ShowUI: One Vision-Language-Action Model for Generalist GUI Agent

Kevin Qinghong Lin[1], **Linjie Li**[2], **Difei Gao**[1], **Zhengyuan Yang**[2],
Zechen Bai[1], **Stan Weixian Lei**[1], **Lijuan Wang**[2], **Mike Zheng Shou**[1]
[1]Show Lab, National University of Singapore [2]Microsoft

Abstract

Graphical User Interface (GUI) automation holds significant promise for enhancing human productivity by assisting with digital tasks. While recent Large Language Models (LLMs) and Large Multimodal Models (LMMs) have been used to build autonomous agents capable of solving complex tasks, they often rely on closed-source, API-based solutions and exhibit limitations in GUI-specific interactions. Inspired by the success of Vision-Language-Action (VLA) models in embodied environments, we explore their potential in the digital GUI world. In this work, we develop a recipe for training a VLA for GUI agent –ShowUI, a 4.2B parameter model based on Phi-3.5-vision-instruct. By leveraging scalable GUI visual data (*e.g.,* screenshots with action trajectory), we aim to develop a generalist GUI agent that demonstrates capabilities across diverse dimensions: grounding, navigation, understanding. ShowUI supports various platforms—including websites, desktops, and mobile phones—and accommodates diverse visual inputs such as single-frame images, multiple frames, and videos. We show that ShowUI achieves significant results across multiple benchmarks, including Screenspot, Mind2Web, AITW, AITZ, GUI-Odyssey, and GUI-World. We provide extensive experiments to analyze the impact of different types of training corpus and model design decisions on downstream tasks. The model, code and data will be open-sourced at `https://github.com/showlab/ShowUI`.

1 Introduction

In the digital age, individuals rely on screens for a vast array of activities (*e.g.,*web browsing, entertainment, *etc*). These activities often necessitate the use of diverse software applications, which are accessed primarily through Graphical User Interfaces (GUIs). Large Language Models (LLMs)[1], which excel in understanding complex language instructions and integrating various tools seamlessly, have shown great potential in GUI automation [2, 3, 4, 5]. They can streamline the navigation of digital interfaces and significantly enhance productivity, *e.g.,*assisting online shopping in website [6] with user textual instruction.

Recently, notable efforts have been made in GUI automation evaluation, benchmarking model performances on the Web [4, 6, 7] or smartphone GUI navigation [8, 9]. They collect the task-required screenshots or HTML codes [10, 11], aiming to recover the scenario and evaluate models in static offline settings. However, these setups are hard to model and fail to capture the task complexity in a realistic simulator. To overcome this limitation, several efforts have developed executable environments with well-defined action spaces, which remove dependencies on pre-defined inputs. They benchmark agents by enveloping a real computer [12, 13] or mobile phone environment [14], fully capturing the task complexity in a realistic simulator. In these benchmarks, the baselines mainly rely on closed-source, API-based LMMs such as GPT-4o, connected with external tools (*e.g.,* OCR or Set-of-Mark[15]). Recognizing the limitations of existing Vision-Language Models—specifically,

Technical report. Work-in-progress

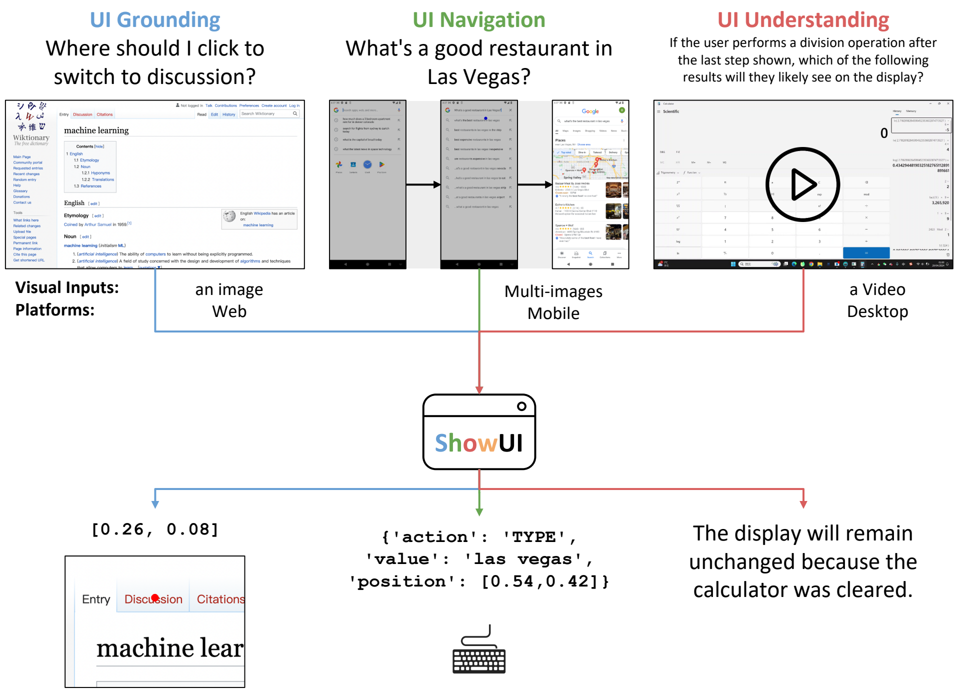


Figure 1: ShowUI: a Vision-Language-Action Model Model building in digital environment, aiming to serve as a Generalist GUI Agent.

the gap between natural visual perception and the unique capabilities required for GUIs, researchers have attempted to **train capable vision-language models to develop agents instead of relying on API-based solutions.** Works such as [16, 17] use massive static screenshots to improve the LMMs' grounding ability. Further efforts leverage navigation action sequence, either generated by prompting LLM [18] or collected from real-world user trajectories [19]. Others have begun to explore action understanding from action recordings [20] or web instructional videos [21].

In contrast, in the embodied world, *e.g.,* robotics [22, 23, 24], autonomous driving [25] and gaming [26, 27], has seen rich exploration of connecting Vision-Language Models with action control, namely Vision-Language-Action (VLA) models that enable interaction with the environment, obtaining observations and feedback, performing decision-marking *etc*. One remarkable effort is OpenVLA [28], which thoroughly discusses how to develop a VLA by learning from data and ablates several model decisions, *etc*. Notably, it is paired with open-sourced resources, which greatly empower development in this field. While in the digital world, especially GUIs, this integration and discussion is less explored.

In this work, we share our **exploration of building Vision-Language-Action models aimed at developing a generalist GUI agent**, namely ShowUI–Vision-Language-Action Transformer. Starting from a representative vision-language model (*i.e.,* Phi-3.5-vision-instruct), we document how to prepare and leverage training data and make design decisions to develop a VLA that serves as a generalist GUI agent. We provide extensive experiments and ablation studies to reveal several key observations and insights. More importantly, our models, codebase will be *fully open-sourced* to empower community development.

2 Related Works

2.1 Vision-Language-Action Models

Vision-Language Models (VLMs) have recently made significant advancements, processing both visual and textual data [29, 30]. GPT-4V [31] integrates visual perception with language generation to perform tasks such as image captioning [32, 33], visual question answering (VQA)[34], and document analysis[35]. Open-source models like LLaVA [36], mini-GPT4 [37], and Phi-3-vision [38] also show strong results, particularly in OCR-free scene text comprehension. The Qwen-VL series [39, 40]

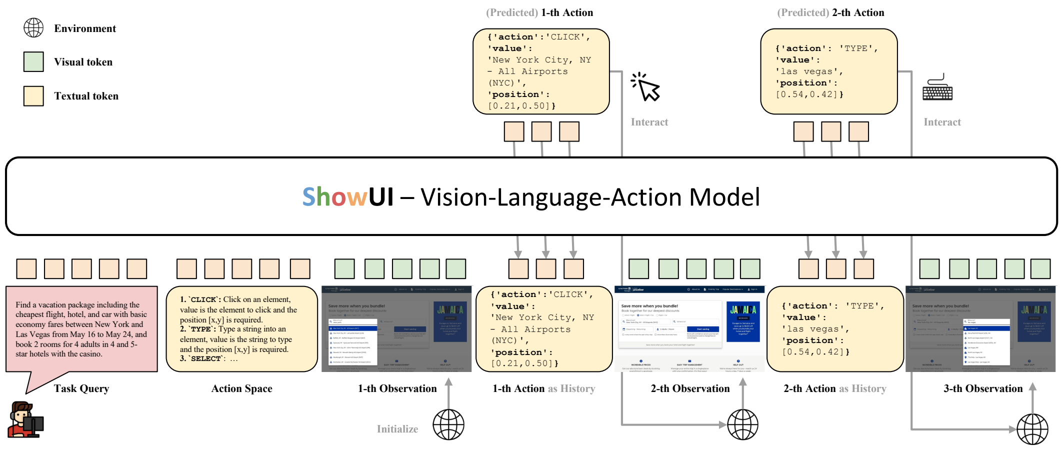


Figure 2: Illustration of our interleaved Vision-Language-Action Model, which accept visual screenshots and predict action in a streaming manner.

adds grounding capabilities, allowing models to localize image regions based on language input. These newly added capabilities enable models to better observe the environment and generate actions accordingly. Thus, Vision-Language-Action (VLA) models extend VLMs to perform actions in natural environments, including RT-2-X [41] and OpenVLA [42]. Additionally, some models [16, 43] have been proposed to generate actions in GUI environments. GUI-based tasks, which often require understanding high-resolution images, are addressed by models like [17, 16], enabling interaction with GUIs through visual grounding, thereby improving accessibility and widget localization.

Despite the notable progress made by the aforementioned works, previous GUI-related VLA models typically only possess limited UI element grounding or question-answering capabilities. This work is the first to systematically compile existing open-source databases, construct corresponding instructional training data, and achieve outstanding performance across various types of tasks, including grounding, navigation, understanding, and planning through training.

2.2 GUI Automation

Recent works on GUI automation have explored a range of environments and tasks. Typical tasks include web-based interaction [6, 7, 4, 44, 45], where efforts have focused on training agents to navigate web interfaces and perform activities such as online shopping [44]. Similarly, efforts in mobile interface navigation have aimed to improve accessibility by developing agents that operate within mobile platforms, including Android and iOS simulators [8, 46]. Additionally, desktop software environments have seen advancements in grounding UI elements within offline contexts, such as analyzing screenshots [17].

In terms of methodology, one branch of work primarily proposes agentic reasoning frameworks for GUI automation [12, 47, 48, 49], which utilize multimodal LLMs and external tools to accomplish complex GUI tasks. Another branch focuses more on building a single VLM model to ground UI elements and then generate actions, as seen in models like SeeClick [17] and Ferret-UI [50]. Our work belongs to the latter category, and we train a model with a broader range of capabilities.

3 Recipe for Training VLA as GUI Agents

3.1 Vision-Language Models as a Starting Point

We develop Vision-Language-Action Model for the GUI Agent by building on top of existing open-source Vision-Language Models, which provide a good foundation, particularly for general visual and textual understanding. We select VLMs guided by the following considerations:

1. **High-Resolution with Enhanced OCR Capability:** Given our focus on screen visual perception, the model must support high-resolution inputs, as typical desktop or web resolutions range from 2K to 4K. This is essential for accurately capturing detailed content. Additionally, Optical Character Recognition (OCR) is crucial, as many interactive elements on GUIs contain textual indicators.

2. **Lightweight Model Sizes:** The model should be efficient enough for deployment on PCs and mobile devices, requiring fast response times. Hence, smaller model sizes (*e.g.,* less than 7B) are preferred to ensure efficient operation on various platforms.
3. **Interleaved Visual Inputs:** In GUI-based tasks, the agent needs to recognize GUI elements and handle navigation by processing both past trajectory history, which can include not only textual information but also visual screenshots. Thus, support for multiple frames is important. Furthermore, in some scenarios, the agent must process dynamic visual inputs, such as video clips demonstrating actions. Therefore, the model should support interleaved single-frame, multi-frames, and video inputs.
4. **Long-Context Support:** The agent may encounter novel software that it has not seen during training, making it necessary to handle unfamiliar scenarios. In such cases, providing tutorial documents or guidelines could be beneficial. This long-context support is essential for future test-time Retrieval-Augmented Generation (RAG) considerations.

Based on these considerations, we summarize recent Vision-Language Models in Tab. 1 and we started by Phi-3.5-vision-instruct for GUI Agent development.

Models	Date	Model size	Image Res.	Interleaved V.	TextVQA	# Ctx. Len.
MiniCPM-V-2	Apr 2024	2.8B	Dyn.	✓	74.1	4K
InternVL2-4B	Jul 2024	4.2B	Dyn.	✓	74.4	128K
Phi-3.5-V	Aug 2024	4.2B	Dyn.	✓	72.0	128K
LLaVA-OV	Aug 2024	0.5B	Dyn.	✓	65.9	32K
Qwen2-VL	Aug 2024	2B	Dyn.	✓	79.7	32K

Table 1: Recent mainstream vision-language models that support (dynamic) high resolution, strong OCR (TextVQA), interleaved visual inputs and long context length.

3.2 Training Data and Design Decision

We begin by identifying fundamental capabilities required to meet practical needs in GUI Agent.

3.2.1 Grounding

The most frequent actions in GUI are `Click` and `Tap`, both of which require precise coordinates, to locate the desired element within a large screenshot. Training data for GUI grounding is typically divided into two categories: text elements (OCR) and visual elements (icons and widgets). OCR data can be easily scaled by scraping text from websites using HTML annotations [17, 18]. In contrast, visual elements are more costly to obtain, with the majority coming from mobile devices, particularly Android [51, 52]. Previous methods [53, 13] have used OCR or SoM [15] to provide an approximate estimation using discrete marker indices. We hope to build a agent with inherited grounding abilities. So, we instruct large multimodal models (LMMs) to output coordinates in textual form as [17, 16].

To enable effective and efficient training, we implement the following schemes: ***(i)* Flexible usage.** Grounding annotations are typically provided as bounding boxes, while actions are represented as point-wise coordinates, *i.e.,* $[\mathbf{x}, \mathbf{y}]$. To enhance flexibility, we randomly augment each bounding box into one of four modes: Text2Point (the primary mode, estimating the bounding box center), Text2Bbox, Point2Text, and Bbox2Text. The latter three modes are particularly useful for recognition tasks or self-labeling data. ***(ii)* Multi-round forward.** Grounding annotations are dense, often containing hundreds of elements per screen, with relatively short text and coordinate data compared to the visual tokens (*e.g.,* 2.4K tokens per image in Phi-3.5-vision-instruct). To optimize training efficiency, we predict multiple annotations for each instance within a single forward pass. This strategy is depicted in Fig. 3. ***(iii)* Diverse resolutions.** Existing screenshots often suffer from resolution bias (*e.g.,*, all web screenshots in [17] are of uniform size). To simulate a range of screen resolutions, from small mobile displays to 4K, and better reflect real-world variability, we apply a data augmentation strategy that includes random cropping (down to $0.5\times$) and padding (up to $1.5\times$).

3.2.2 Navigation

What shell be the best representation for action; Do we need action tokens? json is good choices; The core ability of the agent is navigation, which is conditioned on a user query and requires the agent to jointly predict: (i) the correct action type (*e.g.,* `Click` or `Type`), and (ii) the action parameters (*e.g.,* `Click` by coordinates or `Type` by entering a string). Furthermore, since full navigation typically involves multi-step trajectories, the agent must also account for past actions and

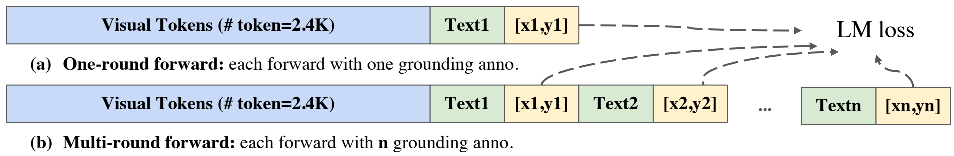


Figure 3: Illustration of multi-round forward pass. Blue represents the visual tokens, which constitute the majority of the total tokens. Green represents the text tokens corresponding to element names, while yellow represents the coordinate tokens, used for loss calculation.

observations. Navigation data is valuable because it simulates human trajectories. One reliable but costly method is to collect human demonstrations [10, 19]. A more efficient approach is to leverage LLMs with annotations to synthesis navigation episodes [18].

One challenge in navigation is the variation in action spaces across different devices and systems, such as: ***(i)*** **Actions are device-specific** (*e.g.*,, `Click` is absent on mobile, while `Press Home` does not exist on the web). ***(ii)*** **Same action but different parameters** (*e.g.*,, `Scroll` has two directions—up and down—on the web, but four directions on mobile). ***(iii)*** **Novel actions at test time** that do not appear during training. To accommodate these variations in a single GUI agent, we structure the output for each action in a JSON-like format (*i.e.*, `{'action':'action_type', 'value':'element', 'position':[x,y]}`), where `[x,y]` represents coordinates on a relative scale. Then, we unify the navigation task prompt as:

> You are an assistant trained to navigate the `{device}` . Given a task instruction, a screen observation, and an action history sequence, output the next action and wait for the next observation.
>
> Here is the action space:
> *# templated by action_type with action description.*
> 1. 'CLICK': Click on an element, value is the element to click and the position [x,y] is required.
> 2. 'TYPE': Type a string into an element, value is the string to type and the position [x,y] is not applicable.
> ...
>
> Format the action as a dictionary with the following keys:
> {'action':'action_type', 'value':'element', 'position':[x,y]}
> Position represents the relative coordinates on the screenshot and should be scaled to a range of 0-1.
>
> Task: `{task}`
> Action History:
> Step 1: `<|image_1|>{past_action_1}`
> Step 2: `<|image_2|>{past_action_2}`
> ...
> Step n: `<|image_n|>{past_action_n}`
> Observation: `<|image_n+1|>`
> What is the next action?

In this format, the blue variable can be adapted based on the context. This approach requires the agent to understand action definitions in-context, rather than just memorizing previously seen actions. As a result, users can flexibly support navigation at test time (*e.g.*, simply providing clear definitions for novel actions). A key component is the observation, which we formulate as **an interleaved image-text format**. This allows the agent to jointly leverage the past action along with its paired observations. Each `past_action` follows the JSON format we previously defined.

3.2.3 Understanding

The understanding ability serves as an auxiliary skill for the GUI agent to improve visual perception and reasoning. This includes: ***(i)*** **Screen captioning [54]:** Generating brief or detailed captions for the current screen, helping the agent build visual context and interpret the interface more clearly. ***(ii)*** **Visual question-answering [55]:** Responding to specific questions based on visual observations, which is useful when determine whether to stop or continue execution. ***(iii)*** **Multi-turn dialogues [18]:** Engaging in free-form conversations with users, which helps maintain the agent's chatbot capabilities and supports natural interaction.

Dataset	Task	Platform	Vis. Input	# Inst.	# Anno.	# Avg. Act.	# Act. type.
SeeClick [17]	Grounding	Web	Image	270K	3.0M	–	–
RICO [51, 52]	Grounding	Mob$_A$	Image	243K	763K	–	–
GUIEnv [18]	Grounding	Web, Mob$_A$	Image	70K	589K	–	–
AMEX [58]	Grounding	Mob$_A$	Image	97K	885K	–	–
GUIAct [18]	Navigation	Web, Mob$_A$	(multi.) Image	79K	191K	2.4	11
LLaVA-665K [36]	Understanding	–	Image	665K	665K	–	–
AMEX [58]	Under.-Cap.	Mob$_A$	Image	22K	22K	–	–
Screen2words [54]	Under.-Chat	Mob$_A$	Image	79K	79K	–	–
GUIChat [18]	Under.-Chat	Web, Mob$_A$	Image	50K	50K	–	–
xLAM [57]	Fun. Call	–	Text	60K	60K	–	–
Screenspot [17]	Grounding	Web, Desk, Mob$_{IA}$	Image	1.3K	1.3K	–	–
MiniWob [59]	Navigation	Web (syn.)	Image	2.7K	9.8K	3.5	2
Mind2Web [6]	Navigation	Web	(multi.) Image	2.3K	17K	3	3
AITW [10]	Navigation	Mob$_A$	(multi.) Image	4.6K	23.6K	12	12
AITZ [60]	Navigation	Mob$_A$	(multi.) Image	2.5K	18.6K	7.4	10
GUI-Odyssey [19]	Navigation	Mob$_A$	(multi.) Image	7.7K	119K	15.4	9
GUI-World [21]	Understanding	Diverse	Video	12.4K	98K	–	–

Table 2: Dataset statistics. **The upper side** datasets are used for pretraining. **The lower side** datasets are used for downstream adaptation, including training and validation / test sets. Under. – Understanding. Mob. – mobile devices, with subscript A – Android and I – iOS. For the navigation task, we report the average number of actions per task and the number of action categories.

3.2.4 Function Calling

One important ability often overlooked by existing GUI training-based works [16, 17] is the function calling ability. Two main considerations should be taken into account when incorporating this into VLA training: ***(i)* Tradeoff between API calls and atomic actions:** Many advanced functions are already available as convenient pipelines, such as PowerPoint automation with `python-pptx`. Utilizing these APIs is often more efficient than performing actions atomically. This comparison is demonstrated in Fig. 4. ***(ii)* Translating GUI action instructions into executable code:** Currently, most GUI agents need to convert textual action instructions into executable commands, such as python script by `PyAutoGUI` [56], then running on devices. To address these issues, we include textual instruction-tuning data from xLAM [57] to support function calling.

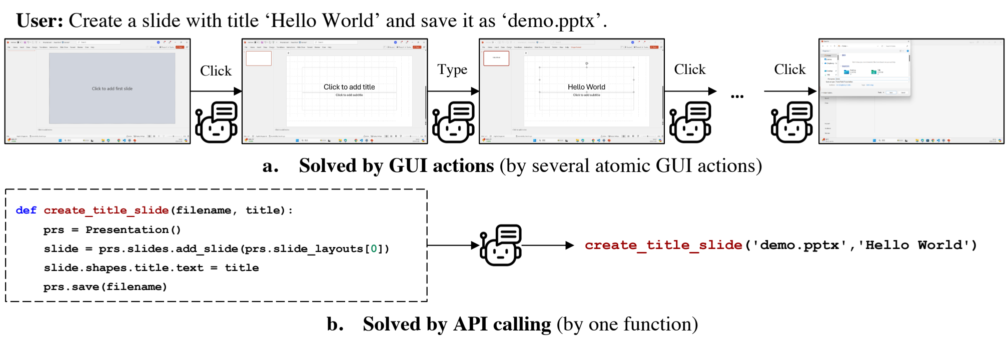


Figure 4: GUI action executions *v.s.* API Function calling for the same user query.

3.3 Training Procedure

In Tab. 2, we categorize the datasets based on their use for either instruction-tuning (upper section) or downstream tasks (lower section). A straightforward approach is to leverage all instruction-tuning data for **pretraining** a general-purpose GUI agent. However, we observed a significant imbalance in the data distribution in terms of task types and platforms, with a large portion of the data focused on grounding tasks. To address this imbalance during pretraining, we adjusted the data sampling strategy to ensure balanced probabilities when sampling from each task category. Additionally, we noticed that most of the training data is sourced from websites and mobile devices (particularly Android), while there is a notable lack of data from desktop and iOS devices.

4 Experiments

4.1 Settings

In the first stage of pretraining, we utilize 64 V100 GPUs, while downstream adaptation is conducted on 8 V100 GPUs. The batch size per GPU is set to 1, with gradient accumulation steps of 2. We use float16 precision for training. To enhance efficiency, we apply LoRA tuning with a rank of 8 and an alpha value of 16 for both the language model and visual encoder, resulting in 0.4% of the total learnable parameters. We also leverage DeepSpeed Zero-2 and use the SDPA attention mechanism. Below, we provide a reference for applying individual techniques on Phi-3.5-vision-instruct to reduce memory usage.

Model size	# Token Len.	LoRA, Zero2, Fp32	+Fp16	+SDPA	+Zero3	+QLoRA
4.2B	2.4K (an image)	32 Gb	17.6 Gb	16.7 Gb	12.2 Gb	8.5 Gb

4.2 Experimental Results

In this section, we structure our experiments on individual downstream tasks to address the following questions: ***(i)*** What are the baseline results on our selected VLMs? What improvements are brought by pretraining, and can they perform zero-shot adaptation? ***(ii)*** Can the abilities learned from specific domains and tasks transfer to other settings? ***(iii)*** How do the models perform on novel tasks and domains that were unseen during training, or on multi-application tasks?

Which training data is missing? We present zero-shot grounding results on the Screenspot [17] benchmark in Tab. 3, which includes three devices and two settings: Text and Icon. This provides a straightforward signal of the shortcomings in each setup. We found: **(a)** While the initial results of Qwen-VL (with grounding training) were higher than Phi-3.5-v, after GUI pretraining, ShowUI significantly outperformed its counterpart—SeeClick. This demonstrates the strong potential of VLMs that meet our criteria in Tab. 1. **(b)** Overall, the text track scores are higher than the icon track, even for desktop text, which was not seen during training. This suggests that text grounding ability learned from web and mobile is transferable across platforms. **(c)** The icon track is more challenging due to its visual-centric grounding. Mobile scores are significantly higher than desktop and web, as nearly all icon/widget training data comes from mobile devices. These results emphasize the **missing of icon/widget grounding data beyond mobile devices**.

Method	PT?	Model size	Mobile		Desktop		Web		**Avg.**
			Text	Icon	Text	Icon	Text	Icon	
Qwen-VL [40]	✗	9.6B	9.5	4.8	5.7	5.0	3.5	2.4	5.2
Phi-3.5-V [38]	✗	4.2B	1.7	1.7	4.1	0.7	4.7	2.0	2.5
Fuyu [61]	✓	8B	41.0	1.3	33.0	3.6	33.9	4.4	19.5
CogAgent [16]	✓	18B	67.0	24.0	74.2	20.0	**70.4**	28.6	47.4
SeeClick [17]	✓	9.6B	78.0	52.0	72.2	30.0	55.7	32.5	53.4
ShowUI	✓	4.2B	**86.1**	**68.1**	**78.8**	**41.7**	68.9	**52.9**	**66.0**

Table 3: *Zero-shot grounding on Screenspot* shows that **icon-based (visual) grounding is more challenging**, especially beyond mobile devices.

Impact by novel task or domains? In Tab. 4 – web navigation Mind2Web [6], we have the following observations: **(a)** Pretraining is significant, brings 6.3% Avg. Step SR. boost over Phi-3.5-vision-instruct. Notably, ShowUI's zero-shot and surpass Qwen-VL with fine-tuning, and achieves relatively high Op. F1 (80%+). **(b)** ShowUI, with fine-tuning, yields comparable scores to MindAct and GPT-4, both of which are text-based agents using HTML inputs. This highlights the potential of GUI visual agents. **(c)** The cross-website and cross-domain settings are harder than cross-tasks. This suggests the **bottleneck is lie in GUI visual perception** (websites/domains are unseen in training) rather than textual task understanding. One future effort for improvement is to **develop training data with good (visually) domain diversity.**

Scores on individual actions? In Tab.5, we examine mobile navigation on AITW[10] and AITZ [60]. Surprisingly, we found that the pretraining gains are minimal in this benchmark compared to web-based tasks. This could be due to the **informative historical context provided in mobile navigation** (# action history 4 steps, # action class 10) compared to web navigation (2 steps, 3 classes), so that pretraining has a weak effect. Notably, we identified a significant issue with ShowUI's zero-shot

Method	PT?	FT?	Cross-Task			Cross-Website			Cross-Domain		
			Ele. Acc	Op.F1	Step SR	Ele. Acc	Op.F1	Step SR	Ele. Acc	Op.F1	Step SR
MindAct [6]	–	–	55.1	75.7	52.0	42.0	65.2	38.9	42.1	66.5	39.6
GPT-4 [1]	–	–	41.6	60.6	36.2	35.8	51.1	30.1	37.1	46.5	26.4
Qwen-VL [40]	✗	✓	15.9	86.7	13.3	13.2	83.5	9.2	14.1	84.3	12.0
SeeClick [17]	✓	✓	28.3	87.0	25.5	21.4	80.6	16.4	23.2	**84.8**	20.8
Phi-3.5-V [38]	✗	✓	42.6	88.9	39.3	36.4	83.5	30.3	36.9	84.4	32.5
ShowUI	✓	✓	**48.3**	**89.1**	**44.5**	**44.6**	**86.1**	**39.4**	**43.9**	84.4	**37.1**
ShowUI-ZS	✓	✗	19.0	83.9	15.2	17.5	80.6	14.5	20.9	81.5	16.2

Table 4: *Web Navigation on Mind2Web.* The top half correspond to methods that required HTML text inputs. We found that **ShowUI's zero-shot performance surpasses Qwen-VL with fine-tuning**. Additionally, **cross-website/domain is harder** than cross-task setting.

Method	PT?	FT?	AITW	AITZ (w. CoAT [60])									GUI-Odyssey	
			Total	Scroll	Click		Type		Press	Stop	Total		Multi-Apps	Avg.
			Match		Class	Match	Class	Match			Class	Match	Match (SR.)	Match (SR.)
Qwen2-VL-72B [62]	–	✗	–	–	–	–	–	–	–	–	89.6	72.1	–	–
GPT-4V [31]	–	✗	50.5	–	–	–	–	–	–	–	–	–	19.0	18.8
GPT-4o [63]	–	✗	–	–	–	–	–	–	–	–	70.0	35.3	16.7	20.4
CogAgent [16]	✓	✗	–	70.2	88.2	66.2	45.8	21.8	24.6	24.6	72.6	53.3	10.7	11.8
AUTO-UI [49]	✓	✓	–	61.4	74.6	32.2	87.4	57.7	74.4	74.4	83.0	47.7	–	–
Qwen-VL [40]	✗	✓	54.3	–	–	–	–	–	–	–	–	–	73.1	72.8
SeeClick [17]	✓	✓	59.3	–	–	–	–	–	–	–	–	–	–	–
OdysseyAgent [19]	✓	✓	–	–	–	–	–	–	–	–	–	–	74.8 (4.4)	74.3 (8.8)
Phi-3.5-V [38]	✗	✓	67.0	90.3	95.6	34.0	97.2	91.0	83.3	93.3	93.8	57.5	87.4	87.7
ShowUI	✓	✓	69.8	91.8	95.8	39.0	95.2	91.6	79.9	92.0	93.5	60.3	86.4 (8.9)	87.5 (23.6)
ShowUI-ZS	✓	✗	28.3	0.8	98.6	3.1	0.0	0.0	0.0	0.0	56.5	1.9	62.4 (0.0)	61.4 (0.0)

Table 5: *Mobile Navigation on AITW, AITZ and GUI-Odyssey.* We found that (i) the overall **gain by pretraining is minor** compared to web-based tasks. (ii) In AITZ, ShowUI's zero-shot performance suffers from significant action bias—the pretrained model **tends to output click actions due to data imbalance**. (iii) In GUI-Odyssey, **Multi-applications with a significant drop in Task SR**, even it has comparable action match with average.

performance: the model heavily favors outputting click actions (96.1% based on statistics), likely due to **action imbalance in the existing pretraining set**, which should be relieved in future.

Is multi-applications challenging? In Tab.5 GUI-Odyssey, we examine the score variations in multi-application settings with average. Despite ShowUI achieves relatively high scores in multi-app contexts (with an action match of 86.4), its Task SR. is only 8.9, significantly lower than the overall average of 23.6. This indicates that multi-app. pose unique challenges that **model struggle with errors during one of the several consecutive steps.**

Dynamic action videos *v.s.* Static screenshots. In Tab.6, we examine GUI video understanding using GUI-World. We found the following: (i) Despite XR, iOS, and other software not being observed during pretraining, ShowUI achieves similar scores to those for websites and Android, suggesting that **knowledge learned from specific scenarios can transfer to novel environments**. (ii) With pretraining on static screenshots, ShowUI achieves 58.2% accuracy, surpassing GUI-Vid even with fine-tuning. This demonstrates that **screenshots pretraining brings a good initialization for dynamic video understanding**. (iii) However, ShowUI shows limited improvement after fine-tuning. This may be due to inconsistencies by crops per frame in video processing, which limits the model's ability to manage resolution size effectively. Overall, the MCQ task is relatively easy and does not effectively differentiate between models. This suggests that the MCQ setting may not be ideal for GUI tasks.

Impact of pretraining with diff. capabilities? we analyze the effects of pretraining with different capabilities on various downstream tasks in Tab. 7. The results indicate that **pretraining with all capabilities yields the best overall performance**, particularly achieving the highest scores in both Grounding and Navigation tasks. Grounding significantly improves navigation, as it is closely tied to core navigation actions (*e.g.,* `Click`). However, we found that understanding data is less helpful for navigation. For the Understanding task (GUI-World), the model pretrained on general-domain multimodal understanding (*i.e.,* Phi-3.5-v) performs best, highlighting that this MCQ setting may not be that effective at reflecting GUI understanding.

Method	FT?	Software	Website	XR	Multi	IOS	Android	**Avg.**
Gemini-Pro-1.5 [64]	✗	81.7	82.6	81.2	81.2	82	81.6	81.7
Qwen-VL-Max [40]	✗	74.9	76.9	74.2	68.8	75.4	73.7	74
GPT-4V [31]	✗	81.5	80.9	80.6	75	82.5	78.3	79.8
GPT-4o [63]	✗	86.5	83.3	84.3	81.1	83.3	90	84.8
ChatUniv [65]	✗	28.4	22.2	20.6	17.5	22.6	23	22.4
Minigpt4Video [66]	✗	18.9	15.3	16.3	15.4	20.1	14.6	16.8
VideoChat2 [67]	✗	45.5	42.6	44	40.4	40.2	44.7	42.9
GUI-Vid [19]	✓	59.9	54.1	55.6	52.9	51.8	53.4	54.6
Phi-3.5-v	✓	85.6	87.2	84.3	86.3	87.6	89.5	87.3
ShowUI	✓	84.6	87.0	84.6	84.3	88.7	87.2	86.2
ShowUI-ZS	✗	56.2	58.9	60.6	54.8	69.2	57.4	58.2

Table 6: *GUI Video understanding on GUI-World multiple-choices question-answering.* We found the following: (i) The **MCQ task is relatively easy**, with pretraining on screenshots achieving 58.2% acc. (ii) The benefits of pretraining are minimal after fine-tuning. (iii) The model can **transfer its knowledge from web-based training to novel software environments**, such as iOS and XR.

PT by?	**Grounding**	**Navigation**		**Understanding**
	Screenspot	Mind2Web	AITW	GUI-World
None – Phi-3.5-v	2.5	34.0	67.0	**87.3**
Grounding	64.9	37.7	68.1	86.2
Navigation	2.7	36.4	69.7	86.6
Understanding	0	35.1	68.8	86.2
All – ShowUI	**66.0**	**40.3**	**69.8**	86.2

Table 7: Impact of different pretraining sources on various downstream tasks. *Pretraining with all capabilities yields the best results in grounding and navigation tasks*, but no significant trend is observed for GUI video understanding.

5 Qualitative Analysis

In Fig.5, we demonstrate how ShowUI is applied to solve a real-world task—opening an Overleaf template for the NeurIPS 2024 submission. The model successfully identifies key actions and grounding elements, reaching the final state successfully in six steps.

6 Conclusions

We developed ShowUI, a Vision-Language-Action (VLA) model for GUI automation across platforms such as websites, desktops, and mobile devices. We provide a clear pathway for building VLA models on top of existing VLMs. By leveraging scalable GUI visual screenshots and action trajectories, ShowUI demonstrates strong performance in grounding, navigation, and understanding tasks, achieving significant results across six benchmarks. Our extensive experiments offer key insights into VLA development for GUI agents. Our work will be fully open-source to support further advancements in GUI automation.

References

[1] OpenAI. Gpt-4 technical report, 2023.

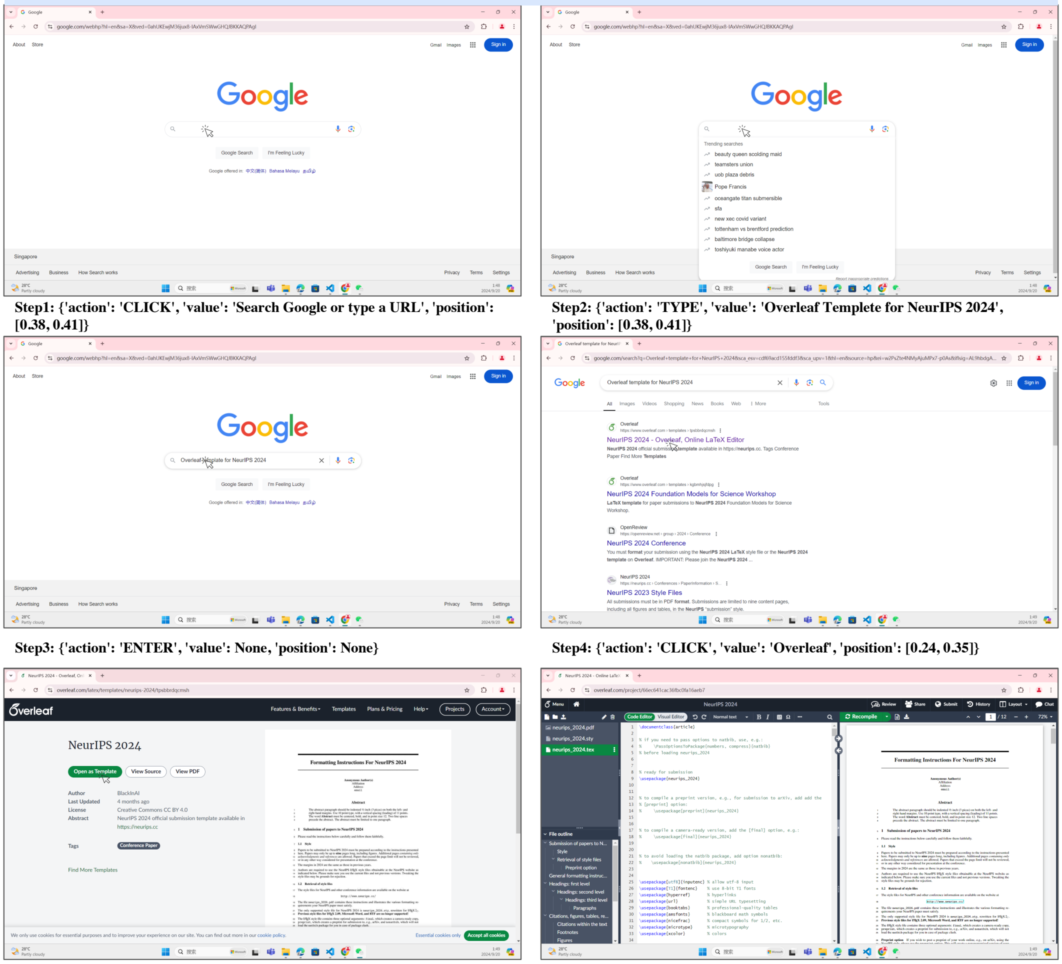
[2] Microsoft copilot. https://copilot.microsoft.com/. Accessed: 2024-04-15.

[3] Izzeddin Gur, Hiroki Furuta, Austin Huang, Mustafa Safdari, Yutaka Matsuo, Douglas Eck, and Aleksandra Faust. A real-world webagent with planning, long context understanding, and program synthesis. *arXiv preprint arXiv:2307.12856*, 2023.

[4] Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Yonatan Bisk, Daniel Fried, Uri Alon, et al. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*, 2023.

[5] Hiroki Furuta, Ofir Nachum, Kuang-Huei Lee, Yutaka Matsuo, Shixiang Shane Gu, and Izzeddin Gur. Multimodal web navigation with instruction-finetuned foundation models. *arXiv preprint arXiv:2305.11854*, 2023.

Task: Open an overleaf template for the NeurIPS 2024 submission.



Step1: {'action': 'CLICK', 'value': 'Search Google or type a URL', 'position': [0.38, 0.41]}

Step2: {'action': 'TYPE', 'value': 'Overleaf Templete for NeurIPS 2024', 'position': [0.38, 0.41]}

Step3: {'action': 'ENTER', 'value': None, 'position': None}

Step4: {'action': 'CLICK', 'value': 'Overleaf', 'position': [0.24, 0.35]}

Step5: {'action': 'CLICK', 'value': 'Open as Template', 'position': [0.18, 0.35]}

Step6: {'action': 'STATUS TASK COMPLETE', 'value': None, 'position': None}

Figure 5: A demonstration how ShowUI response and perform step-by-step action based on user query *Open an overleaf template for the NeurIPS 2024 Submission.*

[6] 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, 2024.

[7] Tianlin Shi, Andrej Karpathy, Linxi Fan, Jonathan Hernandez, and Percy Liang. World of bits: An open-domain platform for web-based agents. In *International Conference on Machine Learning*, pages 3135–3144. PMLR, 2017.

[8] Jiwen Zhang, Jihao Wu, Yihua Teng, Minghui Liao, Nuo Xu, Xiao Xiao, Zhongyu Wei, and Duyu Tang. Android in the zoo: Chain-of-action-thought for gui agents. *arXiv preprint arXiv:2403.02713*, 2024.

[9] Xinbei Ma, Zhuosheng Zhang, and Hai Zhao. Comprehensive cognitive llm agent for smartphone gui automation. *arXiv preprint arXiv:2402.11941*, 2024.

[10] Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy Lillicrap. Android in the wild: A large-scale dataset for android device control. *arXiv preprint arXiv:2307.10088*, 2023.

[11] Hao Wen, Yuanchun Li, Guohong Liu, Shanhui Zhao, Tao Yu, Toby Jia-Jun Li, Shiqi Jiang, Yunhao Liu, Yaqin Zhang, and Yunxin Liu. Empowering llm to use smartphone for intelligent task automation. *arXiv preprint arXiv:2308.15272*, 2023.

[12] Difei Gao, Lei Ji, Zechen Bai, Mingyu Ouyang, Peiran Li, Dongxing Mao, Qinchen Wu, Weichen Zhang, Peiyi Wang, Xiangwu Guo, et al. Assistgui: Task-oriented desktop graphical user interface automation. *arXiv preprint arXiv:2312.13108*, 2023.

[13] Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *arXiv preprint arXiv:2404.07972*, 2024.

[14] Christopher Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyo Campbell-Ajala, et al. Androidworld: A dynamic benchmarking environment for autonomous agents. *arXiv preprint arXiv:2405.14573*, 2024.

[15] Jianwei Yang, Hao Zhang, Feng Li, Xueyan Zou, Chunyuan Li, and Jianfeng Gao. Set-of-mark prompting unleashes extraordinary visual grounding in gpt-4v, 2023.

[16] 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. *arXiv preprint arXiv:2312.08914*, 2023.

[17] Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Yantao Li, Jianbing Zhang, and Zhiyong Wu. Seeclick: Harnessing gui grounding for advanced visual gui agents. *arXiv preprint arXiv:2401.10935*, 2024.

[18] Wentong Chen, Junbo Cui, Jinyi Hu, Yujia Qin, Junjie Fang, Yue Zhao, Chongyi Wang, Jun Liu, Guirong Chen, Yupeng Huo, et al. Guicourse: From general vision language models to versatile gui agents. *arXiv preprint arXiv:2406.11317*, 2024.

[19] Quanfeng Lu, Wenqi Shao, Zitao Liu, Fanqing Meng, Boxuan Li, Botong Chen, Siyuan Huang, Kaipeng Zhang, Yu Qiao, and Ping Luo. Gui odyssey: A comprehensive dataset for cross-app gui navigation on mobile devices. *arXiv preprint arXiv:2406.08451*, 2024.

[20] Qinchen Wu, Difei Gao, Kevin Qinghong Lin, Zhuoyu Wu, Xiangwu Guo, Peiran Li, Weichen Zhang, Hengxu Wang, and Mike Zheng Shou. Gui action narrator: Where and when did that action take place? *arXiv preprint arXiv:2406.13719*, 2024.

[21] Dongping Chen, Yue Huang, Siyuan Wu, Jingyu Tang, Liuyi Chen, Yilin Bai, Zhigang He, Chenlong Wang, Huichi Zhou, Yiqiang Li, et al. Gui-world: A dataset for gui-oriented multimodal llm-based agents. *arXiv preprint arXiv:2406.10819*, 2024.

[22] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

[23] Siyuan Huang, Haonan Chang, Yuhan Liu, Yimeng Zhu, Hao Dong, Peng Gao, Abdeslam Boularias, and Hongsheng Li. A3vlm: Actionable articulation-aware vision language model. *arXiv preprint arXiv:2406.07549*, 2024.

[24] Haoyu Zhen, Xiaowen Qiu, Peihao Chen, Jincheng Yang, Xin Yan, Yilun Du, Yining Hong, and Chuang Gan. 3d-vla: A 3d vision-language-action generative world model. *arXiv preprint arXiv:2403.09631*, 2024.

[25] Hidehisa Arai, Keita Miwa, Kento Sasaki, Yu Yamaguchi, Kohei Watanabe, Shunsuke Aoki, and Issei Yamamoto. Covla: Comprehensive vision-language-action dataset for autonomous driving. *arXiv preprint arXiv:2408.10845*, 2024.

[26] Shalev Lifshitz, Keiran Paster, Harris Chan, Jimmy Ba, and Sheila McIlraith. Steve-1: A generative model for text-to-behavior in minecraft. *Advances in Neural Information Processing Systems*, 36, 2024.

[27] Zihao Wang, Shaofei Cai, Zhancun Mu, Haowei Lin, Ceyao Zhang, Xuejie Liu, Qing Li, Anji Liu, Xiaojian Ma, and Yitao Liang. Omnijarvis: Unified vision-language-action tokenization enables open-world instruction following agents, 2024.

[28] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model, 2024.

[29] Shukang Yin, Chaoyou Fu, Sirui Zhao, Ke Li, Xing Sun, Tong Xu, and Enhong Chen. A survey on multimodal large language models. *arXiv preprint arXiv:2306.13549*, 2023.

[30] Zechen Bai, Pichao Wang, Tianjun Xiao, Tong He, Zongbo Han, Zheng Zhang, and Mike Zheng Shou. Hallucination of multimodal large language models: A survey. *arXiv preprint arXiv:2404.18930*, 2024.

[31] OpenAI. Gpt-4v(ision) system card, 2023.

[32] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pages 19730–19742. PMLR, 2023.

[33] Li Wang, Zechen Bai, Yonghua Zhang, and Hongtao Lu. Show, recall, and tell: Image captioning with recall mechanism. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 12176–12183, 2020.

[34] Drew A Hudson and Christopher D Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6700–6709, 2019.

[35] Chuwei Luo, Yufan Shen, Zhaoqing Zhu, Qi Zheng, Zhi Yu, and Cong Yao. Layoutllm: Layout instruction tuning with large language models for document understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15630–15640, 2024.

[36] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36, 2024.

[37] Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. Minigpt-4: Enhancing vision-language understanding with advanced large language models. *arXiv preprint arXiv:2304.10592*, 2023.

[38] Marah Abdin, Sam Ade Jacobs, Ammar Ahmad Awan, Jyoti Aneja, Ahmed Awadallah, Hany Awadalla, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Harkirat Behl, et al. Phi-3 technical report: A highly capable language model locally on your phone. *arXiv preprint arXiv:2404.14219*, 2024.

[39] Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. Qwen-vl: A frontier large vision-language model with versatile abilities. *arXiv preprint arXiv:2308.12966*, 2023.

[40] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

[41] Abhishek Padalkar, Acorn Pooley, Ajinkya Jain, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anikait Singh, Anthony Brohan, et al. Open x-embodiment: Robotic learning datasets and rt-x models. *arXiv preprint arXiv:2310.08864*, 2023.

[42] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

[43] Keen You, Haotian Zhang, Eldon Schoop, Floris Weers, Amanda Swearngin, Jeffrey Nichols, Yinfei Yang, and Zhe Gan. Ferret-ui: Grounded mobile ui understanding with multimodal llms. *arXiv preprint arXiv:2404.05719*, 2024.

[44] Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757, 2022.

[45] Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Ruslan Salakhutdinov, and Daniel Fried. Visualwebarena: Evaluating multimodal agents on realistic visual web tasks. *arXiv preprint arXiv:2401.13649*, 2024.

[46] Daniel Toyama, Philippe Hamel, Anita Gergely, Gheorghe Comanici, Amelia Glaese, Zafarali Ahmed, Tyler Jackson, Shibl Mourad, and Doina Precup. Androidenv: A reinforcement learning platform for android. *arXiv preprint arXiv:2105.13231*, 2021.

[47] Zhiyong Wu, Chengcheng Han, Zichen Ding, Zhenmin Weng, Zhoumianze Liu, Shunyu Yao, Tao Yu, and Lingpeng Kong. Os-copilot: Towards generalist computer agents with self-improvement. *arXiv preprint arXiv:2402.07456*, 2024.

[48] Chaoyun Zhang, Liqun Li, Shilin He, Xu Zhang, Bo Qiao, Si Qin, Minghua Ma, Yu Kang, Qingwei Lin, Saravan Rajmohan, et al. Ufo: A ui-focused agent for windows os interaction. *arXiv preprint arXiv:2402.07939*, 2024.

[49] Zhuosheng Zhang and Aston Zhang. You only look at screens: Multimodal chain-of-action agents. *arXiv preprint arXiv:2309.11436*, 2023.

[50] Keen You, Haotian Zhang, Eldon Schoop, Floris Weers, Amanda Swearngin, Jeffrey Nichols, Yinfei Yang, and Zhe Gan. Ferret-ui: Grounded mobile ui understanding with multimodal llms. *arXiv preprint arXiv:2404.05719*, 2024.

[51] Yang Li, Jiacong He, Xin Zhou, Yuan Zhang, and Jason Baldridge. Mapping natural language instructions to mobile ui action sequences. *arXiv preprint arXiv:2005.03776*, 2020.

[52] Yang Li, Gang Li, Luheng He, Jingjie Zheng, Hong Li, and Zhiwei Guan. Widget captioning: Generating natural language description for mobile user interface elements. *arXiv preprint arXiv:2010.04295*, 2020.

[53] An Yan, Zhengyuan Yang, Wanrong Zhu, Kevin Lin, Linjie Li, Jianfeng Wang, Jianwei Yang, Yiwu Zhong, Julian McAuley, Jianfeng Gao, et al. Gpt-4v in wonderland: Large multimodal models for zero-shot smartphone gui navigation. *arXiv preprint arXiv:2311.07562*, 2023.

[54] Bryan Wang, Gang Li, Xin Zhou, Zhourong Chen, Tovi Grossman, and Yang Li. Screen2words: Automatic mobile ui summarization with multimodal learning. In *The 34th Annual ACM Symposium on User Interface Software and Technology*, pages 498–510, 2021.

[55] Yu-Chung Hsiao, Fedir Zubach, Maria Wang, and Jindong Chen. Screenqa: Large-scale question-answer pairs over mobile app screenshots, 2024.

[56] PyAutoGUI. Pyautogui. 2024. https://pyautogui.readthedocs.io/en/latest/.

[57] Jianguo Zhang, Tian Lan, Ming Zhu, Zuxin Liu, Thai Hoang, Shirley Kokane, Weiran Yao, Juntao Tan, Akshara Prabhakar, Haolin Chen, et al. xlam: A family of large action models to empower ai agent systems. *arXiv preprint arXiv:2409.03215*, 2024.

[58] Yuxiang Chai, Siyuan Huang, Yazhe Niu, Han Xiao, Liang Liu, Dingyu Zhang, Peng Gao, Shuai Ren, and Hongsheng Li. Amex: Android multi-annotation expo dataset for mobile gui agents. *arXiv preprint arXiv:2407.17490*, 2024.

[59] Evan Zheran Liu, Kelvin Guu, Panupong Pasupat, Tianlin Shi, and Percy Liang. Reinforcement learning on web interfaces using workflow-guided exploration, 2018.

[60] Jiwen Zhang, Jihao Wu, Yihua Teng, Minghui Liao, Nuo Xu, Xiao Xiao, Zhongyu Wei, and Duyu Tang. Android in the zoo: Chain-of-action-thought for gui agents. *arXiv preprint arXiv:2403.02713*, 2024.

[61] Rohan Bavishi, Erich Elsen, Curtis Hawthorne, Maxwell Nye, Augustus Odena, Arushi Somani, and Sağnak Taşırlar. Introducing our multimodal models, 2023.

[62] Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution, 2024.

[63] OpenAI. Hello gpt-4o, May 2024. Accessed: 2024-06-06.

[64] Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

[65] Peng Jin, Ryuichi Takanobu, Wancai Zhang, Xiaochun Cao, and Li Yuan. Chat-univi: Unified visual representation empowers large language models with image and video understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13700–13710, 2024.

[66] Kirolos Ataallah, Xiaoqian Shen, Eslam Abdelrahman, Essam Sleiman, Deyao Zhu, Jian Ding, and Mohamed Elhoseiny. Minigpt4-video: Advancing multimodal llms for video understanding with interleaved visual-textual tokens. *arXiv preprint arXiv:2404.03413*, 2024.

[67] Kunchang Li, Yali Wang, Yinan He, Yizhuo Li, Yi Wang, Yi Liu, Zun Wang, Jilan Xu, Guo Chen, Ping Luo, et al. Mvbench: A comprehensive multi-modal video understanding benchmark. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22195–22206, 2024.