H.; Karlsson, B. F.; and Lin, C.-Y. 2022. Multi-level knowledge distillation for out-of-distribution detection in text. *arXiv preprint arXiv:2211.11300*.

Wu, Z.; Cheng, P.; Wu, Z.; Ju, T.; Zhang, Z.; and Liu, G. 2025a. Smoothing Grounding and Reasoning for MLLM-Powered GUI Agents with Query-Oriented Pivot Tasks. *arXiv preprint arXiv:2503.00401*.

Wu, Z.; Huang, H.; Lou, X.; Qu, X.; Cheng, P.; Wu, Z.; Liu, W.; Zhang, W.; Wang, J.; Wang, Z.; et al. 2025b. Verios: Query-driven proactive human-agent-gui interaction for trustworthy os agents. *arXiv preprint arXiv:2509.07553*.

Wu, Z.; Huang, H.; Yang, Y.; Song, Y.; Lou, X.; Liu, W.; Zhang, W.; Wang, J.; and Zhang, Z. 2025c. Quick on the Uptake: Eliciting Implicit Intents from Human Demonstrations for Personalized Mobile-Use Agents. *arXiv preprint arXiv:2508.08645*.

Wu, Z.; Wu, Z.; Xu, F.; Wang, Y.; Sun, Q.; Jia, C.; Cheng, K.; Ding, Z.; Chen, L.; Liang, P. P.; et al. 2024. Os-atlas: A foundation action model for generalist gui agents. *arXiv preprint arXiv:2410.23218*.

Xia, X.; and Luo, R. 2025. GUI-R1: A Generalist R1-Style Vision-Language Action Model For GUI Agents. *arXiv preprint arXiv:2504.10458*.

Xu, Y.; Wang, Z.; Wang, J.; Lu, D.; Xie, T.; Saha, A.; Sahoo, D.; Yu, T.; and Xiong, C. 2024. Aguvis: Unified Pure Vision Agents for Autonomous GUI Interaction. *arXiv preprint arXiv:2412.04454*.

Yang, J.; Zhou, K.; Li, Y.; and Liu, Z. 2024. Generalized out-of-distribution detection: A survey. *International Journal of Computer Vision*, 132(12): 5635–5662.

Yao, S.; Zhao, J.; Yu, D.; Du, N.; Shafran, I.; Narasimhan, K.; and Cao, Y. 2023. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.

Zhang, C.; He, S.; Qian, J.; Li, B.; Li, L.; Qin, S.; Kang, Y.; Ma, M.; Lin, Q.; Rajmohan, S.; et al. 2024a. Large language model-brained gui agents: A survey. *arXiv preprint arXiv:2411.18279*.

Zhang, J.; Wu, J.; Yihua, T.; Liao, M.; Xu, N.; Xiao, X.; Wei, Z.; and Tang, D. 2024b. Android in the Zoo: Chain-of-Action-Thought for GUI Agents. In Al-Onaizan, Y.; Bansal, M.; and Chen, Y.-N., eds., *Findings of the Association for Computational Linguistics: EMNLP 2024*, 12016–12031. Miami, Florida, USA: Association for Computational Linguistics.

Zhang, Z.; and Zhang, A. 2024. You Only Look at Screens: Multimodal Chain-of-Action Agents. In Ku, L.-W.; Martins, A.; and Srikumar, V., eds., *Findings of the Association for Computational Linguistics: ACL 2024*, 3132–3149. Bangkok, Thailand: Association for Computational Linguistics.

Zhou, Y.; Bai, H.; Cemri, M.; Pan, J.; Suhr, A.; Levine, S.; and Kumar, A. 2024. DigiRL: Training In-The-Wild Device-Control Agents with Autonomous Reinforcement Learning. In *Automated Reinforcement Learning: Exploring Meta-Learning, AutoML, and LLMs*.