LearnAct: Few-Shot Mobile GUI Agent with a Unified Demonstration Benchmark

LearnAct: 具有统一示范基准的少样本移动 GUI 代理

Guangyi Liu†

刘光义 †

Zhejiang University

浙江大学

Hangzhou, China

中国杭州

Pengxiang Zhao†

赵鹏翔 †

Zhejiang University

浙江大学

Hangzhou, China

中国杭州

Liang Liu*

刘亮 *

vivo AI Lab

vivo 人工智能实验室

Hangzhou, China

中国杭州

Zhiming Chen

陈志明

vivo AI Lab

vivo 人工智能实验室

Hangzhou, China

中国杭州

Yuxiang Chai

柴宇翔

vivo AI Lab

vivo 人工智能实验室

Hangzhou, China

中国杭州

Shuai Ren

任帅

vivo AI Lab

vivo 人工智能实验室

ShenZhen, China

中国深圳

Hao Wang

王昊

vivo AI Lab

vivo 人工智能实验室

ShenZhen, China

中国深圳

Shibo He

何世博

Zhejiang University

浙江大学

Hangzhou, China

中国杭州

Wenchao Meng

孟文超

Zhejiang University

浙江大学

Hangzhou, China

中国杭州

wmengzju@zju.edu.cn

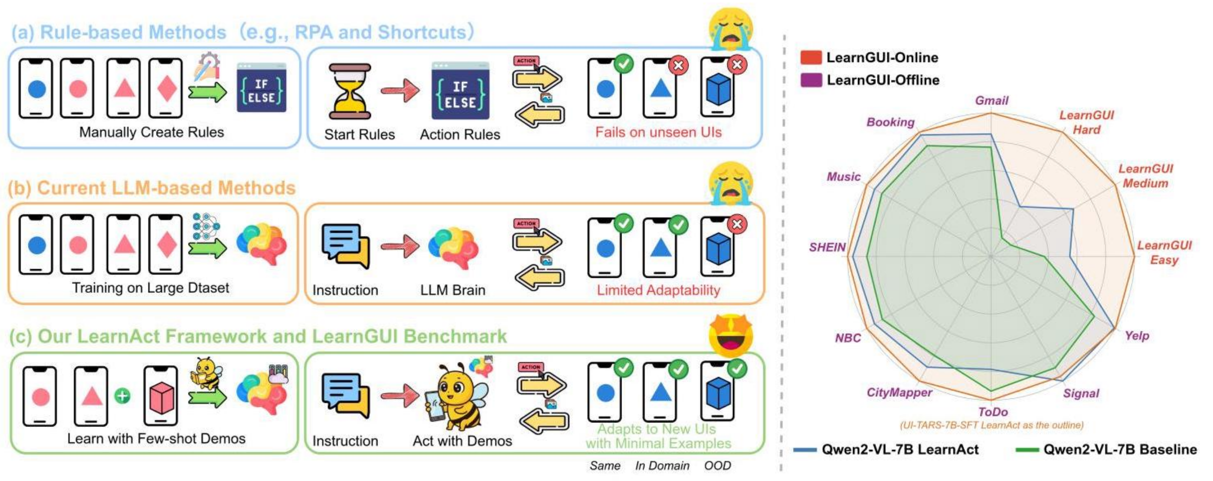


Figure 1: The LearnAct Framework and LearnGUI Benchmark focus on addressing the long-tail challenges in mobile GUI agent performance through demonstration-based learning. From rule-based automation to LLM-powered agents, mobile GUI automation has evolved significantly, yet still struggles with long-tail scenarios due to interface diversity. Our LearnAct framework introduces demonstration based learning to effectively handle these challenges, outperforming existing methods in both offline and online evaluations.

图 1:LearnAct 框架与 LearnGUI 基准通过基于示范的学习，聚焦解决移动 GUI 代理性能中的长尾挑战。从基于规则的自动化到大型语言模型 (LLM) 驱动的代理，移动 GUI 自动化已显著发展，但由于界面多样性，仍难以应对长尾场景。我们的 LearnAct 框架引入基于示范的学习，有效处理这些挑战，在离线和在线评测中均优于现有方法。

ABSTRACT

摘要

Mobile GUI agents show promise in automating tasks but face generalization challenges in diverse real-world scenarios. Traditional approaches using pre-training or fine-tuning with massive datasets struggle with the diversity of mobile applications and user-specific tasks. We propose enhancing mobile GUI agent capabilities through human demonstrations, focusing on improving performance in unseen scenarios rather than pursuing universal generalization through larger datasets. To realize this paradigm, we introduce LearnGUI, the first comprehensive dataset specifically designed for studying demonstration-based learning in mobile GUI agents. It comprises 2,252 offline tasks and 101 online tasks with high-quality

移动 GUI 代理在自动化任务中展现出潜力，但在多样化的真实场景中面临泛化挑战。传统方法通过大规模数据集的预训练或微调，难以应对移动应用和用户特定任务的多样性。我们提出通过人类示范提升移动 GUI 代理能力，重点改善未见场景的表现，而非通过更大数据集追求普适泛化。为实现该范式，介绍了 LearnGUI，这是首个专门为研究基于示范学习的移动 GUI 代理设计的综合数据集，包含 2252 个离线任务和 101 个高质量在线任务。

† Equal Contribution, ‡ Project Lead, ☒ Corresponding Author.

† 同等贡献，‡ 项目负责人，☒ 通讯作者。

human demonstrations. We further develop LearnAct, a sophisticated multi-agent framework that automatically extracts knowledge from demonstrations to enhance task completion. This framework integrates three specialized agents: DemoParser for knowledge extraction, KnowSeeker for relevant knowledge retrieval, and ActExecutor for demonstration-enhanced task execution. Our experimental results show significant performance gains in both offline and online evaluations. In offline assessments, a single demonstration improves model performance, increasing Gemini-1.5-Pro's accuracy from 19.3% to 51.7%. In online evaluations, our framework enhances UI-TARS-7B-SFT's task success rate from 18.1% to 32.8%. Learn-Act framework and LearnGUI benchmark establish demonstration-based learning as a promising direction for more adaptable, personalized, and deployable mobile GUI agents. The project resources are available at https://lgy0404.github.io/LearnAct.

人类演示。我们进一步开发了 LearnAct, 这是一个复杂的多智能体框架, 能够自动从演示中提取知识以提升任务完成效果。该框架整合了三个专门的智能体: 用于知识提取的 DemoParser, 用于相关知识检索的 KnowSeeker, 以及用于演示增强任务执行的 ActExecutor。我们的实验结果显示, 在离线和在线评估中均取得了显著的性能提升。在离线评估中，单次演示即可提升模型性能，使 Gemini-1.5-Pro 的准确率从 19.3% 提升至 51.7%。在在线评估中，我们的框架将 UI-TARS-7B-SFT 的任务成功率从 18.1% 提升至 32.8%。LearnAct 框架和 LearnGUI 基准确立了基于演示学习的方向，为更具适应性、个性化和可部署的移动 GUI 智能体开辟了前景。项目资源可访问 https://lgy0404.github.io/LearnAct。

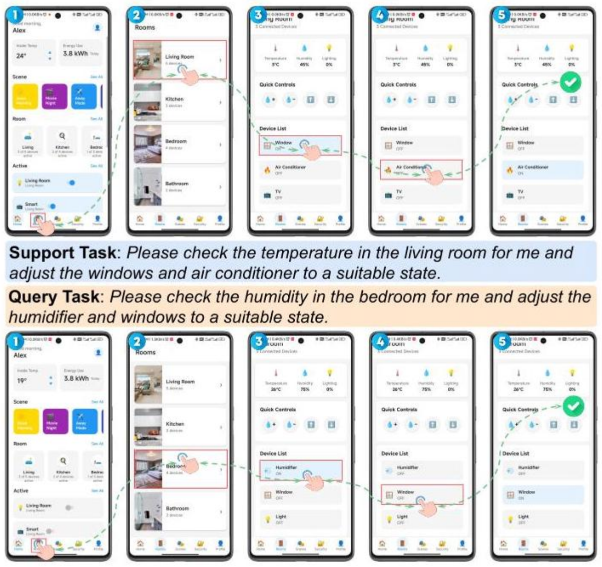

Support Task: *Please check the temperature in the living room for me and adjust the windows and air conditioner to a suitable state.*
Query Task: *Please check the humidity in the bedroom for me and adjust the humidifier and windows to a suitable state.*

Figure 2: A toy example for demonstration learning on mobile GUI Agent. We build a benchmark named LearnGUI for demonstration learning on Mobile GUI Agent, which provides different few-shot task combinations and offers multi-dimensional metrics including task similarity, UI similarity, and action similarity between support tasks and query tasks.

图 2: 移动 GUI 智能体演示学习的示例。我们构建了名为 LearnGUI 的基准，用于移动 GUI 智能体的演示学习，提供不同的少样本任务组合，并提供多维度指标，包括支持任务与查询任务之间的任务相似度、界面相似度和动作相似度。

1 INTRODUCTION

1 引言

Mobile device automation has evolved significantly over time, from simple rule-based scripts to sophisticated AI-powered agents [17, 32, 38, 43]. Traditional automation approaches like Robotic Process Automation (RPA) [1] and rule-based shortcuts [10, 13] relied on predefined scripts to execute repetitive tasks, but they struggled with dynamic interfaces, required frequent maintenance when apps updated, and lacked understanding of complex user intentions.

移动设备自动化经历了显著发展，从简单的基于规则的脚本演变为复杂的 AI 驱动智能体 [17, 32, 38, 43]。传统自动化方法如机器人流程自动化 (RPA)[1] 和基于规则的快捷方式 [10, 13] 依赖预定义脚本执行重复任务，但在动态界面下表现不佳，应用更新时需频繁维护，且难以理解复杂的用户意图。

More recently, mobile Graphical User Interface (GUI) agents have emerged as a transformative technology with the potential to revolutionize how humans interact with mobile devices. These agents leverage Large Language Models (LLMs) to autonomously complete human tasks through environmental interaction [6, 18, 20, 28, 29, 33, 36, 37, 46]. They perceive phone states of mobile phone by observing screens (through screenshots or application UI trees) and generate actions (such as CLICK, TYPE, SWIPE, PRESS_BACK, PRESS_HOME, and

PRESS_ENTER) that are executed via the phone user interface [17, 32, 38, 43] . By harnessing the powerful perception and reasoning capabilities of LLMs, mobile GUI agents have the potential to fundamentally change how people interact with their mobile devices, bringing to life the ”J.A.R.V.I.S.” effect seen in science fiction.

近年来，移动图形用户界面 (GUI) 智能体作为一项变革性技术出现，有望彻底改变人们与移动设备的交互方式。这些智能体利用大型语言模型 (LLMs) 通过环境交互自主完成用户任务 [6, 18, 20, 28, 29, 33, 36, 37, 46]。它们通过观察屏幕 (截图或应用 UI 树) 感知手机状态，并生成通过手机用户界面执行的动作 (如点击、输入、滑动、返回键、主页键和回车键) [17, 32, 38, 43] 。借助 LLMs 强大的感知和推理能力，移动 GUI 智能体有潜力从根本上改变人们与移动设备的交互，带来科幻作品中“贾维斯”(J.A.R.V.I.S.) 般的体验。

Despite these promising advances, mobile GUI agents continue to face significant challenges in real-world deployment scenarios. The immense diversity of mobile applications and user interfaces creates pervasive long-tail scenarios where current agents struggle

尽管取得了这些令人鼓舞的进展，移动 GUI 智能体在实际部署场景中仍面临重大挑战。移动应用和用户界面的极大多样性导致普遍存在长尾场景，当前智能体难以有效应对。

to perform effectively. The prevailing approaches to building modern mobile GUI agents rely on either the inherent capabilities of general-purpose LLMs [18, 20, 28, 29, 34, 36, 37, 46] or fine-tuning with large volumes of data [11, 16, 41, 48]. However, these methods face fundamental limitations when confronted with diverse real-world usage scenarios. As of 2025, billions of users interact with 1.68 million applications on Google Play alone [17], each with unique task requirements and UI layouts [32, 43]. Pre-training or fine-tuning datasets cannot feasibly cover this immense variety, leading to poor performance in unseen scenarios and hindering the widespread adoption of mobile GUI agents [14], as illustrated in Figure 1 (left side). Traditional approaches simply cannot cover the entire spectrum of possible interactions and user-specific requirements across this heterogeneous landscape.

构建现代移动 GUI 智能体的主流方法依赖于通用大型语言模型的固有能力 [18, 20, 28, 29, 34, 36, 37, 46] 或大量数据的微调 [11, 16, 41, 48]。然而，这些方法在面对多样化的真实使用场景时存在根本性限制。截至 2025 年，仅 Google Play 上就有 16.8 万个应用，数十亿用户使用 [17]，每个应用具有独特的任务需求和界面布局 [32, 43]。预训练或微调数据集无法覆盖如此庞大的多样性，导致在未见场景中表现不佳，阻碍了移动 GUI 智能体的广泛应用 [14]，如图 1(左侧) 所示。传统方法无法涵盖这一异构环境中所有可能的交互和用户特定需求。

To address these limitations, we propose a novel paradigm that enhances mobile GUI agent capabilities through few-shot demonstration learning. Unlike traditional approaches that either lack flexibility or require massive datasets, our demonstration-based approach achieves both robustness and personalization by learning from a small number of user-provided examples. We recognize that mobile users have unique, repetitive tasks with inherent variability-such as smart home control with dynamic configurations, health monitoring with personalized parameters, or enterprise software with company-specific layouts. These scenarios combine stable patterns with variable elements, creating a ”personalization gap” that pre-trained models cannot bridge. By leveraging user-specific demonstrations, our approach enables personalized assistants that learn both consistent patterns and adaptation strategies, acquiring task-specific knowledge impossible to cover in general training datasets. This personalization allows mobile GUI agents to overcome performance bottlenecks and provide truly helpful automation for the tasks users most want to delegate.

为解决这些限制，我们提出了一种通过少样本演示学习增强移动 GUI 智能体能力的新范式。不同于缺乏灵活性或需大量数据的传统方法，我们的基于演示的方法通过学习少量用户示例，实现了鲁棒性和个性化。我们认识到移动用户拥有独特且重复的任务，且存在内在变异性——如智能家居控制的动态配置、个性化参数的健康监测，或企业软件的公司特定布局。这些场景结合了稳定模式与可变元素，形成了预训练模型无法弥合的“个性化鸿沟”。通过利用用户特定的演示，我们的方法使个性化助手能够学习一致模式和适应策略，掌握通用训练数据集中无法涵盖的任务特定知识。这种个性化使移动 GUI 智能体突破性能瓶颈，为用户最希望委托的任务提供真正有用的自动化支持。

To fill the gap in high-quality demonstration data, we introduce LearnGUI, the first dataset specifically designed to research and evaluate mobile GUI agents' ability to learn from few-shot demonstrations. Built upon AMEX [5] and Android-World [23], LearnGUI comprises 2,252 offline few-shot tasks and 101 online tasks with high-quality human demonstrations. This dataset enables systematic research into demonstration-based learning for mobile GUI agents. A toy example for LearnGUI is shown in Figure 2.

为填补高质量示范数据的空白，我们引入了 LearnGUI，这是首个专门设计用于研究和评估移动 GUI 代理从少量示范中学习能力的数据集。LearnGUI 基于 AMEX [5] 和 Android-World [23] 构建，包含 2252 个离线少样本任务和 101 个带有高质量人工示范的在线任务。该数据集支持对基于示范学习的移动 GUI 代理进行系统性研究。LearnGUI 的示例见图 2。

Furthermore, we present LearnAct, a multi-agent framework that automatically understands human demonstrations, generates instructional knowledge, and uses this knowledge to assist mobile GUI agents in reasoning about unseen scenarios. LearnAct consists of three specialized agents: (1) DemoParser, a knowledge generation agent that extracts usable knowledge from demonstration trajectories to form a knowledge base; (2) KnowSeeker, a knowledge retrieval agent that searches the knowledge base for demonstration knowledge relevant to the current task; and (3) ActExecutor, a task execution agent that combines user instructions, real-time GUI environment, and retrieved demonstration knowledge to perform tasks effectively.

此外，我们提出了 LearnAct，一个多代理框架，能够自动理解人类示范，生成指导性知识，并利用这些知识辅助移动 GUI 代理推理未知场景。LearnAct 由三个专门代理组成:(1) DemoParser，知识生成代理，从示范轨迹中提取可用知识以构建知识库；(2) KnowSeeker，知识检索代理，在知识库中搜索与当前任务相关的示范知识；(3) ActExecutor，任务执行代理，结合用户指令、实时 GUI 环境和检索到的示范知识，有效执行任务。

Our experimental results decisively validate the effectiveness of demonstration-based learning for mobile GUI agents, as shown in Figure 1 (right side). In offline evaluations, a single demonstration dramatically improves model performance across diverse scenarios,

我们的实验结果明确验证了基于示范学习对移动 GUI 代理的有效性，如图 1(右侧) 所示。在离线评估中，单一示范显著提升了模型在多样场景下的表现，

Table 1: Comparison of different datasets and environments for benchmarking Mobile GUI agents. Column definitions: # Inst. (number of instructions), # Apps (number of applications), #Step (average steps per task), Env. (supports environment interactions), HL (has high-level instructions), LL (has low-level instructions), GT (provides ground truth trajectories), FS (supports few-shot learning).

表 1: 用于移动 GUI 代理基准测试的不同数据集和环境比较。列定义:# Inst.(指令数量)，# Apps(应用数量)，#Step(每个任务的平均步骤数)，Env.(支持环境交互)，HL(包含高级指令)，LL(包含低级指令)，GT(提供真实轨迹)，FS(支持少样本学习)。

Dataset	#Inst.	#Apps	#Step	Env.	HL	LL	**GT**	FS
PixelHelp [15]	187	4	4.2	✘	✓	✘	✓	✘
MoTIF [4]	276	125	4.5	✘	✓	✓	✓	✘
UIBert [3]	16,660	-	1	✘	✘	✓	✓	✘
UGIF [27]	523	12	6.3	✘	✓	✓	✓	✘
AITW [24]	30,378	357	6.5	✘	✓	✘	✓	✘
AITZ [45]	2,504	70	7.5	✘	✓	✓	✓	✘
AndroidControl [14]	15,283	833	4.8	✘	✓	✓	✓	✘
AMEX [5]	2,946	110	12.8	✘	✓	✘	✓	✘
MobileAgentBench [30]	100	10	-	✘	✓	✘	✘	✘
AppAgent [44]	50	10	-	✘	✓	✘	✘	✘
LlamaTouch [47]	496	57	7.01	✓	✓	✘	✓	✘
AndroidWorld [23]	116	20	-	✓	✓	✘	✘	✘
AndroidLab [40]	138	9	8.5	✓	✓	✘	✘	✘
LearnGUI (Ours)	2,353	73	13.2	✓	✓	✓	✓	✓

数据集	实例数	应用数	步骤数	环境	高层次	低层次	**GT**	功能选择
PixelHelp [15]	187	4	4.2	✘	✓	✘	✓	✘
MoTIF [4]	276	125	4.5	✘	✓	✓	✓	✘
UIBert [3]	16,660	-	1	✘	✘	✓	✓	✘
UGIF [27]	523	12	6.3	✘	✓	✓	✓	✘
AITW [24]	30,378	357	6.5	✘	✓	✘	✓	✘
AITZ [45]	2,504	70	7.5	✘	✓	✓	✓	✘
AndroidControl [14]	15,283	833	4.8	✘	✓	✓	✓	✘
AMEX [5]	2,946	110	12.8	✘	✓	✘	✓	✘
MobileAgentBench [30]	100	10	-	✘	✓	✘	✘	✘
AppAgent [44]	50	10	-	✘	✓	✘	✘	✘
LlamaTouch [47]	496	57	7.01	✓	✓	✘	✓	✘
AndroidWorld [23]	116	20	-	✓	✓	✘	✘	✘
AndroidLab [40]	138	9	8.5	✓	✓	✘	✘	✘
LearnGUI(本研究)	2,353	73	13.2	✓	✓	✓	✓	✓

with the most striking results seen in Gemini-1.5-Pro [26], whose accuracy increases from 19.3% to 51.7% (a 198.9% relative improvement). Performance gains are particularly pronounced in complex applications, with accuracy in CityMapper increasing from 14.1% to 69.4% and in To-Do apps from 17.4% to 69.2%. For real-world online evaluations, our framework demonstrates exceptional effectiveness, with Qwen2-VL-7B [31] with LearnAct achieving significant performance gains, while UI-TARS-7B-SFT [22]'s task success rate improves from 18.1% to 32.8% (+14.7%). These findings offer a practical pathway to developing more adaptable and personalized mobile GUI agents.

最显著的结果出现在 Gemini-1.5-Pro [26] 中，其准确率从 19.3% 提升至 51.7%(相对提升 198.9%)。性能提升在复杂应用中尤为明显，CityMapper 的准确率从 14.1% 提升至 69.4%，待办事项应用的准确率从 17.4% 提升至 69.2%。在真实在线评估中，我们的框架表现出卓越的效果，Qwen2-VL-7B [31] 结合 LearnAct 实现了显著的性能提升，而 UI-TARS-7B-SFT [22] 的任务成功率从 18.1% 提升至 32.8%(+14.7%)。这些发现为开发更具适应性和个性化的移动 GUI 代理提供了实用路径。

In summary, our contributions are as follows:

总之，我们的贡献如下:

- We develop LearnGUI, the first dataset designed for studying demonstration-based learning in mobile GUI agents, comprising 2,252 offline and 101 online tasks with high-quality human demonstrations.

    - 我们开发了 LearnGUI，这是首个专为研究基于示范学习的移动 GUI 代理设计的数据集，包含 2,252 个离线任务和 101 个在线任务，均配有高质量的人类示范。

- We design and implement LearnAct, a sophisticated multi-agent framework that systematically extracts, retrieves, and leverages knowledge from human demonstrations. This framework includes three specialized components: DemoParser (knowledge extraction), KnowSeeker (knowledge retrieval), and ActExecutor (task execution).

    - 我们设计并实现了 LearnAct,一个复杂的多代理框架,系统地提取、检索并利用人类示范中的知识。该框架包括三个专门组件:DemoParser(知识提取)、KnowSeeker(知识检索) 和 ActExecutor(任务执行)。

- Our evaluations demonstrate unprecedented performance gains: a single demonstration improves Gemini-1.5-Pro [26]'s accuracy by 198.9% in offline tests, while enhancing UI-TARS- 7B-SFT [22]'s online task success rate from 18.1% to 32.8%, advancing mobile GUI agents toward greater adaptability and practical deployability.

    - 我们的评估展示了前所未有的性能提升: 单次示范使 Gemini-1.5-Pro [26] 在离线测试中的准确率提升 198.9%，同时将 UI-TARS-7B-SFT [22] 的在线任务成功率从 18.1% 提升至 32.8%，推动移动 GUI 代理向更高的适应性和实用部署迈进。

2 RELATED WORK

2 相关工作

Mobile GUI Datasets and Environments. The development of mobile GUI agents relies heavily on high-quality datasets for training and evaluation. Table 1 compares LearnGUI and existing mobile

移动 GUI 数据集与环境。移动 GUI 代理的发展高度依赖于高质量的数据集用于训练和评估。表 1 比较了 LearnGUI 与现有的移动

GUI datasets and benchmarks. These resources can be broadly categorized into static datasets and dynamic benchmarking environments. Static datasets [3-5, 14, 15, 24, 27, 30, 45] typically provide natural language task descriptions, UI states (screenshots and/or application UI trees), and corresponding user actions (CLICK, SWIPE, TYPE, and other standardized interactions). These datasets vary in scale, ranging from hundreds to tens of thousands of instructions across different applications. Recent work like AppAgent [44] has explored demonstration-based learning but without ground truth annotations or systematic analysis, providing only 50 tasks across 10 applications with high-level instructions. Notably, the average task length varies significantly across datasets, with AMEX [5] featuring substantially longer sequences (12.8 steps on average) compared to AndroidControl (4.8 steps) and AITW [24] (6.5 steps). Benchmarking environments, on the other hand, typically select a limited number of tasks and applications to provide dynamic testing environments [17]. These frameworks evaluate agent performance through metrics such as task completion rates, critical state achievements, and execution time. Examples include Llama-Touch [47], AndroidWorld [23], and AndroidLab [40], which offer interactive environments but lack few-shot demonstration capabilities. We present the first systematic study of demonstration-based learning for mobile GUI agents through LearnGUI, which distinguishes itself through three key innovations. First, it is designed to evaluate few-shot learning capabilities with a comprehensive collection of 2,252 offline tasks and 101 online tasks. Built upon AMEX [5] and AndroidWorld [23], which feature longer, more complex tasks ideal for out-of-distribution and demonstration-based learning scenarios, LearnGUI provides a unified framework for both offline and online evaluation. Second, while the original AMEX [5] dataset contains 2,946 independent tasks unsuitable for few-shot evaluation, we conducted detailed analyses to transform and enhance this resource. Specifically, we made three key modifications: (1) Action Space Standardization, refining the original action space by removing inconsistent TASK_IMPOSSIBLE actions, enhancing TASK_COMPLETE to support information retrieval tasks, and standardizing formats for consistency; (2) K-shot Task Combinations, constructing systematic task groupings by recovering application context, computing instruction similarity within applications, and creating k-shot combinations with similar tasks as support demonstrations; and (3) Similarity Measurement, computing UI and action similarity through descriptive representations, enabling analysis of how different similarity types affect learning efficacy. Third, regarding online evaluation, AndroidWorld [23] originally provides 116 dynamically constructed tasks without human demonstration trajectories. We collected 101 high-quality human demonstrations based on AndroidWorld's environment and dynamic instructions, forming LearnGUI-Online for evaluating the few-shot capabilities of mobile GUI agents in real-time scenarios. By addressing the limitations of existing datasets, LearnGUI enables systematic research into few-shot learning for mobile GUI agents with varying k-shot configurations and controlled similarity conditions between support and query tasks.

GUI 数据集和基准。这些资源大致可分为静态数据集和动态基准环境。静态数据集 [3-5, 14, 15, 24, 27, 30, 45] 通常提供自然语言任务描述、UI 状态 (截图和/或应用 UI 树) 及相应的用户操作 (点击、滑动、输入及其他标准化交互)。这些数据集规模不一，涵盖数百至数万条不同应用的指令。近期工作如 AppAgent [44] 探索了基于示范的学习，但缺乏真实标注和系统分析，仅提供 10 个应用共 50 个任务的高层次指令。值得注意的是，不同数据集的平均任务长度差异显著，AMEX [5] 的序列较长 (平均 12.8 步)，而 AndroidControl(4.8 步) 和 AITW [24](6.5 步) 较短。基准环境通常选择有限的任务和应用，提供动态测试环境 [17]，通过任务完成率、关键状态达成率和执行时间等指标评估代理性能。典型例子包括 Llama-Touch [47]、AndroidWorld [23] 和 AndroidLab [40]，它们提供交互环境但缺乏少量示范学习能力。我们通过 LearnGUI 首次系统研究了移动 GUI 代理的基于示范学习，具有三大创新。首先，LearnGUI 设计用于评估少量示范学习能力，包含 2,252 个离线任务和 101 个在线任务。基于 AMEX [5] 和 AndroidWorld [23]，这两个数据集任务较长且复杂，适合分布外和基于示范的学习场景，LearnGUI 提供统一的离线和在线评估框架。其次，原 AMEX [5] 包含 2,946 个独立任务，不适合少量示范评估，我们进行了详细分析并改进该资源，主要包括:(1) 动作空间标准化，剔除不一致的 TASK_IMPOSSIBLE 动作，增强 TASK_COMPLETE 以支持信息检索任务，统一格式以保持一致性；(2)K-shot 任务组合，通过恢复应用上下文、计算应用内指令相似度，构建相似任务的 k-shot 组合作为支持示范；(3) 相似度测量，通过描述性表示计算 UI 和动作相似度，分析不同相似度类型对学习效果的影响。第三，关于在线评估，AndroidWorld [23] 原提供 116 个动态构建任务但无人工示范轨迹。我们基于 AndroidWorld 环境和动态指令收集了 101 个高质量人类示范，形成 LearnGUI-Online，用于评估移动 GUI 代理在实时场景下的少量示范能力。通过解决现有数据集的局限，LearnGUI 支持系统研究移动 GUI 代理的少量示范学习，涵盖不同 k-shot 配置及支持与查询任务间的相似度控制。

Mobile GUI Agents. Mobile GUI agents are intelligent systems that leverage large language models to understand, plan, and execute tasks on mobile devices by integrating natural language

移动 GUI 代理。移动 GUI 代理是利用大型语言模型，通过整合自然语言处理、多模态感知和动作执行能力，在移动设备上理解、规划和执行任务的智能系统

processing, multimodal perception, and action execution capabilities [32, 38]. Recent developments in this field have explored various approaches to enhance agent performance and generalizability. One prominent category of work focuses on designing effective prompting strategies to guide pre-trained LLMs without additional training [8, 35, 42]. By crafting prompts that incorporate task descriptions, interface states, and action histories, researchers can direct model behavior toward specific automation goals [25, 28, 29, 37]. These approaches leverage the inherent capabilities of general-purpose LLMs but often struggle with complex tasks. A second category involves adapting LLMs specifically for mobile automation through fine-tuning techniques [7, 9, 11, 16, 19, 21, 41] . These methods train models on GUI-specific data to enhance their understanding of and interaction with graphical interfaces. While improving performance over pre-training approaches, these fine-tuned models require substantial training data and still face generalization challenges. Despite the progress made by both approaches, a fundamental limitation persists: the inability to generalize effectively to out-of-distribution scenarios. These methods both struggle with unseen applications, novel UI layouts, or unexpected task variations. These limitations stem from the impossibility of covering all potential real-world scenarios during training, creating significant bottlenecks in mobile GUI agent development. To address these critical challenges, we introduce LearnAct, a sophisticated multi-agent framework that learns and reasons from screenshots without requiring UI tree information. The framework extracts, retrieves, and utilizes demonstration knowledge through three specialized components, enabling effective adaptation to new scenarios with minimal demonstrations.

处理、多模态感知和动作执行能力 [32, 38]。该领域的最新进展探索了多种方法以提升代理的性能和泛化能力。一类重要的工作聚焦于设计有效的提示策略，以在无需额外训练的情况下引导预训练大型语言模型 (LLM)[8, 35, 42]。通过构建包含任务描述、界面状态和操作历史的提示，研究人员能够引导模型行为朝向特定的自动化目标 [25, 28, 29, 37]。这些方法利用了通用大型语言模型的内在能力，但在处理复杂任务时常常表现不足。第二类方法则通过微调技术专门针对移动自动化调整大型语言模型 $[7, 9, 11, 16, 19, 21, 41]$ 。这些方法在 GUI 特定数据上训练模型，以增强其对图形界面的理解和交互能力。虽然相较于预训练方法性能有所提升，但这些微调模型需要大量训练数据，且仍面临泛化挑战。尽管两种方法均取得了一定进展，根本限制依然存在: 无法有效泛化到分布外场景。这些方法在面对未见过的应用、新颖的 UI 布局或意外的任务变化时均表现不佳。这些限制源于训练过程中无法覆盖所有潜在的现实场景，成为移动 GUI 代理开发中的重大瓶颈。为应对这些关键挑战，我们提出了 LearnAct，一种复杂的多代理框架，能够从截图中学习和推理，无需 UI 树信息。该框架通过三个专门组件提取、检索并利用示范知识，实现以最少示范有效适应新场景。

3 LEARNGUI DATASET

3 LEARNGUI 数据集

3.1 Task Definition

3.1 任务定义

Mobile GUI tasks require agents to interact with digital environments by executing actions to fulfill user instructions. These tasks can be formally described as a Partially Observable Markov Decision Process (POMDP), defined as $\mathcal{M} = (\mathcal{S}, \mathcal{O}, \mathcal{A}, \mathcal{T}, \mathcal{R})$, where $\mathcal{S}$ is the state space (current state of the mobile device), $\mathcal{O}$ is the observation space (instructions, screenshots, UI trees, etc.), $\mathcal{A}$ is the action space (e.g., click, type, swipe), $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the state transition function, and $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to [0, 1]$ is the reward function. For example, a user might request the agent to ”find the cheapest hotel in Paris for next weekend.” The agent must perceive the current screen-either through an image or a UI tree-and execute a sequence of actions to complete the given task.

移动 GUI 任务要求代理通过执行操作与数字环境交互，以完成用户指令。此类任务可形式化描述为部分可观测马尔可夫决策过程 (POMDP)，定义为 $\mathcal{M} = (\mathcal{S}, \mathcal{O}, \mathcal{A}, \mathcal{T}, \mathcal{R})$ ，其中 $\mathcal{S}$ 为状态空间 (移动设备的当前状态)，$\mathcal{O}$ 为观测空间 (指令、截图、UI 树等)，$\mathcal{A}$ 为动作空间 (如点击、输入、滑动)，$\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ 为状态转移函数，$\mathcal{R} : \mathcal{S} \times \mathcal{A} \to [0, 1]$ 为奖励函数。例如，用户可能要求代理“查找下周末巴黎最便宜的酒店”。代理必须通过图像或 UI 树感知当前屏幕，并执行一系列操作以完成指定任务。

The key innovation in our approach is the integration of human demonstration knowledge into this POMDP framework. By incorporating demonstration knowledge $\mathcal{D}$ into the decision process, we enhance the agent’s ability to handle out-of-distribution scenarios. This knowledge influences the agent’s policy $\pi : \mathcal{O} \times \mathcal{D} \to \mathcal{A}$, which maps observations and relevant demonstration knowledge to actions, providing valuable examples of successful interaction patterns.

我们方法的关键创新在于将人类示范知识整合进该 POMDP 框架。通过将示范知识 $\mathcal{D}$ 纳入决策过程，增强了代理处理分布外场景的能力。该知识影响代理的策略 $\pi : \mathcal{O} \times \mathcal{D} \to \mathcal{A}$，该策略将观测和相关示范知识映射到动作，提供了成功交互模式的宝贵示例。

To study the impact of demonstration-based learning on mobile GUI agents, we need a dataset that provides various k-shot demonstrations with controlled similarity relationships between support and query tasks. This allows us to systematically investigate

为了研究基于示范学习对移动 GUI 代理的影响，我们需要一个提供多种 k-shot 示范且支持与查询任务之间相似性关系受控的数据集。这使我们能够系统地探讨

how demonstration quantity and task similarity affect agent performance. While cross-application knowledge transfer remains an interesting research direction, we focus on within-application task learning, as this represents the most practical use case where users would provide demonstrations for applications they frequently use.

示范数量和任务相似性如何影响代理性能。尽管跨应用知识迁移是一个有趣的研究方向，我们重点关注应用内任务学习，因为这是用户为常用应用提供示范的最实际场景。

Our dataset design specifically enables research on three key dimensions:

我们的数据集设计特别支持对三个关键维度的研究:

(1) Unified comprehensive evaluation framework: Learn-GUI provides a standardized platform for studying few-shot demonstration learning in mobile GUI agents, featuring a unified action space and evaluation protocols that reflect real-world use cases

(1) 统一的综合评估框架:Learn-GUI 提供了一个标准化平台，用于研究移动 GUI 代理中的少样本示范学习，具备统一的动作空间和反映现实使用场景的评估协议

(2) K-shot demonstration learning: The dataset systematically explores how varying quantities of demonstrations (k=1, 2, or 3) affect agent performance, enabling research on the optimal number of examples needed

(2) K-shot 示范学习: 该数据集系统地探索不同示范数量 (k=1、2 或 3) 对代理性能的影响，支持对所需示范数量的最优研究

(3) Multi-dimensional similarity analysis: LearnGUI enables investigation of how different types of similarity between demonstration and query tasks influence learning efficacy and generalization capabilities

(3) 多维度相似性分析:LearnGUI 使得研究示范任务与查询任务之间不同类型相似性如何影响学习效果和泛化能力成为可能

This comprehensive approach allows for a nuanced analysis of how mobile GUI agents can leverage human demonstrations to improve task performance, especially in scenarios not covered by their training data.

这一综合方法允许细致分析移动 GUI 代理如何利用人类示范提升任务表现，尤其是在训练数据未覆盖的场景中。

3.2 Data Collection

3.2 数据收集

The LearnGUI dataset consists of two components: LearnGUI-Offline for systematic evaluation of few-shot learning capabilities across varying similarity conditions, and LearnGUI-Online for real-time assessment in an interactive environment. Both components share a unified action space to ensure consistent evaluation, as detailed in Table 2.

LearnGUI 数据集包含两个部分:LearnGUI-Offline 用于在不同相似度条件下系统评估少样本学习能力，LearnGUI-Online 用于在交互环境中进行实时评估。两个部分共享统一的动作空间以确保评估的一致性，详见表 2。

Table 2: LearnGUI Action Space

表 2:LearnGUI 动作空间

Action	Definition
CLICK[x, y]	Click at coordinates (x, y).
TYPE[text]	Type the specified text.
SWIPE [direction]	Swipe in the specified direction.
PRESS_HOME	Go to the home screen.
PRESS_BACK	Go back to the previous app screen.
PRESS_ENTER	Press the enter button.
TASK_COMPLETE[answer] answer inside brackets if required.	Mark the task as complete. Provide

操作	定义
点击 [x, y]	在坐标 (x, y) 处点击。
输入 [text]	输入指定的文本。
滑动 [方向]	向指定方向滑动。
按下主页键	返回主屏幕。
按下返回键	返回到上一个应用界面。
按下回车键	按下回车按钮。
任务完成 [答案]	标记任务为完成。如有需要，在括号内提供答案。

3.2.1 LearnGUI-Offline. We built LearnGUI-Offline by restructuring and enhancing the AMEX dataset [5], which contains 2,946 independent mobile tasks. To transform this resource for few-shot learning evaluation, we made several key modifications:

3.2.1 LearnGUI-Offline。我们通过重构和增强 AMEX 数据集 [5] 构建了 LearnGUI-Offline，该数据集包含 2,946 个独立的移动任务。为了将该资源转化为少样本学习评估，我们进行了若干关键修改:

Action Space Standardization. We refined the original action space to better align with real-world scenarios. First, we removed

动作空间标准化。我们优化了原始动作空间，使其更贴合真实场景。首先，我们移除了

Table 3: Statistics of LearnGUI dataset splits. Each split is analyzed across multiple dimensions: Tasks (number of tasks), Apps (number of applications covered), Step actions (total action steps), similarity metrics (Avg Ins/UI/Act $_{\text{Sim}}$), and distribution across four similarity profiles categorized by high (SH) and low (SL) UI and action similarity.

表 3:LearnGUI 数据集划分的统计信息。每个划分从多个维度进行分析: 任务数 (Tasks)、涵盖的应用数 (Apps)、动作步骤总数 (Step actions)、相似度指标 (平均插入/界面/动作相似度 $_{\text{Sim}}$)，以及按高 (SH) 和低 (SL) 界面与动作相似度分类的四种相似度分布。

Split	K-shot	Tasks	Apps	Step actions	Avg Inssm	Avg UISim	Avg ActSim	UISHActsh	UISHActsL	UISLACTSH	UISLACTSL
Offline-Train	1-shot	2,001	44	26,184	0.845	0.901	0.858	364	400	403	834
Offline-Train	2-shot	2,001	44	26,184	0.818	0.898	0.845	216	360	358	1,067
Offline-Train	3-shot	2,001	44	26,184	0.798	0.895	0.836	152	346	310	1,193
Offline-Test	1-shot	251	9	3,469	0.798	0.868	0.867	37	49	56	109
Offline-Test	2-shot	251	9	3,469	0.767	0.855	0.853	15	42	55	139
Offline-Test	3-shot	251	9	3,469	0.745	0.847	0.847	10	36	49	156
Online-Test	1-shot	101	20	1,423	-	-	-	-	-	-	-

划分	K 次示例	任务	应用	步骤操作	平均 Inssm	平均 UISim	平均 ActSim	UISHActsh	UISHActsL	UISLACTSH	UISLACTSL
离线训练	1 次示例	2,001	44	26,184	0.845	0.901	0.858	364	400	403	834
离线训练	2 次示例	2,001	44	26,184	0.818	0.898	0.845	216	360	358	1,067
离线训练	3 次示例	2,001	44	26,184	0.798	0.895	0.836	152	346	310	1,193
离线测试	1 次示例	251	9	3,469	0.798	0.868	0.867	37	49	56	109
离线测试	2 次示例	251	9	3,469	0.767	0.855	0.853	15	42	55	139
离线测试	3 次示例	251	9	3,469	0.745	0.847	0.847	10	36	49	156
在线测试	1 次示例	101	20	1,423	-	-	-	-	-	-	-

TASK_IMPOSSIBLE actions due to inconsistent labeling in the original dataset, which included errors such as tasks being incorrectly marked as impossible. Second, we enhanced TASK_COMPLETE to TASK_COMPLETE[answer] for information retrieval tasks. Many mobile tasks require returning specific information rather than just completion status. This aligns with both AMEX [5] and Android-World [23] paradigms.

由于原始数据集中存在标注不一致的问题 (例如任务被错误地标记为不可能完成), 导致 TASK_IMPOSSIBLE 操作无法执行。其次，我们将 TASK_COMPLETE 增强为 TASK_COMPLETE[answer] 以适应信息检索任务。许多移动任务需要返回具体信息，而不仅仅是完成状态。这与 AMEX [5] 和 Android-World [23] 的范式相一致。

K-shot Task Combinations. We constructed systematic k-shot task combinations through a multi-step process. We began by recovering the application context for each task through instruction and screenshot analysis, as the original dataset lacked explicit app labels. Next, we computed instruction similarity between tasks within the same application using the all-MiniLM-L6-v2 model. Finally, we created k-shot combinations $(k = 1, 2, 3)$ for each query task by selecting the k most similar tasks within the same application as support demonstrations, ensuring that the average similarity exceeded a minimum threshold of 0.6 . This process yielded 2,252 tasks with valid k-shot combinations.

K-shot 任务组合。我们通过多步骤流程构建了系统的 k-shot 任务组合。首先，通过指令和截图分析恢复每个任务的应用上下文，因为原始数据集缺少明确的应用标签。接着，使用 all-MiniLM-L6-v2 模型计算同一应用内任务间的指令相似度。最后，为每个查询任务创建 k-shot 组合 ($k = 1, 2, 3$)，通过选择同一应用中与之最相似的 k 个任务作为支持示范，确保平均相似度超过 0.6 的最低阈值。该过程产生了 2252 个具有有效 k-shot 组合的任务。

Similarity Measurement. To enable multi-dimensional similarity analysis, we computed metrics across three key dimensions. For Instruction Similarity, we utilized the scores calculated during the K-shot Task Combinations process. For UI Similarity, we merged the UI trees from all steps of each task and calculated similarity using TF-IDF vectorization and cosine similarity, capturing the visual and structural similarity of interfaces. For Action Similarity, following the DemoParser approach detailed in Section 4.1, we generated descriptive representations of each action and computed embedding-based cosine similarity between task pairs.

相似度测量。为了实现多维度的相似度分析，我们在三个关键维度上计算了指标。对于指令相似度，采用了 K-shot 任务组合过程中计算的分数。对于 UI 相似度，我们合并了每个任务所有步骤的 UI 树，利用 TF-IDF 向量化和余弦相似度计算，捕捉界面的视觉和结构相似性。对于动作相似度，遵循第 4.1 节中 DemoParser 方法，生成每个动作的描述性表示，并计算任务对之间基于嵌入的余弦相似度。

3.2.2 LearnGUI-Online. For evaluating mobile GUI agents in real-time interactive scenarios, we developed LearnGUI-Online based on the AndroidWorld environment [23]. While AndroidWorld provides 116 dynamically constructed task templates, it lacks human demonstration trajectories essential for few-shot learning evaluation.

3.2.2 LearnGUI-Online。为了评估移动 GUI 代理在实时交互场景中的表现，我们基于 AndroidWorld 环境 [23] 开发了 LearnGUI-Online。虽然 AndroidWorld 提供了 116 个动态构建的任务模板，但缺乏少样本学习评估所需的人类演示轨迹。

We identified 101 tasks suitable for human completion, excluding 15 tasks that proved challenging for human users. We then collected high-quality human demonstrations for these tasks. For tasks with dynamic elements, we generated specific instances and recorded corresponding demonstrations.

我们确定了 101 个适合人类完成的任务，排除了 15 个对人类用户来说较为困难的任务。随后，我们为这些任务收集了高质量的人类演示。对于包含动态元素的任务，我们生成了具体实例并记录了相应的演示。

The resulting LearnGUI-Online dataset provides a realistic testbed for evaluating few-shot learning capabilities in mobile GUI agents under authentic conditions.

由此产生的 LearnGUI-Online 数据集为在真实条件下评估移动 GUI 代理的少样本学习能力提供了一个现实的测试平台。

3.3 Dataset Statistics

3.3 数据集统计

Table 1 presents the comprehensive statistics of the LearnGUI dataset in comparison with existing datasets. With 2,353 instructions across 73 applications and an average of 13.2 steps per task, LearnGUI offers rich data for studying demonstration-based learning in mobile GUI agents. The dataset provides various k-shot combinations $(k = 1, 2, 3)$ for each task, along with multi-dimensional similarity metrics across instruction, UI, and action dimensions. This design enables systematic analysis of how different types and quantities of demonstrations affect learning outcomes. The similarity distribution reflects the natural variation in mobile tasks within applications, with a meaningful spread across similarity levels that allows for a detailed investigation of knowledge transfer under different conditions. A detailed visualization of these similarity distributions is provided in Appendix A.

表 1 展示了 LearnGUI 数据集与现有数据集的综合统计信息。LearnGUI 包含 2353 条指令，覆盖 73 个应用，平均每个任务 13.2 步，提供了丰富的数据用于研究基于示范的移动 GUI 代理学习。数据集为每个任务提供了多种 k-shot 组合 $(k = 1, 2, 3)$，并涵盖指令、UI 和动作三个维度的多维相似度指标。该设计支持系统分析不同类型和数量的示范如何影响学习效果。相似度分布反映了应用内移动任务的自然变异，具有有意义的相似度层次分布，便于在不同条件下深入研究知识迁移。附录 A 中提供了这些相似度分布的详细可视化。

3.4 Dataset Splits

3.4 数据集划分

We divided LearnGUI-Offline into training and testing splits to enable systematic evaluation of few-shot learning capabilities. Table 3 presents the detailed statistics of these splits, including the distribution of tasks across different similarity profiles.

我们将 LearnGUI-Offline 划分为训练集和测试集，以支持少样本学习能力的系统评估。表 3 展示了这些划分的详细统计，包括不同相似度特征下任务的分布情况。

The training set contains 2,001 tasks for each k-shot configuration (1, 2, and 3), spanning 44 applications with an average of 13.1 steps per task. The test set includes 251 tasks per k-shot configuration across 9 applications. Both splits maintain the same action space and similarity measurement methodology.

训练集包含每种 k-shot 配置 (1、2 和 3) 下的 2001 个任务，覆盖 44 个应用，平均每个任务 13.1 步。测试集包含每种 k-shot 配置下的 251 个任务，分布于 9 个应用。两个划分均保持相同的动作空间和相似度测量方法。

Based on empirical analysis, we established threshold values of 0.9447 for UI similarity and 0.9015 for action similarity to classify tasks into high (SH) and low (SL) similarity categories, enabling systematic analysis of how different similarity types affect learning from demonstrations.

基于经验分析，我们设定了 UI 相似度阈值为 0.9447，动作相似度阈值为 0.9015，用以将任务分类为高相似度 (SH) 和低相似度 (SL) 类别，从而系统分析不同相似度类型如何影响示范学习。

As shown in Figure 3, we classify tasks into four categories based on UI and action similarity:

如图 3 所示，我们基于 UI 和动作相似度将任务分为四类:

- $UI_{SH}Act_{SH}$: High UI similarity and high action similarity. For example, in a smart home app, two tasks that both involve adjusting the brightness of different lights in the living room would navigate through similar UI screens.

- $UI_{SH}Act_{SH}$: 高 UI 相似度且高动作相似度。例如，在智能家居应用中，两个任务均涉及调整客厅不同灯光的亮度，操作界面相似。

- $UI_{SH}Act_{SL}$: High UI similarity but low action similarity. For instance, in a smart home app, turning on all lights with a single button press versus adjusting each light's color temperature.

- $UI_{SH}Act_{SL}$: 高 UI 相似度但低动作相似度。例如，在智能家居应用中，一键开启所有灯光与分别调整每盏灯的色温。

- $UI_{SL}Act_{SH}$: Low UI similarity but high action similarity. For example, setting a schedule for lights versus setting a

- $UI_{SL}Act_{SH}$: 低 UI 相似度但高动作相似度。例如，设置灯光的定时计划与设置

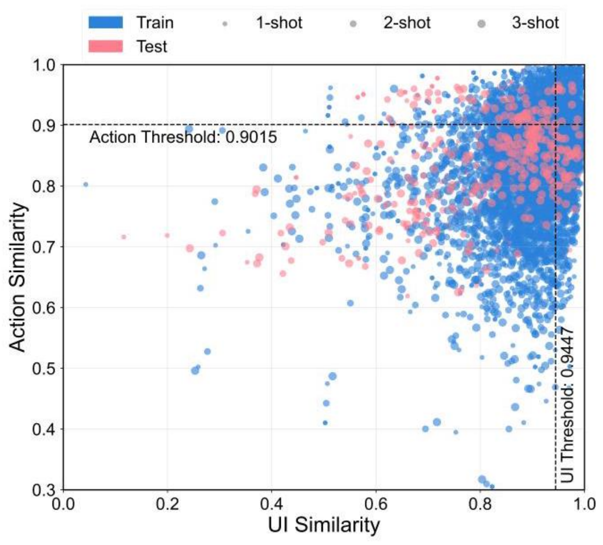


Figure 3: Joint distribution of UI similarity and action similarity in LearnGUI-Offline. The scatter plot shows the relationship between UI and action similarity measures across task pairs. The quadrant divisions represent our categorization of tasks into four profiles: $UI_{SH}Act_{SH}$, $UI_{SH}Act_{SL}$, $UI_{SL}Act_{SH}$, and $UI_{SL}Act_{SL}$, enabling analysis of how different similarity combinations affect learning transfer.

图 3:LearnGUI-Offline 中 UI 相似度与动作相似度的联合分布。散点图展示了任务对之间 UI 与动作相似度度量的关系。象限划分代表我们将任务分类为四种类型: $UI_{SH}Act_{SH}$, $UI_{SH}Act_{SL}$, $UI_{SL}Act_{SH}$, 和 $UI_{SL}Act_{SL}$ ， 便于分析不同相似度组合如何影响学习迁移。

schedule for the thermostat-different UI screens but similar action patterns.

恒温器的日程安排——不同的 UI 界面但相似的动作模式。

- UISLActsL: Low UI similarity and low action similarity. For instance, checking security camera footage versus creating a scene that coordinates multiple devices.

- UISLActsL: 低 UI 相似度且低动作相似度。例如，查看安全摄像头录像与创建协调多个设备的场景。

This categorization enables a detailed analysis of how different types of similarity affect learning efficacy. For instance, we can investigate whether UI similarity or action similarity has a greater impact on successful knowledge transfer from demonstrations.

该分类使我们能够详细分析不同类型的相似度如何影响学习效果。例如，我们可以探究 UI 相似度或动作相似度哪一项对示范知识成功迁移的影响更大。

Additionally, the LearnGUI-Online test set contains 101 tasks across 20 applications. Unlike the offline dataset, these tasks are evaluated in real time through direct interaction with the mobile environment.

此外，LearnGUI-Online 测试集包含 20 个应用中的 101 个任务。与离线数据集不同，这些任务通过与移动环境的实时交互进行评估。

The comprehensive structure of LearnGUI, with its carefully designed splits and similarity classifications, provides a resource for studying how mobile GUI agents can learn from demonstrations under varying conditions of task similarity and demonstration quantity.

LearnGUI 的全面结构，结合精心设计的划分和相似度分类，为研究移动 GUI 代理如何在不同任务相似度和示范数量条件下从示范中学习提供了资源。

4 METHOD: LEARNACT

4 方法:LEARNACT

Building on the insights from our LearnGUI dataset, we introduce LearnAct, a novel framework designed to break through the limitations of traditional training approaches for mobile GUI agents. Rather than pursuing universal generalization through extensive training data, LearnAct establishes demonstration-based learning as a paradigm for developing more adaptable, personalized, and practically deployable mobile GUI agents. As illustrated in Figure 4,

基于 LearnGUI 数据集的洞见，我们提出了 LearnAct，一种旨在突破传统移动 GUI 代理训练方法局限性的创新框架。LearnAct 不追求通过大量训练数据实现普适泛化，而是将基于示范的学习确立为开发更具适应性、个性化和实用部署能力的移动 GUI 代理的新范式。如图 4 所示,

LearnAct is a sophisticated multi-agent framework that automatically understands human demonstrations, generates instructional knowledge, and leverages this knowledge to assist mobile GUI agents in reasoning about unseen scenarios. The LearnAct framework consists of three specialized components, each addressing a critical aspect of demonstration-based learning: (1) DemoParser (Section 4.1), a knowledge generation agent that extracts usable knowledge from demonstration trajectories to form a knowledge base; (2) KnowSeeker (Section 4.2), a knowledge retrieval agent that searches the knowledge base for demonstration knowledge relevant to the current task; and (3) ActExecutor (Section 4.3), a task execution agent that combines user instructions, real-time GUI environment, and retrieved demonstration knowledge to perform tasks effectively.

LearnAct 是一个复杂的多代理框架，能够自动理解人类示范，生成指导性知识，并利用这些知识辅助移动 GUI 代理推理未知场景。LearnAct 框架由三个专门组件组成，分别解决基于示范学习的关键环节:(1)DemoParser(第 4.1 节)，一个知识生成代理，从示范轨迹中提取可用知识构建知识库；(2)KnowSeeker(第 4.2 节)，一个知识检索代理，搜索知识库中与当前任务相关的示范知识；(3)ActExecutor(第 4.3 节)，一个任务执行代理，结合用户指令、实时 GUI 环境和检索到的示范知识高效完成任务。

4.1 DemoParser

4.1 DemoParser

The DemoParser transforms raw human demonstrations into structured demonstration knowledge. It takes as input a raw action sequence (consisting of coordinates-based clicks, swipes, and text inputs) along with corresponding screenshots and task instructions. It then utilizes a vision-language model to generate semantically descriptive action descriptions that capture the essence of each demonstration step (e.g., ”On Search Page, click the search box, to enter keywords”). Building on these descriptions, it constructs a structured knowledge base that records both the high-level action semantics and the contexts in which they occur, as shown in Figure 5.

DemoParser 将原始人类示范转化为结构化的示范知识。其输入为原始动作序列 (包括基于坐标的点击、滑动和文本输入)、对应截图及任务指令。随后利用视觉-语言模型生成语义描述性动作说明，捕捉每一步示范的核心内容 (例如，“在搜索页面，点击搜索框，输入关键词”)。基于这些描述，构建结构化知识库，记录高层次动作语义及其发生的上下文，如图 5 所示。

Formally, DemoParser implements a knowledge generation function $G : \mathcal{I} \times \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{K}$, where $\mathcal{I}$ represents the space of instructions, $\mathcal{S}$ is the space of screenshot sequences, $\mathcal{A}$ is the space of action sequences, and $\mathcal{K}$ is the knowledge space. For each demonstration trajectory $(i, s, a) \in \mathcal{I} \times \mathcal{S} \times \mathcal{A}$, DemoParser generates a knowledge entry $k \in \mathcal{K}$ that encapsulates the demonstration in a semantically descriptive format, converting raw coordinate-based actions (e.g., CLICK[123,456]) into meaningful operation descriptions (e.g., ”click search box”).

形式上，DemoParser 实现了一个知识生成函数 $G : \mathcal{I} \times \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{K}$ ，其中 $\mathcal{I}$ 表示指令空间，$\mathcal{S}$ 为截图序列空间，$\mathcal{A}$ 为动作序列空间，$\mathcal{K}$ 为知识空间。对于每条示范轨迹 $(i, s, a) \in \mathcal{I} \times \mathcal{S} \times \mathcal{A}$ ，DemoParser 生成一个知识条目 $k \in \mathcal{K}$ ，以语义描述格式封装示范，将原始基于坐标的动作 (如 CLICK[123,456]) 转换为有意义的操作描述 (如“点击搜索框”)。

The knowledge generation process is decomposed into a sequence of description generation steps for each action in the demonstration trajectory. Let d_j represent the description for action a_j , which is generated using a context-aware description function $\delta : \mathcal{I} \times \mathcal{A}_j \times \mathcal{V}_j \times \mathcal{H}_{j-1} \rightarrow \mathcal{D}$, where $\mathcal{V}_j$ is the visual representation of action a_j execution and $\mathcal{H}_{j-1} = \{d_1, d_2, \ldots, d_{j-1}\}$ is the history of previous action descriptions.

知识生成过程被分解为演示轨迹中每个动作的描述生成步骤序列。设 d_j 表示动作 a_j 的描述，该描述通过上下文感知描述函数 $\delta : \mathcal{I} \times \mathcal{A}_j \times \mathcal{V}_j \times \mathcal{H}_{j-1} \rightarrow \mathcal{D}$ 生成，其中 $\mathcal{V}_j$ 是动作 a_j 执行的视觉表示，$\mathcal{H}_{j-1} = \{d_1, d_2, \ldots, d_{j-1}\}$ 是之前动作描述的历史记录。

Algorithm 1 in Appendix B.3 outlines the knowledge generation process. For each demonstration, DemoParser preserves the original task instruction and action sequence while generating semantically descriptive action descriptions. These descriptions provide crucial context about the purpose and significance of each action in the demonstration, enabling more effective knowledge transfer to new scenarios.

附录 B.3 中的算法 1 概述了知识生成过程。对于每个演示，DemoParser 在生成语义描述性动作描述的同时，保留原始任务指令和动作序列。这些描述提供了关于演示中每个动作目的和意义的重要上下文，从而实现更有效的知识迁移到新场景。

For intermediate actions, DemoParser analyzes a visual representation of the action execution, showing before-action and after-action screenshots with the action visualized (e.g., click locations highlighted). The framework combines this visual input with the task instruction, action history, and current action to generate a description that follows a standardized format: ”[On/In] [Screen

对于中间动作，DemoParser 分析动作执行的视觉表示，展示动作前后的截图并可视化动作 (例如，突出显示点击位置)。该框架将此视觉输入与任务指令、动作历史和当前动作结合，生成遵循标准格式的描述："[在] [屏幕

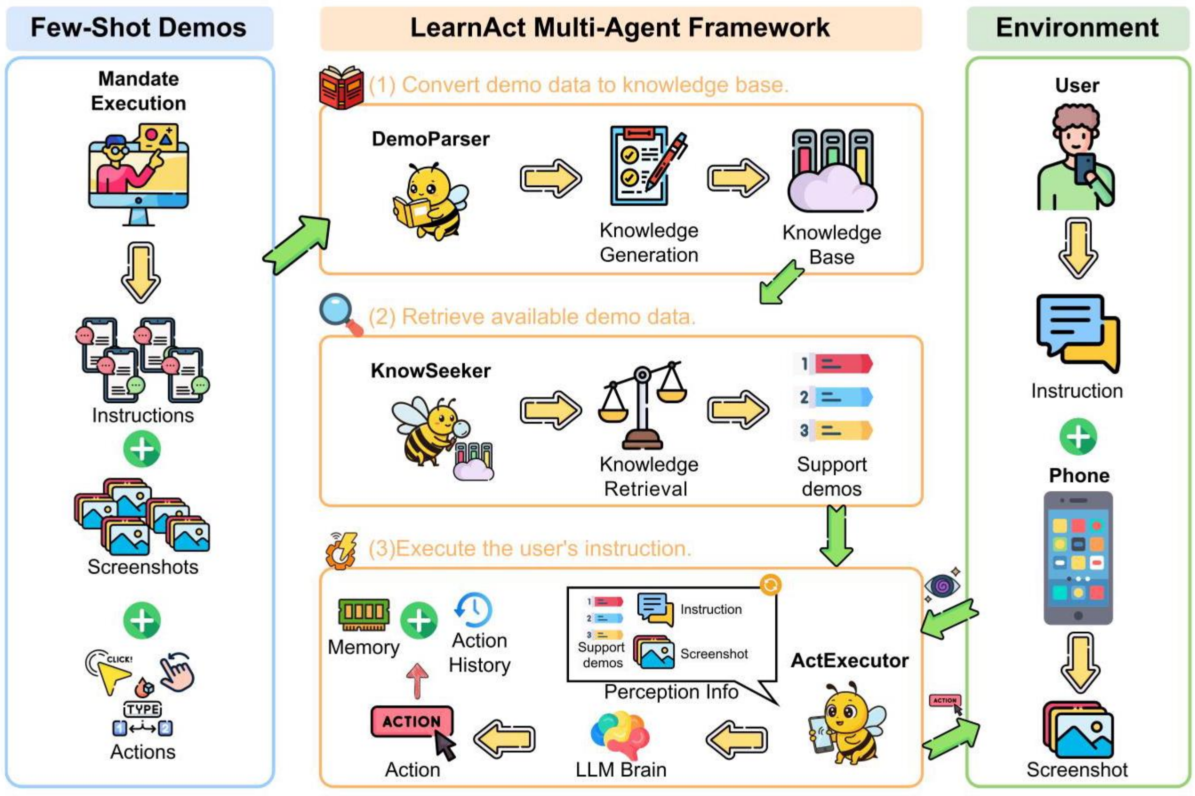


Figure 4: Illustration of the overall framework of LearnAct. Architecture diagram showing the three main

components (DemoParser, KnowSeeker, ActExecutor) and their interconnections within the LearnAct system, including data flow from human demonstrations to execution.

图 4:LearnAct 整体框架示意图。架构图展示了 LearnAct 系统中的三个主要组件 (DemoParser、KnowSeeker、ActExecutor) 及其相互连接，包括从人类演示到执行的数据流。

Name], [Action Details], to [Purpose]". For example: "On Home Screen, tap 'Settings' icon, to access device configuration." For terminal actions, DemoParser processes the final screenshot, task instruction, and complete action history to generate a conclusion in the format: "[On/In] [Screen], complete task, [Reason/Answer]"

名称], [动作细节], 以达到 [目的]"。例如:“在主屏幕，点击‘设置’图标，以访问设备配置。”对于终端动作，DemoParser 处理最终截图、任务指令和完整动作历史，生成格式为:“[在] [屏幕]，完成任务，[原因/答案]”的结论。

A distinctive feature of DemoParser is its memory mechanism, which captures critical information observed during task execution that may be necessary for future steps. The model identifies and annotates task-relevant information that is directly related to the user's instruction, will likely be needed in subsequent steps, and has not been previously recorded. These memory annotations are included in the action descriptions when appropriate: "[On/In] [Screen], [Action], to [Purpose]. [Memory: important information for future steps]". For example, in a shopping task, a memory annotation might capture: "[Memory: iPhone 13 Pro costs $999 with 128GB storage]". The detailed prompt for this memory mechanism is provided in Appendix B.1.

DemoParser 的一个显著特征是其记忆机制，该机制捕捉任务执行过程中观察到的可能对后续步骤必要的关键信息。模型识别并标注与用户指令直接相关、可能在后续步骤需要且尚未记录的任务相关信息。这些记忆注释在适当时包含在动作描述中:“[在] [屏幕]，[动作]，以达到 [目的]。[记忆: 未来步骤的重要信息]”。例如，在购物任务中，记忆注释可能是:“[记忆:iPhone 13 Pro 售价 999 美元，存储 128GB]”。该记忆机制的详细提示见附录 B.1。

This memory mechanism is particularly valuable for complex tasks requiring information retention across multiple steps, such as comparing prices, remembering account details, or tracking status changes. By transforming raw demonstrations into structured, semantically descriptive knowledge with memory capabilities, De-moParser enables effective knowledge transfer from human demonstrations to automated task execution.

该记忆机制对于需要跨多个步骤保留信息的复杂任务尤为重要，如价格比较、记忆账户详情或跟踪状态变化。通过将原始演示转化为具有记忆能力的结构化语义描述知识，DemoParser 实现了从人类演示到自动化任务执行的有效知识迁移。

4.2 KnowSeeker

4.2 KnowSeeker

KnowSeeker is the retrieval component of the LearnAct framework that identifies demonstration knowledge most relevant to the current task context. As depicted in Figure 6, this agent serves as the bridge between the knowledge base generated by DemoParser and the execution environment of ActExecutor. While DemoParser focuses on transforming demonstrations into structured knowledge, KnowSeeker specializes in efficiently accessing and selecting the most applicable knowledge for a specific task, addressing the critical challenge of knowledge relevance in few-shot learning scenarios.

KnowSeeker 是 LearnAct 框架中的检索组件，负责识别与当前任务上下文最相关的演示知识。如图 6 所示，该代理作为 DemoParser 生成的知识库与 ActExecutor 执行环境之间的桥梁。DemoParser 专注于将演示转化为结构化知识，而 KnowSeeker 专长于高效访问和选择特定任务最适用的知识，解决了少样本学习场景中知识相关性的关键挑战。

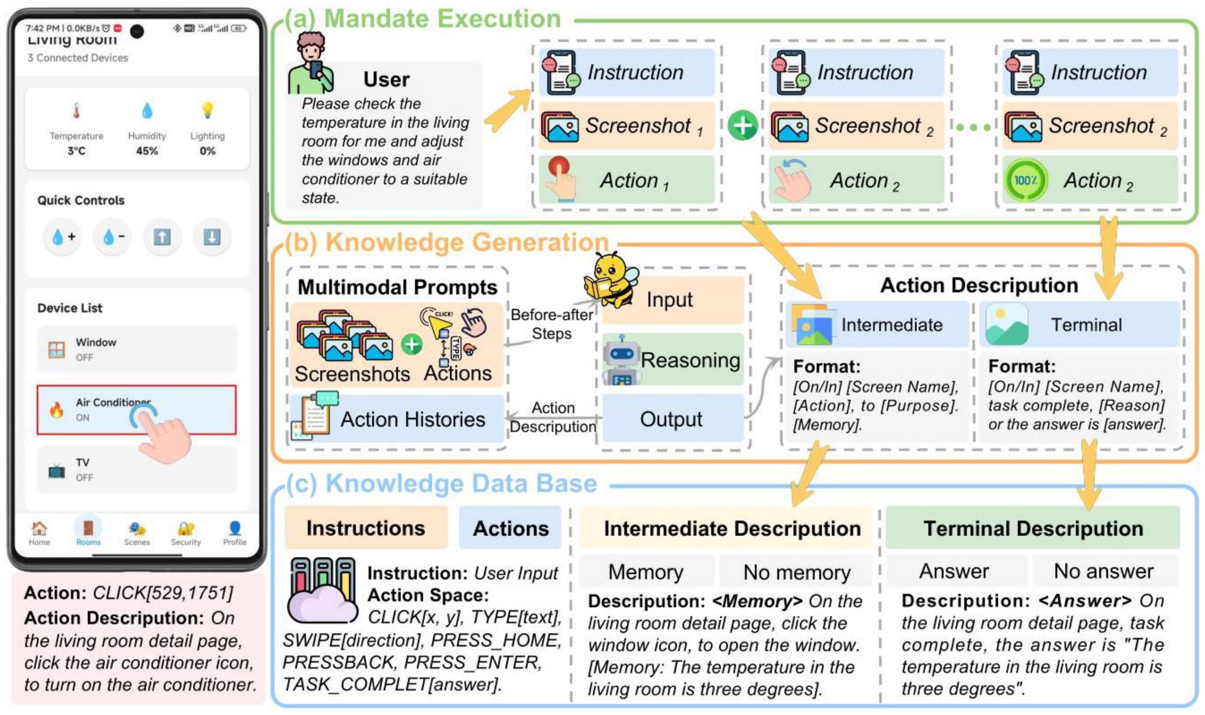
Formally, KnowSeeker implements a retrieval function $R : \mathcal{I} \times \mathcal{K} \rightarrow \mathcal{K}^{(s)}$, where $\mathcal{I}$ is the instruction space, $\mathcal{K}$ is the knowledge base, and $\mathcal{K}^{(s)} \subset \mathcal{K}$ is a subset of knowledge entries determined to be relevant for the given instruction. This retrieval process is crucial for effective knowledge utilization, as it filters the potentially vast knowledge base to focus exclusively on demonstrations that offer valuable insights for the current task.

形式上, KnowSeeker 实现了检索函数 $R : \mathcal{I} \times \mathcal{K} \rightarrow \mathcal{K}^{(s)}$, 其中 $\mathcal{I}$ 是指令空间, $\mathcal{K}$ 是知识库, $\mathcal{K}^{(s)} \subset \mathcal{K}$ 是针对给定指令确定的相关知识条目子集。该检索过程对于有效利用知识至关重要，因为它过滤了潜在庞大的知识库，专注于为当前任务提供有价值见解的演示。

The core of KnowSeeker's retrieval mechanism relies on semantic similarity measurement between the current task instruction and the instructions associated with demonstrations in the knowledge base. This similarity-based retrieval can be formally defined as:

KnowSeeker 检索机制的核心依赖于当前任务指令与知识库中演示指令之间的语义相似度测量。该基于相似度的检索可形式化定义为:

$$R(i, K) = \{k_j \in K \mid \text{sim}(i, i_j) \geq \tau_s\}_{j=1}^{\text{top}-k} \tag{1}$$



Please check the temperature in the living room for me and adjust the windows and air conditioner to a suitable state.

Action: *CLICK[529,1751]*
Action Description: *On the living room detail page, click the air conditioner icon, to turn on the air conditioner.*

Format: *[On/In] [Screen Name], [Action], to [Purpose]. [Memory].*

Format: *[On/In] [Screen Name], task complete, [Reason] or the answer is [answer].*

Instruction: *User Input*
Action Space: *CLICK[x, y], TYPE[text], SWIPE[direction], PRESS_HOME, PRESSBACK, PRESS_ENTER, TASK_COMPLET[answer].*

Description: <Memory> *On the living room detail page, click the window icon, to open the window. [Memory: The temperature in the living room is three degrees].*

Description: <Answer> *On the living room detail page, task complete, the answer is "The temperature in the living room is three degrees".*

Figure 5: Pipeline of DemoParser Agent. Input instructions and corresponding actions and screenshots; output low-level action descriptions and create knowledge database. This process transforms high-level user instructions into precise operation sequences while building a reusable domain knowledge base to improve mobile interface interaction automation efficiency.

图 5:DemoParser 代理流程。输入指令及对应动作和截图；输出低级动作描述并创建知识数据库。该过程将高级用户指令转化为精确操作序列，同时构建可复用的领域知识库，以提升移动界面交互自动化效率。

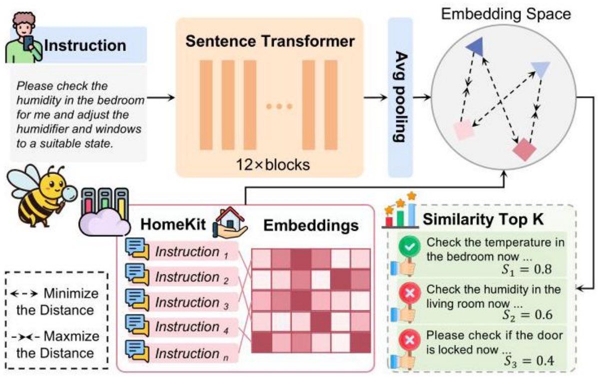


Figure 6: Pipeline of KnowSeeker Agent. The KnowSeeker Agent converts demo trajectories from the knowledge base into a vector database. When executing user tasks, KnowSeeker retrieves the top-k relevant demos from the vector database for subsequent use. This approach enables efficient retrieval of similar demonstrations to assist with new task execution.

图 6:KnowSeeker 代理流程。KnowSeeker 代理将知识库中的演示轨迹转换为向量数据库。在执行用户任务时，KnowSeeker 从向量数据库中检索前 k 个相关演示以供后续使用。这种方法实现了高效检索相似演示，辅助新任务的执行。

where i is the current instruction, i_j is the instruction associated with knowledge entry k_j, sim $(\cdot,\cdot)$ is a similarity function, τ_s is a

其中 i 是当前指令，i_j 是与知识条目相关的指令，k_j, sim $(\cdot,\cdot)$ 是相似度函数，τ_s 是一个

similarity threshold, and top $-k$ indicates selection of the k most similar entries.

相似度阈值，top $-k$ 表示选择最相似的 k 条目。

To implement this similarity measurement efficiently, KnowSeeker employs a two-phase approach:

为高效实现该相似度测量，KnowSeeker 采用两阶段方法:

(1) Embedding Generation: Instructions are transformed into dense vector representations using a pre-trained sentence transformer model. Specifically, we utilize the all-MiniLM-L6-v2 model, which offers an optimal balance between computational efficiency and semantic representational power. This model has been fine-tuned on

diverse natural language understanding tasks, making it particularly well-suited for capturing the semantic essence of mobile GUI task instructions.

(1) 嵌入生成: 使用预训练的句子转换器模型将指令转换为密集向量表示。具体而言，我们采用 all-MiniLM-L6-v2 模型，该模型在计算效率与语义表示能力之间实现了最佳平衡。该模型经过多样化自然语言理解任务的微调，特别适合捕捉移动 GUI 任务指令的语义精髓。

(2) Similarity Computation: The cosine similarity between embedding vectors is calculated to quantify the semantic relationship between instructions. For instructions i and i_j with corresponding embeddings e_i and e_j , the similarity is computed as:

(2) 相似度计算: 通过计算嵌入向量间的余弦相似度来量化指令间的语义关系。对于指令 i 和 i_j 及其对应嵌入 e_i 和 e_j ，相似度计算公式为:

$$\operatorname{sim}(i, i_j) = \frac{e_i \cdot e_j}{\left\|e_i\right\| \cdot \left\|e_j\right\|} \tag{2}$$

To optimize retrieval efficiency, KnowSeeker pre-computes em-beddings for all instructions in the knowledge base during initialization. This approach transforms the potentially expensive operation of computing pairwise similarities during runtime into a more manageable vector comparison task. The pre-computation process is described as:

为优化检索效率，KnowSeeker 在初始化时预先计算知识库中所有指令的嵌入。此举将运行时计算成对相似度的高开销操作转化为更易管理的向量比较任务。预计算过程描述如下:

$$E = \{e_j = f_{\text{embed}}(i_j) \mid k_j \in K\} \tag{3}$$

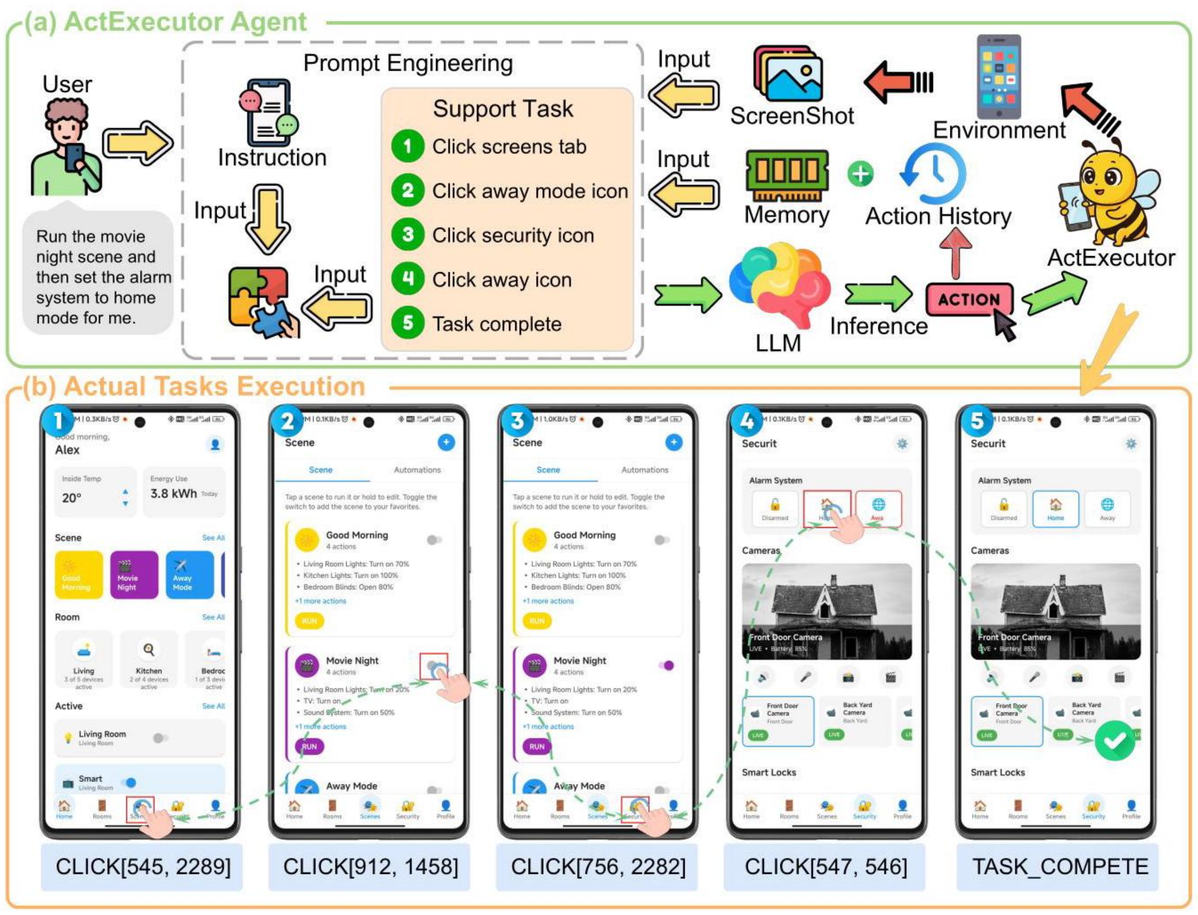


Figure 7: Pipeline of ActExecutor Agent. The ActExecutor Agent executes the low-level action descriptions generated by the Action Planner Agent. It uses the KnowSeeker Agent to retrieve relevant demonstrations from the knowledge base and execute the actions in the demonstrations. This approach enables efficient execution of low-level actions to assist with new task execution.

图 7:ActExecutor 代理流程。ActExecutor 代理执行由 Action Planner 代理生成的低级动作描述。它利用 KnowSeeker 代理从知识库中检索相关演示并执行演示中的动作。这种方法实现了低级动作的高效执行，辅助新任务的完成。

where f_{embed} is the embedding function implemented by the sentence transformer model.

其中 f_{embed} 是由句子转换器模型实现的嵌入函数。

During task execution, when presented with a new instruction i , KnowSeeker: 1. Computes the embedding $e_i = f_{\text{embed}}(i)$ 2. Calculates similarities $S = \{\operatorname{sim}(e_i, e_j) \mid e_j \in E\}$ 3. Selects the top- k knowledge entries based on similarity scores

在任务执行过程中，面对新指令 i 时，KnowSeeker:1. 计算嵌入 $e_i = f_{\text{embed}}(i)$ 2. 计算相似度 $S = \{\operatorname{sim}(e_i, e_j) \mid e_j \in E\}$ 3. 基于相似度分数选择前 k 条知识条目

This approach ensures that knowledge retrieval scales efficiently with the size of the knowledge base, enabling rapid identification of relevant demonstrations even as the framework's experiential knowledge grows over time.

By systematically identifying the most relevant demonstration knowledge, KnowSeeker enables ActEx-ecutor to perform tasks more effectively, particularly in unfamiliar scenarios.

该方法确保知识检索随着知识库规模的增长依然高效，能够快速识别相关演示。通过系统地识别最相关的演示知识，KnowSeeker 使 ActExecutor 能够更有效地执行任务，尤其是在陌生场景中。

4.3 ActExecutor

4.3 ActExecutor

ActExecutor is the execution component of the LearnAct framework that translates retrieved demonstration knowledge into effective actions in the target environment. As illustrated in Figure 7, this agent represents the culmination of the LearnAct pipeline, integrating user instructions, real-time GUI observations, and demonstration knowledge to navigate even unfamiliar mobile applications successfully. While DemoParser creates structured knowledge and KnowSeeker retrieves relevant demonstrations, ActExecutor applies this knowledge to solve practical tasks, addressing the critical challenge of knowledge utilization in few-shot learning scenarios.

ActExecutor 是 LearnAct 框架中的执行组件，将检索到的演示知识转化为目标环境中的有效动作。如图 7 所示，该代理代表 LearnAct 流程的终点，整合用户指令、实时 GUI 观察和演示知识，成功导航甚至陌生的移动应用。DemoParser 负责创建结构化知识，KnowSeeker 负责检索相关演示，而 ActExecutor 则应用这些知识解决实际任务，攻克少样本学习场景中知识利用的关键难题。

ActExecutor implements the POMDP framework introduced earlier, with the critical enhancement of incorporating demonstration knowledge into the decision-making process. The execution process can be formally described as a sequential decision-making loop that iteratively selects actions $a_t \in \mathcal{A}$ based on current observations $o_t \in O$ and demonstration knowledge $\mathcal{D}$, following policy $\pi : \mathcal{O} \times \mathcal{D} \rightarrow \mathcal{A}$.

ActExecutor 实现了前述的部分可观测马尔可夫决策过程 (POMDP) 框架，关键改进是将演示知识纳入决策过程。执行过程可形式化描述为一个序贯决策循环，基于当前观察 $o_t \in O$ 和演示知识 $\mathcal{D}$ ，按照策略 $\pi : \mathcal{O} \times \mathcal{D} \rightarrow \mathcal{A}$ 迭代选择动作 $a_t \in \mathcal{A}$ 。

The ActExecutor policy π is implemented through a large vision-language model that processes a carefully constructed prompt integrating all available information sources. This prompt-based policy can be expressed as:

ActExecutor 策略 π 通过一个大型视觉语言模型实现，该模型处理精心构建的提示，整合所有可用信息源。该基于提示的策略可表达为:

$$\pi\left(o_t, \mathcal{D}\right) = f_{LLM}\left(P\left(i, o_t, h_{t-1}, \mathcal{D}\right)\right) \tag{4}$$

where i is the user instruction, o_t is the current observation (screenshot), h_{t-1} is the action history up to time $t-1, \mathcal{D}$ is the retrieved demonstration knowledge, P is a prompt construction function, and f_{LLM} is the LLM-based decision function.

其中 i 是用户指令，o_t 是当前观察 (截图)，h_{t-1} 是截至时间的动作历史，$t-1, \mathcal{D}$ 是检索到的演示知识，P 是提示构建函数，f_{LLM} 是基于大型语言模型的决策函数。

Algorithm 2 in Appendix B.3 outlines the execution process. For each task, ActExecutor processes the user instruction and screen-shot observations through a sequence of perception, decision, and action phases until the task is completed or a maximum step limit is reached.

附录 B.3 中的算法 2 概述了执行过程。对于每个任务，ActExecutor 通过感知、决策和行动三个阶段处理用户指令和屏幕截图观察，直到任务完成或达到最大步数限制。

The execution process integrates three key phases:

执行过程整合了三个关键阶段:

(1) Perception Phase: ActExecutor perceives the current state of the mobile device through screenshot observations o_t . These observations provide the visual context essential for understanding the available interaction options and current application state.

(1) 感知阶段:ActExecutor 通过屏幕截图观察 o_t 感知移动设备的当前状态。这些观察提供了理解可用交互选项和当前应用状态所必需的视觉上下文。

(2) Decision Phase: The agent constructs a comprehensive prompt that integrates the user instruction i , current observation o_t , action history h , and retrieved demonstrations $\mathcal{D}$. This prompt is processed by a large vision-language model using templates detailed in Appendix B.2, resulting in a selected action from the predefined action space described in Table 2.

(2) 决策阶段: 代理构建一个综合提示，整合用户指令 i 、当前观察 o_t 、动作历史 h 和检索到的示范 $\mathcal{D}$ 。该提示通过附录 B.2 中详细说明的模板，由大型视觉语言模型处理，最终从表 2 描述的预定义动作空间中选择一个动作。

(3) Action Phase: The selected action a_t is executed in the mobile environment, generating a state transition according to the transition function $\mathcal{T}$ of the POMDP. Additionally, the agent generates a description d_t of the executed action using a process similar to DemoParser's description generation, which serves as part of the action history for subsequent steps.

(3) 行动阶段: 选定的动作 a_t 在移动环境中执行，根据部分可观测马尔可夫决策过程 (POMDP) 的转移函数 $\mathcal{T}$ 产生状态转移。此外，代理使用类似于 DemoParser 描述生成的过程生成执行动作的描述 d_t ，作为后续步骤动作历史的一部分。

The prompt construction function P plays a critical role in Ac-tExecutor's effectiveness. It integrates the agent's role definition, demonstration examples, task and observation context, action history, and the action space definition into a comprehensive prompt that guides the model's decision-making.

提示构建函数 P 在 ActExecutor 的有效性中起关键作用。它将代理的角色定义、示范示例、任务和观察上下文、动作历史及动作空间定义整合成一个综合提示，引导模型的决策过程。

This approach enables ActExecutor to leverage demonstrations as exemplars that guide its decision-making process. When faced with a novel UI state, the agent identifies analogous situations from demonstrations and adapts the demonstrated actions to the current context. This capability is particularly valuable for handling out-of-distribution scenarios where the agent lacks direct experience.

该方法使 ActExecutor 能够利用示范作为范例指导其决策过程。当遇到新的用户界面状态时，代理识别示范中的类似情境，并将示范动作适应当前上下文。这一能力对于处理代理缺乏直接经验的分布外场景尤为重要。

By closing the loop between demonstration knowledge and task execution, ActExecutor completes the LearnAct framework's end-to-end pipeline for demonstration-based learning. The combination of knowledge generation (DemoParser), knowledge retrieval (KnowSeeker), and knowledge-guided execution (ActExecutor) enables effective few-shot learning for mobile GUI agents, addressing the fundamental challenge of generalization to unseen scenarios with minimal examples.

通过闭合示范知识与任务执行之间的循环，ActExecutor 完成了 LearnAct 框架基于示范学习的端到端流程。知识生成 (DemoParser)、知识检索 (KnowSeeker) 和知识引导执行 (ActExecutor) 的结合，实现了移动 GUI 代理的有效少样本学习，解决了以最少示例泛化到未见场景的根本挑战。

5 EXPERIMENTS

5 实验

We conducted comprehensive evaluations of the LearnAct framework through both offline and online experiments. The offline experiments were performed on the LearnGUI-Offline dataset to evaluate step-by-step task execution capabilities, while the online experiments utilized the LearnGUI-Online platform to assess end-to-end task completion in real-world interactive scenarios. We evaluated a diverse set of models, including both commercial (e.g., Gemini- 1.5-Pro [26]) and open-source models (e.g., UI-TARS-7B-SFT [22], Qwen2-VL-7B [31]), to demonstrate the broad applicability of our approach across different model architectures and capabilities.

我们通过离线和在线实验对 LearnAct 框架进行了全面评估。离线实验在 LearnGUI-Offline 数据集上进行，以评估逐步任务执行能力；在线实验利用 LearnGUI-Online 平台，评估真实交互场景中的端到端任务完成情况。我们评估了多种模型，包括商业模型 (如 Gemini-1.5-Pro [26]) 和开源模型 (如 UI-TARS-7B-SFT [22]、Qwen2-VL-7B [31])，以展示我们方法在不同模型架构和能力上的广泛适用性。

5.1 Experiment Setup

5.1 实验设置

The diverse similarity profiles in LearnGUI provide a unique opportunity to evaluate mobile GUI agents' capabilities. Our experiments have two primary goals: (1) to evaluate the feasibility and effectiveness of enhancing mobile agents through few-shot demonstrations as a means to overcome the limitations of traditional pre-training or fine-tuning approaches; and (2) to investigate how different factors such as demonstration quantity (k=1,2,3) and various similarity aspects (instruction, UI, and action) influence the effectiveness of demonstration-based learning.

LearnGUI 中多样的相似性特征为评估移动 GUI 代理能力提供了独特机会。我们的实验有两个主要目标:(1) 评估通过少样本示范增强移动代理的可行性和有效性，以克服传统预训练或微调方法的局限；(2) 探讨示范数量 (k=1,2,3) 及不同相似性维度 (指令、UI 和动作) 如何影响基于示范学习的效果。

Implementation Details. We conducted experiments with three foundation models: Gemini-1.5-Pro [26], UI-TARS-7B-SFT [22], and Qwen2-VL-7B [31]. For all models, we set the temperature to zero to obtain deterministic responses. For Qwen2-VL-7B [31] and UI-TARS-7B-SFT [22], we employed parameter-efficient fine-tuning using LoRA with rank 64, alpha 128, and dropout probability 0.1. We targeted all modules while freezing the vision encoder to ensure computational efficiency. Training used a learning rate of 1e-5 with cosine scheduling, batch size of 1, gradient accumulation over 8 steps, a warmup ratio of 0.001, and was conducted for 1 epoch. All fine-tuning experiments were conducted on 8 NVIDIA L40S GPUs. For offline experiments, Gemini-1.5-Pro [26] was evaluated directly on the LearnGUI-Offline test set without additional training. UI-TARS-7B-SFT [22] and Qwen2-VL-7B [31] were fine-tuned on the LearnGUI-Offline training set before evaluation. For online experiments, we deployed all models except Gemini-1.5-Pro [26] (which showed limited task completion capabilities in preliminary tests despite accuracy improvements) to the LearnGUI-Online environment, using 1-shot demonstration retrieval for all LearnAct-enhanced models.

实现细节。我们使用三种基础模型进行了实验:Gemini-1.5-Pro [26]、UI-TARS-7B-SFT [22] 和 Qwen2-VL-7B [31]。对于所有模型，我们将温度设置为零以获得确定性响应。对于 Qwen2-VL-7B [31] 和 UI-TARS-7B-SFT [22]，我们采用了参数高效微调方法 LoRA，秩为 64，alpha 为 128，丢弃概率为 0.1。我们针对所有模块进行微调，同时冻结视觉编码器以确保计算效率。训练使用学习率 1e-5，采用余弦调度，批量大小为 1，梯度累积 8 步，预热比例为 0.001，训练 1 个 epoch。所有微调实验均在 8 块 NVIDIA L40S GPU 上进行。离线实验中，Gemini-1.5-Pro [26] 直接在 LearnGUI-Offline 测试集上评估，无需额外训练。UI-TARS-7B-SFT [22] 和 Qwen2-VL-7B [31] 则先在 LearnGUI-Offline 训练集上微调后再评估。在线实验中，除 Gemini-1.5-Pro [26](尽管准确率有所提升，但在初步测试中任务完成能力有限) 外，所有模型均部署到 LearnGUI-Online 环境，所有 LearnAct 增强模型均采用一-shot 示范检索。

Baselines. To rigorously evaluate our approach, we compared LearnAct against several baselines. These include: (1) SPHINX-GUI Agent, the original agent developed for the AMEX dataset [5], providing a reference point for task execution on similar data; (2) Zero-shot inference versions of all models (Gemini-1.5-Pro [26], UI-TARS-7B-SFT [22], and Qwen2-VL-7B [31]) within the Learn-Act framework but without demonstration knowledge, maintaining identical execution environments for fair comparison; and (3) For online evaluation, we additionally compared against GPT-4o, Gemini-Pro-1.5, Claude Computer-Use, and Aguvis to benchmark against current advanced systems.

基线方法。为了严格评估我们的方法，我们将 LearnAct 与多个基线进行了比较。这些基线包括:(1)SPHINX-GUI Agent，针对 AMEX 数据集 [5] 开发的原始代理，作为类似数据任务执行的参考点；(2) 所有模型 (Gemini-1.5-Pro [26]、UI-TARS-7B-SFT [22] 和 Qwen2-VL-7B [31]) 在 Learn-Act 框架下的零样本推理版本，但不使用示范知识，保持相同执行环境以保证公平比较；(3) 在线评估中，我们还与 GPT-4o、Gemini-Pro-1.5、Claude Computer-Use 和 Aguvis 进行了对比，以与当前先进系统进行基准测试。

Evaluation Metrics. For offline evaluation, we adopted mainstream evaluation protocols widely used in recent mobile GUI agent research, such as UI-TARS [22] and OS-ATLAS [39]. Specifically, we

评估指标。对于离线评估，我们采用了近年来移动 GUI 代理研究中广泛使用的主流评估协议，如 UI-TARS [22] 和 OS-ATLAS [39]。具体而言，我们

Table 4: Performance comparison of mobile GUI agents on LearnGUI-Offline dataset (action match accuracy %). Results show absolute values and relative improvements [in brackets] compared to baselines. Performance is evaluated across different models and number of support examples (1/2/3 -shot) .

表 4:LearnGUI-Offline 数据集上移动 GUI 代理的性能比较 (动作匹配准确率%)。结果展示了绝对值及相对于基线的相对提升 [括号内]。性能评估涵盖不同模型及支持示例数量 (1/2/3 -shot) 。

Models	Method	Supports	Average	Gmail	Booking	Music	SHEIN	NBC	CityMapper	ToDo	Signal	Yelp
SPHINX-GUI Agent[5]	AMEX	0-shot	67.2	45.9	64.5	74.4	71.8	70.3	67.4	79.3	64.9	66.3
gemini-1.5-pro	Baseline	0-shot	19.3	20.1	16.4	24.5	10.2	35.6	14.1	17.4	27.9	15.2
		1-shot	51.7 [+32.4]	55.5	47.1	60.0	35.7	56.4	54.7	60.6	63.1	54.6
	LearnAct	2-shot	55.6 [+36.3]	57.5	53.2	55.3	39.6	56.1	58.2	68.1	69.7	60.0
		3-shot	57.7 [+38.4]	58.4	56.6	54.6	43.9	53.9	69.4	69.2	70.5	57.6
UI-TARS-7B-SFT	Baseline	0-shot	77.5	68.1	81.0	81.1	72.9	80.9	70.6	66.0	92.6	82.4
		1-shot	82.8 [+5.3]	79.9	82.9	86.6	75.7	86.3	79.4	84.0	89.3	83.0
	LearnAct	2-shot	81.9 [+4.4]	80.1	80.7	86.2	76.1	87.2	80.0	83.7	84.4	84.2
		3-shot	82.1 [+4.6]	79.9	80.9	86.2	75.7	86.9	81.2	85.8	84.4	84.2
Qwen2-VL-7B	Baseline	0-shot	71.8	60.8	73.9	76.0	65.5	75.5	62.9	78.7	82.8	69.1
		1-shot	77.3 [+5.5]	75.0	77.5	77.8	69.8	83.5	72.9	78.0	83.6	78.8
	LearnAct	2-shot	78.5 [+6.7]	75.0	78.0	77.8	73.3	86.0	73.5	81.9	87.7	77.6
		3-shot	79.4 [+7.6]	75.0	78.8	78.6	72.6	87.8	77.1	82.6	87.7	80.6

模型	方法	支持	平均	Gmail	Booking	音乐	SHEIN	NBC	CityMapper	待办事项	Signal	Yelp
SPHINX-GUI Agent[5]	AMEX	零样本	67.2	45.9	64.5	74.4	71.8	70.3	67.4	79.3	64.9	66.3
gemini-1.5-pro	基线	零样本	19.3	20.1	16.4	24.5	10.2	35.6	14.1	17.4	27.9	15.2
		一样本	51.7 [+32.4]	55.5	47.1	60.0	35.7	56.4	54.7	60.6	63.1	54.6
	LearnAct	二样本	55.6 [+36.3]	57.5	53.2	55.3	39.6	56.1	58.2	68.1	69.7	60.0
		三样本	57.7 [+38.4]	58.4	56.6	54.6	43.9	53.9	69.4	69.2	70.5	57.6
UI-TARS-7B-SFT	基线	零样本	77.5	68.1	81.0	81.1	72.9	80.9	70.6	66.0	92.6	82.4
		一样本	82.8 [+5.3]	79.9	82.9	86.6	75.7	86.3	79.4	84.0	89.3	83.0
	LearnAct	二样本	81.9 [+4.4]	80.1	80.7	86.2	76.1	87.2	80.0	83.7	84.4	84.2
		三样本	82.1 [+4.6]	79.9	80.9	86.2	75.7	86.9	81.2	85.8	84.4	84.2
Qwen2-VL-7B	基线	零样本	71.8	60.8	73.9	76.0	65.5	75.5	62.9	78.7	82.8	69.1
		一样本	77.3 [+5.5]	75.0	77.5	77.8	69.8	83.5	72.9	78.0	83.6	78.8
	LearnAct	二样本	78.5 [+6.7]	75.0	78.0	77.8	73.3	86.0	73.5	81.9	87.7	77.6
		三样本	79.4 [+7.6]	75.0	78.8	78.6	72.6	87.8	77.1	82.6	87.7	80.6

measured step accuracy, which consists of two components: action type accuracy and action match accuracy. Action type accuracy measures the percentage of steps where the predicted action type (CLICK, TYPE, SWIPE, etc.) matches the ground truth. Action match accuracy measures the percentage of steps where both the action type and its parameters are correct, following standard evaluation criteria. For CLICK actions, coordinates are considered correct if they fall within 14% of the screen width from the ground truth. For TYPE actions, the content is correct if the F1 score between prediction and ground truth exceeds 0.5. For SWIPE actions, the direction must precisely match the ground truth. For other actions (e.g., PRESS_BACK), an exact match is required. For TASK_COMPLETE actions, we only verify the action type and ignore the answer field. For online evaluation, we measured the task success rate (SR), which represents the percentage of tasks completed successfully in the real-time interactive environment.

测量步骤准确率，包括两个组成部分: 动作类型准确率和动作匹配准确率。动作类型准确率衡量预测的动作类型 (点击 CLICK、输入 TYPE、滑动 SWIPE 等) 与真实标签匹配的步骤百分比。动作匹配准确率衡量动作类型及其参数均正确的步骤百分比，遵循标准评估标准。对于点击动作，坐标若在屏幕宽度的 14% 范围内即视为正确。对于输入动作，预测与真实内容的 F1 分数超过 0.5 即视为正确。对于滑动动作，方向必须与真实标签完全一致。对于其他动作 (如按返回键 PRESS_BACK)，要求完全匹配。对于任务完成动作 TASK_COMPLETE，仅验证动作类型，忽略答案字段。在线评估中，我们测量了任务成功率 (SR)，表示在实时交互环境中成功完成任务的百分比。

5.2 Main Results

5.2 主要结果

5.2.1 Offline Agent Capability Evaluation. Table 4 presents the performance comparison of different models on the LearnGUI-Offline dataset. The results demonstrate the substantial improvements achieved by the LearnAct framework across all tested models. Gemini-1.5-Pro [26] shows the most dramatic improvement, with performance increasing from 19.3% to 51.7% (+32.4%) with just a single demonstration, and further improving to 57.7% (+38.4%) with three demonstrations. This represents a 198.9% relative improvement, highlighting the powerful potential of demonstration-based learning even for advanced foundation models. UI-TARS- 7B-SFT [22], despite already having strong zero-shot performance (77.5%), still achieves significant gains with LearnAct, reaching 82.8% (+5.3%) with a single demonstration. This indicates that even models specifically fine-tuned for GUI tasks can benefit from demonstration knowledge. Qwen2-VL-7B [31] demonstrates consistent improvement from 71.8% to 77.3% (+5.5%) with one demonstration, and to 79.4% (+7.6%) with three demonstrations, confirming that

5.2.1 离线代理能力评估。表 4 展示了不同模型在 LearnGUI-Offline 数据集上的性能对比。结果表明 LearnAct 框架在所有测试模型上均取得了显著提升。Gemini-1.5-Pro [26] 的提升最为显著，单次示范下性能从 19.3% 提升至 51.7%(+32.4%)，三次示范下进一步提升至 57.7%(+38.4%)，相对提升达 198.9%，凸显了基于示范学习对先进基础模型的强大潜力。UI-TARS-7B-SFT [22] 尽管已具备较强的零样本性能 (77.5%)，但通过 LearnAct 仍实现显著提升，单次示范下达到 82.8%(+5.3%)，表明即使是专门针对 GUI 任务微调的模型也能从示范知识中获益。Qwen2-VL-7B [31] 表现出持续提升，单次示范从 71.8% 提升至 77.3%(+5.5%)，三次示范达到 79.4%(+7.6%)，证实了

the benefits of LearnAct generalize across models with different architectures and capabilities.

LearnAct 的优势可推广至不同架构和能力的模型。

The results also reveal interesting patterns regarding the impact of demonstration quantity. For Gemini-1.5-Pro [26], performance scales monotonically with the number of demonstrations, suggesting that less specialized foundation models can benefit substantially from additional examples. In contrast, UI-TARS-7B-SFT [22] achieves its peak performance with just one demonstration, indicating that models already fine-tuned for GUI tasks may efficiently extract necessary information from minimal demonstrations.

结果还揭示了示范数量对性能影响的有趣规律。对于 Gemini-1.5-Pro [26]，性能随示范数量单调增长，表明较少专门化的基础模型能从更多示范中显著受益。相比之下，UI-TARS-7B-SFT [22] 仅需一次示范即可达到峰值性能，说明已针对 GUI 任务微调的模型能高效从少量示范中提取必要信息。

Application-specific results highlight LearnAct's consistent improvement across diverse scenarios, with particularly notable gains in complex applications like CityMapper (from 14.1% to 69.4% for Gemini-1.5-Pro [26]) and To-Do apps (from 17.4% to 69.2%). This suggests that demonstration-based learning is especially valuable for navigating applications with complex interactions and nonstandard interfaces.

应用场景的结果突出显示了 LearnAct 在多样化场景中的持续提升，尤其是在复杂应用如 CityMapper(Gemini-1.5-Pro [26] 从 14.1% 提升至 69.4%) 和待办事项应用 (从 17.4% 提升至 69.2%) 中表现尤为显著。这表明基于示范的学习对于处理复杂交互和非标准界面的应用尤为重要。

To further understand the factors influencing LearnAct's effectiveness, we analyzed performance across different similarity profiles, as shown in Table 5. Several important insights emerge: Gemini-1.5-Pro [26] shows substantial improvements across all similarity combinations, with the largest gains in action match accuracy (ranging from +29.3% to +39.6%). This indicates that demonstration knowledge significantly enhances the model's ability to execute precise actions regardless of similarity conditions. UI-TARS-7BSFT [22] exhibits the most pronounced improvements in $\mathrm{UI_{SH}Act_{SH}}$ scenarios (+13.9% with 3-shot), suggesting that the model can extract maximum value from demonstrations when both UI and action patterns are similar to the target task. Qwen2-VL-7B [31] shows notably large improvements in action type accuracy for 2-shot settings (e.g., +67.4% for $\mathrm{UI_{SH}Act_{SH}}$), potentially indicating a threshold effect where multiple demonstrations trigger significant pattern recognition improvements.

为进一步理解影响 LearnAct 效果的因素，我们分析了不同相似性配置下的性能，如表 5 所示。几个重要见解浮现:Gemini-1.5-Pro [26] 在所有相似性组合中均表现出显著提升，动作匹配准确率提升最大 (范围为 +29.3% 至 +39.6%)，表明示范知识显著增强了模型在各种相似性条件下执行精确动作的能力。UI-TARS-7B-SFT [22] 在 $\mathrm{UI_{SH}Act_{SH}}$ 场景下表现出最明显的提升 (3 次示范提升 +13.9%)，说明当 UI 和动作模式均与目标任务相似时，模型能最大化利用示范信息。Qwen2-VL-7B [31] 在 2 次示范设置下动作类型准确率提升显著 (例如 $\mathrm{UI_{SH}Act_{SH}}$ 提升 +67.4%)，可能表明存在阈值效应，多次示范触发了显著的模式识别提升。

Interestingly, while UI similarity generally correlates with higher performance gains, we observe that action similarity also plays a

有趣的是，尽管 UI 相似性通常与更高的性能提升相关，我们也观察到动作相似性同样起到了

Table 5: Performance breakdown of LearnAct-Offline on different UI and action combinations. Performance metrics (type and match accuracy) across four similarity quadrants showing absolute values and relative improvements [in brackets] compared to baselines. Results are grouped by model and number of support examples (1/2/3-shot).

表 5:LearnAct-Offline 在不同 UI 和动作组合上的性能细分。性能指标 (类型准确率和匹配准确率) 在四个相似性象限中的绝对值及相对提升 [括号内]，相较基线。结果按模型及支持示范数量 (1/2/3 次示范) 分组。

Models	Supports	UISHActSH		UISHActsL		UISLACTSH		UISLACTSL	
		type	match	type	match	type	match	type	match
gemini-1.5-pro	1-shot	79.5 [+12.8]	50.2 [+35.6]	78.1 [+12.3]	47.8 [+33.2]	77.5 [+9.2]	52.3 [+30.5]	77.9 [+14.1]	44.2 [+29.3]
	2-shot	77.7 [+13.0]	53.9 [+37.3]	73.2 [+10.8]	49.9 [+34.7]	80.0 [+9.0]	56.5 [+34.8]	77.2 [+12.9]	48.9 [+34.4]
	3-shot	72.3 [+15.8]	53.5 [+39.6]	72.8 [+12.9]	49.5 [+34.6]	78.7 [+10.4]	60.0 [+38.4]	79.2 [+12.8]	51.6 [+36.3]
Qwen2-VL-7B	1-shot	86.0 [+5.3]	72.2 [+6.3]	85.4 [+4.9]	69.6 [+5.5]	86.0 [+2.0]	76.2 [+5.4]	82.9 [+1.3]	69.4 [+4.3]
	2-shot	85.0 [+67.4]	75.6 [+9.3]	84.0 [+67.2]	71.2 [+5.7]	86.9 [+73.3]	76.8 [+6.3]	84.0 [+68.5]	70.5 [+5.5]
	3-shot	80.2 [+5.0]	70.3 [+7.9]	82.9 [+4.7]	70.2 [+5.7]	85.6 [+1.9]	77.5 [+8.4]	85.6 [+3.4]	72.8 [+6.6]
UI-TARS-7B-SFT	1-shot	88.1 [+1.9]	77.8 [+6.6]	87.2 [+2.1]	75.3 [+6.4]	87.7 [+0.3]	80.1 [+5.9]	85.0 [-0.2]	75.0 [+2.8]
	2-shot	85.5 [+2.1]	76.7 [+8.3]	85.7 [+1.6]	75.9 [+4.9]	87.3 [-0.4]	79.1 [+5.9]	84.9 [-0.8]	74.1 [+2.1]
	3-shot	87.1 [+7.9]	78.2 [+13.9]	85.5 [+2.6]	75.4 [+4.9]	86.0 [-0.9]	78.9 [+6.8]	85.5 [-0.9]	75.2 [+2.7]

模型	支持	UISHActSH		UISHActsL		UISLACTSH		UISLACTSL	
		类型	匹配	类型	匹配	类型	匹配	类型	匹配
gemini-1.5-pro	一例示范	79.5 [+12.8]	50.2 [+35.6]	78.1 [+12.3]	47.8 [+33.2]	77.5 [+9.2]	52.3 [+30.5]	77.9 [+14.1]	44.2 [+29.3]
	二例示范	77.7 [+13.0]	53.9 [+37.3]	73.2 [+10.8]	49.9 [+34.7]	80.0 [+9.0]	56.5 [+34.8]	77.2 [+12.9]	48.9 [+34.4]
	三例示范	72.3 [+15.8]	53.5 [+39.6]	72.8 [+12.9]	49.5 [+34.6]	78.7 [+10.4]	60.0 [+38.4]	79.2 [+12.8]	51.6 [+36.3]
Qwen2-VL-7B	一例示范	86.0 [+5.3]	72.2 [+6.3]	85.4 [+4.9]	69.6 [+5.5]	86.0 [+2.0]	76.2 [+5.4]	82.9 [+1.3]	69.4 [+4.3]
	二例示范	85.0 [+67.4]	75.6 [+9.3]	84.0 [+67.2]	71.2 [+5.7]	86.9 [+73.3]	76.8 [+6.3]	84.0 [+68.5]	70.5 [+5.5]
	三例示范	80.2 [+5.0]	70.3 [+7.9]	82.9 [+4.7]	70.2 [+5.7]	85.6 [+1.9]	77.5 [+8.4]	85.6 [+3.4]	72.8 [+6.6]
UI-TARS-7B-SFT	一例示范	88.1 [+1.9]	77.8 [+6.6]	87.2 [+2.1]	75.3 [+6.4]	87.7 [+0.3]	80.1 [+5.9]	85.0 [-0.2]	75.0 [+2.8]
	二例示范	85.5 [+2.1]	76.7 [+8.3]	85.7 [+1.6]	75.9 [+4.9]	87.3 [-0.4]	79.1 [+5.9]	84.9 [-0.8]	74.1 [+2.1]
	三例示范	87.1 [+7.9]	78.2 [+13.9]	85.5 [+2.6]	75.4 [+4.9]	86.0 [-0.9]	78.9 [+6.8]	85.5 [-0.9]	75.2 [+2.7]

Table 6: Performance comparison of different models on the LearnGUI-Online benchmark. Comparison of models with different inputs (Image, Image+AXTree) and parameters, measuring task success rate (LearnGUI-Online $_{SR}$) with improvements shown in brackets for models with LearnAct enhancement.

表 6: 不同模型在 LearnGUI-Online 基准测试上的性能比较。比较了不同输入 (图像，图像 +AXTree) 和参数的模型，测量任务成功率 (LearnGUI-Online $_{SR}$)，括号中显示了采用 LearnAct 增强模型的提升。

Input	Models	#Params	LearnGUI-OnlinesR
Image + AXTree	GPT-40[12]	-	34.5
Image + AXTree	Gemini-Pro-1.5[26]	-	22.8
Image	Claude Computer-Use[2]	-	27.9
Image	Aguvis[41]	72B	26.1
Image	Owen2-VL-7B + 0-shot	7B	9.9
Image	Owen2-VL-7B + LearnAct	7B	21.1 [+11.2]
Image	UI-TARS-7B-SFT + 0-shot	7B	18.1
Image	UI-TARS-7B-SFT + LearnAct	7B	32.8 [+14.7]

输入	模型	参数数量	LearnGUI-OnlinesR
图像 + AXTree	GPT-40[12]	-	34.5
图像 + AXTree	Gemini-Pro-1.5[26]	-	22.8
图像	Claude Computer-Use[2]	-	27.9
图像	Aguvis[41]	72B	26.1
图像	Owen2-VL-7B + 零样本	7B	9.9
图像	Owen2-VL-7B + LearnAct	7B	21.1 [+11.2]
图像	UI-TARS-7B-SFT + 零样本	7B	18.1
图像	UI-TARS-7B-SFT + LearnAct	7B	32.8 [+14.7]

crucial role. For instance, Gemini-1.5-Pro [26] achieves its highest match accuracy in $UI_{SL}Act_{SH}$ scenarios (+38.4% with 3-shot), suggesting that action similarity can sometimes compensate for UI differences. This finding highlights the importance of considering both UI and action similarity when designing demonstration-based learning approaches for mobile GUI agents.

关键作用。例如，Gemini-1.5-Pro [26] 在 $UI_{SL}Act_{SH}$ 场景中实现了最高的匹配准确率 (3-shot 提升 38.4%)，表明动作相似性有时可以弥补界面差异。该发现强调了在设计基于示范的移动 GUI 代理学习方法时，同时考虑界面和动作相似性的重要性。

These results validate our hypothesized framework design, demonstrating that LearnAct successfully leverages demonstration similarity to enhance performance across varying conditions, with the most substantial benefits observed when demonstrations can provide both perceptual and procedural knowledge relevant to the target task.

这些结果验证了我们假设的框架设计，表明 LearnAct 成功利用示范相似性提升了不同条件下的性能，且当示范能够提供与目标任务相关的感知和过程知识时，收益最为显著。

5.2.2 Online Agent Capability Evaluation. While offline evaluations provide valuable insights into step-by-step execution capabilities, real-world deployment requires successful end-to-end task completion. Table 6 presents the results of our online evaluation on the LearnGUI-Online benchmark, which reveals several important findings. The LearnAct framework substantially improves performance for both evaluated models, with Qwen2-VL-7B [31] improving from 9.9% to 21.1% (+11.2%) and UI-TARS-7B-SFT [22] from 18.1% to 32.8% (+14.7%). These significant gains demonstrate that the benefits of demonstration-based learning translate effectively to real-world interactive scenarios. Qwen2-VL-7B [31] with LearnAct achieves 21.1% success rate, showing meaningful improvements over its baseline performance. This suggests that the quality and relevance of demonstrations are highly effective for enhancing model capabilities. UI-TARS-7B-SFT [22] with LearnAct achieves 32.8% success rate, approaching the performance of GPT- 4o (34.5%) despite using a much smaller model. This indicates that demonstration-based learning can help bridge the gap between smaller specialized models and large foundation models. Detailed visualizations of these performance comparisons are provided in Appendix C.1.To provide concrete examples of how LearnAct performs in real-world scenarios, we present three detailed case studies in Appendix C.2.

5.2.2 在线代理能力评估。虽然离线评估提供了逐步执行能力的宝贵见解，但实际部署需要成功完成端到端任务。表 6 展示了我们在 LearnGUI-Online 基准上的在线评估结果，揭示了若干重要发现。LearnAct 框架显著提升了两个评估模型的性能，Qwen2-VL-7B [31] 从 9.9% 提升至 21.1%(+11.2%)，UI-TARS-7B-SFT [22] 从 18.1% 提升至 32.8%(+14.7%)。这些显著提升表明基于示范的学习效果有效转化为真实交互场景。Qwen2-VL-7B [31] 结合 LearnAct 实现 21.1% 的成功率，较基线表现有明显改进，表明示范的质量和相关性对增强模型能力极为有效。UI-TARS-7B-SFT [22] 结合 LearnAct 实现 32.8% 的成功率，接近 GPT-4o(34.5%) 的表现，尽管使用了更小的模型，说明基于示范的学习有助于缩小小型专用模型与大型基础模型之间的差距。详细的性能对比可见附录 C.1。为具体展示 LearnAct 在真实场景中的表现，附录 C.2 提供了三个详细案例研究。

The most striking finding is the effectiveness of our demonstration-based learning approach. The LearnAct framework provides significant performance improvements through its demonstration mechanism, with gains of up to 14.7% in task success rate. This demonstrates the power of high-quality demonstrations for enhancing model performance, highlighting the importance of relevant examples over simply increasing model size.

最显著的发现是我们基于示范的学习方法的有效性。LearnAct 框架通过示范机制显著提升性能，任务成功率最高提升 14.7%。这证明了高质量示范在提升模型性能中的强大作用，强调了相关示例的重要性，而非单纯扩大模型规模。

These results confirm that the LearnAct framework provides a practical pathway to developing effective mobile GUI agents, making it particularly valuable for application-specific customization and personalization scenarios.

这些结果确认 LearnAct 框架为开发高效移动 GUI 代理提供了切实可行的路径，特别适用于应用特定的定制化和个性化场景。

5.3 Ablation Study

5.3 消融研究

To understand the contribution of each component in the LearnAct framework, we conducted ablation experiments on the LearnGUI-Offline dataset using Gemini-1.5-Pro [26]. As shown in Table 7, we systematically evaluated the impact of removing either the DemoParser or KnowSeeker component while keeping all other settings constant.

为理解 LearnAct 框架中各组件的贡献，我们在 LearnGUI-Offline 数据集上使用 Gemini-1.5-Pro [26] 进行了消融实验。如表 7 所示，我们系统评估了在保持其他设置不变的情况下，移除 DemoParser 或 KnowSeeker 组件的影响。

The results reveal several important insights. Both components are essential, as removing either component leads to substantial performance degradation compared to the full LearnAct framework. The complete framework achieves 51.7% accuracy, while removing DemoParser reduces performance to 40.6% (-11.1%) and removing

结果揭示了若干重要见解。两个组件均不可或缺，移除任一组件均导致性能较完整 LearnAct 框架显著下降。完整框架达到 51.7% 的准确率，移除 DemoParser 后性能降至 40.6%(-11.1%)，移除

Table 7: Ablation study of LearnAct components. Performance comparison across four configurations: baseline (no components), DemoParser only, KnowSeeker only, and both components combined. Results are presented as overall average accuracy and per-application breakdown across nine applications.

表 7:LearnAct 组件的消融研究。四种配置下的性能比较: 基线 (无组件)、仅 DemoParser、仅 KnowSeeker 及两组件结合。结果以整体平均准确率及九个应用的分项表现呈现。

DemoParser	Ablation Setting									
KnowSeeker	Average	Gmail	Booking	Music	SHEIN	NBC	CityMapper	ToDo	Signal	Yelp
Baseline		19.3	20.1	16.4	24.5	10.2	35.6	14.1	17.4	27.9	15.2
	✓	40.6	47.7	31.3	55.4	29.1	47.0	43.0	58.2	48.8	50.7
✓		41.6	46.9	34.1	52.7	27.9	51.9	45.3	51.4	61.1	51.8
✓	✓	51.7	55.5	47.1	60.0	35.7	56.4	54.7	60.6	63.1	54.6

演示解析器	消融设置										
KnowSeeker	平均值	Gmail	Booking	音乐	SHEIN	NBC	CityMapper	待办事项	Signal	Yelp	
基线		19.3	20.1	16.4	24.5	10.2	35.6	14.1	17.4	27.9	15.2
	✓	40.6	47.7	31.3	55.4	29.1	47.0	43.0	58.2	48.8	50.7
✓		41.6	46.9	34.1	52.7	27.9	51.9	45.3	51.4	61.1	51.8
✓	✓	51.7	55.5	47.1	60.0	35.7	56.4	54.7	60.6	63.1	54.6

KnowSeeker reduces it to 41.6% (-10.1%). Regarding DemoParser's contribution, comparing "KnowSeeker only" (40.6%) to the baseline (19.3%), we observe that even without action descriptions, relevant demonstrations improve performance by 21.3%. However, the addition of DemoParser's action descriptions further enhances performance by 11.1%, confirming the value of structured knowledge extraction. For KnowSeeker's contribution, the "DemoParser only" configuration (41.6%) also substantially outperforms the baseline, indicating that detailed action descriptions are valuable even with randomly selected demonstrations. However, KnowSeeker's retrieval of relevant demonstrations provides an additional 10.1% improvement, highlighting the importance of demonstration relevance.

KnowSeeker 将其降低到 41.6%(-10.1%)。关于 DemoParser 的贡献,通过比较"仅 KnowSeeker"(40.6%) 与基线 (19.3%)，我们观察到即使没有动作描述，相关示范也能提升 21.3% 的性能。然而，加入 DemoParser 的动作描述进一步提升了 11.1% 的性能，证实了结构化知识提取的价值。对于 KnowSeeker 的贡献，"仅 DemoParser" 配置 (41.6%) 也显著优于基线，表明即使示范随机选择，详细的动作描述依然有价值。但 KnowSeeker 检索相关示范带来了额外 10.1% 的提升，凸显了示范相关性的重要性。

The performance variations across applications are particularly informative. For instance, in the Signal application, DemoParser appears more important (61.1% vs. 48.8% for KnowSeeker only), suggesting that detailed action descriptions are crucial for applications with complex interaction patterns. Conversely, for the ToDo application, KnowSeeker seems more valuable (58.2% vs. 51.4% for DemoParser only), indicating that demonstration relevance may be more critical for applications with varied task types.

各应用中的性能差异尤具启示性。例如，在 Signal 应用中，DemoParser 显得更为重要 (61.1% 对比仅 KnowSeeker 的 48.8%)，表明复杂交互模式的应用中详细动作描述至关重要。相反，在 ToDo 应用中，KnowSeeker 更具价值 (58.2% 对比仅 DemoParser 的 51.4%)，说明示范相关性对任务类型多样的应用更为关键。

These findings validate our multi-agent framework design, confirming that both knowledge extraction (DemoParser) and relevant demonstration retrieval (KnowSeeker) play complementary and essential roles in enabling effective demonstration-based learning for mobile GUI agents.

这些发现验证了我们的多智能体框架设计，确认知识提取 (DemoParser) 和相关示范检索 (KnowSeeker) 在实现基于示范的移动 GUI 代理有效学习中发挥着互补且关键的作用。

6 DISCUSSION AND FUTURE WORK

6 讨论与未来工作

Our experimental results demonstrate that demonstration-based learning significantly enhances mobile GUI agents' capabilities. The substantial performance improvements across all evaluated models validate our core hypothesis that demonstration-based learning effectively addresses generalization challenges. Even advanced foundation models like Gemini-1.5-Pro [26] show dramatic improvements (198.9% relative improvement). Our multi-dimensional similarity analysis reveals that both UI similarity and action similarity influence learning efficacy, with action similarity sometimes compensating for UI differences.

我们的实验结果表明，基于示范的学习显著提升了移动 GUI 代理的能力。所有评估模型的显著性能提升验证了我们的核心假设: 基于示范的学习有效解决了泛化难题。即使是先进的基础模型如 Gemini-1.5-Pro [26] 也表现出显著提升 (相对提升 198.9%)。我们的多维相似性分析显示，UI 相似性和动作相似性均影响学习效果，且动作相似性有时可弥补 UI 差异。

Data Collection and Dataset Expansion. While our approach shows promising results, several limitations and future directions warrant consideration. First, regarding data collection, our current dataset, while comprehensive, could benefit from greater diversity and representativeness. The LearnGUI dataset, comprising 2,252 offline tasks and 101 online tasks, represents a significant step forward but remains limited in scale compared to the vast diversity of mobile applications and user interactions. Future work should expand the dataset to include a broader range of applications, particularly those with complex interaction patterns and specialized domains.

数据收集与数据集扩展。尽管我们的方法展现出良好效果, 但仍存在若干限制和未来方向值得关注。首先，在数据收集方面，当前数据集虽较全面，但在多样性和代表性上仍有提升空间。LearnGUI 数据集包含 2252 个离线任务和 101 个在线任务，虽是重要进展，但与移动应用和用户交互的广泛多样性相比，规模仍有限。未来工作应扩展数据集，涵盖更多应用，尤其是复杂交互模式和专业领域的应用。

K-shot Learning Analysis. Second, our current investigation of k-shot learning is limited to $k = 1, 2$, and 3 demonstrations. While these configurations provide valuable insights, a more comprehensive analysis of how demonstration quantity affects performance would be beneficial. Future research could explore the relationship between the number of demonstrations and performance gains, potentially identifying optimal demonstration counts for different scenarios and model architectures.

K-shot 学习分析。其次，我们当前对 k-shot 学习的研究仅限于 $k = 1, 2$ 和 3 个示范。虽然这些配置提供了有价值的见解，但更全面地分析示范数量对性能的影响将更有益。未来研究可探讨示范数量与性能提升的关系，可能为不同场景和模型架构确定最佳示范数量。

Enhanced Learning and Execution Strategies. Third, our learning and execution strategies could be enhanced to better leverage the relationship between support tasks and query tasks. While our current approach effectively retrieves relevant demonstrations, more sophisticated methods could be developed to extract and transfer knowledge more efficiently. For instance, techniques for abstracting common patterns across demonstrations, identifying critical decision points, and adapting demonstrated strategies to novel scenarios could further improve performance.

增强的学习与执行策略。第三，我们的学习和执行策略可进一步优化，以更好地利用支持任务与查询任务之间的关系。当前方法虽能有效检索相关示范，但可开发更复杂的方法以更高效地提取和迁移知识。例如，抽象示范中的共性模式、识别关键决策点及将示范策略适应新场景的技术，均有望提升性能。

Agent Self-Learning. A promising direction for future research is to enable agents to learn from their own successful executions. Currently, our framework relies exclusively on human demonstrations, but agents could potentially learn from their own successful task completions. By incorporating these successful agent executions into the knowledge base, we could enable a form of ”self-learning” where agents continuously improve their capabilities through their own experiences.

代理自我学习。未来研究的一个有前景方向是使代理能够从自身成功执行中学习。目前，我们的框架完全依赖于人工示范，但代理有潜力从自身成功完成的任务中学习。通过将这些成功执行纳入知识库，可实现一种“自我学习”模式，使代理通过自身经验持续提升能力。

By addressing these limitations and pursuing these research directions, demonstration-based learning can evolve into a robust paradigm for developing adaptable, personalized, and practically deployable mobile GUI agents that effectively address the diverse needs of real-world users. The insights gained from our multidimensional similarity analysis provide valuable guidance for future research in this domain, suggesting that both UI similarity and action similarity play crucial roles in successful knowledge transfer.

通过解决这些限制并推进上述研究方向，基于示范的学习有望发展为一种强健的范式，助力开发适应性强、个性化且实用的移动 GUI 代理，有效满足现实用户的多样化需求。我们多维相似性分析所得的洞见为该领域未来研究提供了宝贵指导，表明 UI 相似性和动作相似性在成功知识迁移中均扮演关键角色。

7 CONCLUSION

7 结论

This paper introduces a novel demonstration-based learning paradigm that fundamentally addresses the generalization challenges

本文提出了一种新颖的基于示范的学习范式，根本性地解决了泛化挑战

faced by mobile GUI agents. Rather than pursuing universal coverage through ever-larger datasets, our approach leverages human demonstrations to enhance agent performance in unseen scenarios. We developed LearnGUI, the first comprehensive dataset for studying demonstration-based learning in mobile GUI agents, comprising 2,252 offline tasks and 101 online tasks with high-quality human demonstrations. We further designed LearnAct, a sophisticated multi-agent framework with three specialized components: DemoParser for knowledge extraction, KnowSeeker for relevant knowledge retrieval, and ActExecutor for demonstration-enhanced task execution. Our experimental results demonstrate remarkable performance gains, with a single demonstration increasing Gemini-1.5-Pro [26]'s accuracy from 19.3% to 51.7% in offline tests and enhancing UI-TARS-7B-SFT [22]'s online task success rate from 18.1% to 32.8%. These findings establish demonstration-based learning as a promising direction for developing more adaptable, personalized, and practically deployable mobile GUI agents.

移动 GUI 代理面临的挑战。我们的方法不是通过不断增大数据集来追求普适覆盖，而是利用人类示范来提升代理在未见场景中的表现。我们开发了 LearnGUI，这是首个用于研究基于示范学习的移动 GUI 代理的综合数据集，包含 2252 个离线任务和 101 个带有高质量人类示范的在线任务。我们进一步设计了 LearnAct，一个复杂的多代理框架，包含三个专门组件: 用于知识提取的 DemoParser、用于相关知识检索的 KnowSeeker，以及用于示范增强任务执行的 ActExecutor。实验结果显示显著性能提升，单次示范使 Gemini-1.5-Pro [26] 在离线测试中的准确率从 19.3% 提升至 51.7%，并将 UI-TARS-7B-SFT [22] 的在线任务成功率从 18.1% 提升至 32.8%。这些发现确立了基于示范学习作为开发更具适应性、个性化和实用性的移动 GUI 代理的有前景方向。

REFERENCES

参考文献

[1] Simone Agostinelli, Andrea Marrella, and Massimo Mecella. 2019. Research challenges for intelligent robotic process automation. In Business Process Management Workshops: BPM 2019 International Workshops, Vienna, Austria, September 1-6, 2019, Revised Selected Papers 17. Springer, 12-18.

Simone Agostinelli, Andrea Marrella, 和 Massimo Mecella. 2019. 智能机器人流程自动化的研究挑战。载于《业务流程管理研讨会:BPM 2019 国际研讨会，奥地利维也纳，2019 年 9 月 1-6 日，修订精选论文集》第 17 卷。施普林格，12-18 页。

[2] Anthropic. 2024. Developing a computer use model. https://www.anthropic.com/news/developing-computer-use

Anthropic. 2024. 开发计算机使用模型。https://www.anthropic.com/news/developing-computer-use

[3] Chongyang Bai, Xiaoxue Zang, Ying Xu, Srinivas Sunkara, Abhinav Rastogi, Jindong Chen, et al. 2021. Uibert: Learning generic multimodal representations for ui understanding. arXiv preprint arXiv:2107.13731 (2021).

Chongyang Bai, Xiaoxue Zang, Ying Xu, Srinivas Sunkara, Abhinav Rastogi, Jindong Chen 等. 2021. Uibert: 学习通用多模态表示以理解用户界面。arXiv 预印本 arXiv:2107.13731 (2021)。

[4] Andrea Burns, Deniz Arsan, Sanjna Agrawal, Ranjitha Kumar, Kate Saenko, and Bryan A Plummer. 2021. Mobile app tasks with iterative feedback (motif): Addressing task feasibility in interactive visual environments. arXiv preprint arXiv:2104.08560 (2021).

Andrea Burns, Deniz Arsan, Sanjna Agrawal, Ranjitha Kumar, Kate Saenko, 和 Bryan A Plummer. 2021. 带迭代反馈的移动应用任务 (motif): 解决交互式视觉环境中的任务可行性问题。arXiv 预印本 arXiv:2104.08560 (2021)。

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

Yuxiang Chai, Siyuan Huang, Yazhe Niu, Han Xiao, Liang Liu, Dingyu Zhang, Peng Gao, Shuai Ren, 和 Hongsheng Li. 2024. Amex: 面向移动 GUI 代理的 Android 多注释展示数据集。arXiv 预印本 arXiv:2407.17490 (2024)。

[6] Yuxiang Chai, Hanhao Li, Jiayu Zhang, Liang Liu, Guozhi Wang, Shuai Ren, Siyuan Huang, and Hongsheng Li. 2025. A3: Android Agent Arena for Mobile GUI Agents. arXiv preprint arXiv:2501.01149 (2025).

Yuxiang Chai, Hanhao Li, Jiayu Zhang, Liang Liu, Guozhi Wang, Shuai Ren, Siyuan Huang, 和 Hongsheng Li. 2025. A3: 面向移动 GUI 代理的 Android 代理竞技场。arXiv 预印本 arXiv:2501.01149 (2025)。

[7] Wentong Chen, Junbo Cui, Jinyi Hu, Yujia Qin, Junjie Fang, Yue Zhao, Chongyi Wang, Jun Liu, Guirong Chen, Yupeng Huo, et al. 2024. GUICourse: From General Vision Language Models to Versatile GUI Agents. arXiv preprint arXiv:2406.11317 (2024).

Wentong Chen, Junbo Cui, Jinyi Hu, Yujia Qin, Junjie Fang, Yue Zhao, Chongyi Wang, Jun Liu, Guirong Chen, Yupeng Huo 等. 2024. GUICourse: 从通用视觉语言模型到多功能 GUI 代理。arXiv 预印本 arXiv:2406.11317 (2024)。

[8] Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. 2022. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. arXiv preprint arXiv:2211.12588 (2022).

Wenhu Chen, Xueguang Ma, Xinyi Wang, 和 William W Cohen. 2022. 思维程序提示: 将计算与推理分离以解决数值推理任务。arXiv 预印本 arXiv:2211.12588 (2022)。

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

Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Yantao Li, Jianbing Zhang, 和 Zhiyong Wu. 2024. Seeclick: 利用 GUI 定位实现高级视觉 GUI 代理。arXiv 预印本 arXiv:2401.10935 (2024)。

[10] Tiago Guerreiro, Ricardo Gamboa, and Joaquim Jorge. 2008. Mnemonical body shortcuts: improving mobile interaction. In Proceedings of the 15th European conference on Cognitive ergonomics: the ergonomics of cool interaction. 1-8.

Tiago Guerreiro, Ricardo Gamboa, 和 Joaquim Jorge. 2008. 记忆体快捷方式: 提升移动交互体验。载于第 15 届欧洲认知工效学会议论文集: 酷交互的人体工学。1-8 页。

[11] Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxiao Dong, Ming Ding, 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.

Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxiao Dong, Ming Ding 等. 2024. Cogagent: 面向 GUI 代理的视觉语言模型。载于 IEEE/CVF 计算机视觉与模式识别会议论文集。14281-14290 页。

[12] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow,

Akila Welihinda, Alan Hayes, Alec Radford, et al. 2024. Gpt-4o system card. arXiv preprint arXiv:2410.21276 (2024).

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, 等. 2024. Gpt-4o 系统卡. arXiv 预印本 arXiv:2410.21276 (2024).

[13] Courtney Kennedy and Stephen E Everett. 2011. Use of cognitive shortcuts in landline and cell phone surveys. Public Opinion Quarterly 75, 2 (2011), 336-348.

Courtney Kennedy 和 Stephen E Everett. 2011. 固定电话和手机调查中认知捷径的使用. 《公众舆论季刊》75 卷第 2 期 (2011), 336-348.

[14] Wei Li, William Bishop, Alice Li, Chris Rawles, Folawiyo Campbell-Ajala, Divya Tyamagundlu, and Oriana Riva. 2024. On the Effects of Data Scale on Computer Control Agents. arXiv preprint arXiv:2406.03679 (2024).

Wei Li, William Bishop, Alice Li, Chris Rawles, Folawiyo Campbell-Ajala, Divya Tyamagundlu, 和 Oriana Riva. 2024. 数据规模对计算机控制代理影响的研究. arXiv 预印本 arXiv:2406.03679 (2024).

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

Yang Li, Jiacong He, Xin Zhou, Yuan Zhang, 和 Jason Baldridge. 2020. 将自然语言指令映射到移动 UI 操作序列. arXiv 预印本 arXiv:2005.03776 (2020).

[16] Kevin Qinghong Lin, Linjie Li, Difei Gao, Zhengyuan Yang, Shiwei Wu, Zechen Bai, Weixian Lei, Lijuan Wang, and Mike Zheng Shou. 2024. Showui: One vision-language-action model for gui visual agent. arXiv preprint arXiv:2411.17465 (2024).

Kevin Qinghong Lin, Linjie Li, Difei Gao, Zhengyuan Yang, Shiwei Wu, Zechen Bai, Weixian Lei, Lijuan Wang, 和 Mike Zheng Shou. 2024. Showui: 一种用于 GUI 视觉代理的视觉-语言-动作统一模型. arXiv 预印本 arXiv:2411.17465 (2024).

[17] William Liu, Liang Liu, Yaxuan Guo, Han Xiao, Weifeng Lin, Yuxiang Chai, Shuai Ren, Xiaoyu Liang, Linghao Li, Wenhao Wang, et al. 2025. Llm-powered gui agents in phone automation: Surveying progress and prospects. (2025).

William Liu, Liang Liu, Yaxuan Guo, Han Xiao, Weifeng Lin, Yuxiang Chai, Shuai Ren, Xiaoyu Liang, Linghao Li, Wenhao Wang, 等. 2025. 基于大语言模型的手机自动化 GUI 代理: 进展与前景综述. (2025).

[18] Zhe Liu, Cheng Li, Chunyang Chen, Junjie Wang, Boyu Wu, Yawen Wang, Jun Hu, and Qing Wang. 2024. Vision-driven Automated Mobile GUI Testing via Multimodal Large Language Model. arXiv preprint arXiv:2407.03037 (2024).

Zhe Liu, Cheng Li, Chunyang Chen, Junjie Wang, Boyu Wu, Yawen Wang, Jun Hu, 和 Qing Wang. 2024. 基于视觉驱动的多模态大语言模型移动 GUI 自动化测试. arXiv 预印本 arXiv:2407.03037 (2024).

[19] Quanfeng Lu, Wenqi Shao, Zitao Liu, Fanqing Meng, Boxuan Li, Botong Chen, Siyuan Huang, Kaipeng Zhang, Yu Qiao, and Ping Luo. 2024. GUI Odyssey: A Comprehensive Dataset for Cross-App GUI Navigation on Mobile Devices. arXiv preprint arXiv:2406.08451 (2024).

Quanfeng Lu, Wenqi Shao, Zitao Liu, Fanqing Meng, Boxuan Li, Botong Chen, Siyuan Huang, Kaipeng Zhang, Yu Qiao, 和 Ping Luo. 2024. GUI Odyssey: 面向移动设备跨应用 GUI 导航的综合数据集. arXiv 预印本 arXiv:2406.08451 (2024).

[20] Yadong Lu, Jianwei Yang, Yelong Shen, and Ahmed Awadallah. 2024. Omniparser for pure vision based gui agent. arXiv preprint arXiv:2408.00203 (2024).

Yadong Lu, Jianwei Yang, Yelong Shen, 和 Ahmed Awadallah. 2024. Omniparser: 基于纯视觉的 GUI 代理. arXiv 预印本 arXiv:2408.00203 (2024).

[21] Pawel Pawlowski, Krystian Zawistowski, Wojciech Lapacz, Marcin Sko-rupa, Adam Wiacek, Sebastien Postansque, and Jakub Hoscilowicz. 2024. TinyClick: Single-Turn Agent for Empowering GUI Automation. arXiv preprint arXiv:2410.11871 (2024).

Pawel Pawlowski, Krystian Zawistowski, Wojciech Lapacz, Marcin Sko-rupa, Adam Wiacek, Sebastien Postansque, 和 Jakub Hoscilowicz. 2024. TinyClick: 用于增强 GUI 自动化的单轮代理. arXiv 预印本 arXiv:2410.11871 (2024).

[22] Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, et al. 2025. UI-TARS: Pioneering Automated GUI Interaction with Native Agents. arXiv preprint arXiv:2501.12326 (2025).

Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, 等. 2025. UI-TARS: 开创性的原生代理自动化 GUI 交互. arXiv 预印本 arXiv:2501.12326 (2025).

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

Christopher Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyo Campbell-Ajala, 等. 2024. AndroidWorld: 面向自主代理的动态基准环境. arXiv 预印本 arXiv:2405.14573 (2024).

[24] Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy Lilli-crap. 2024. Androidinthewild: A large-scale dataset for android device control. Advances in Neural Information Processing Systems 36 (2024).

Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, 和 Timothy Lillicrap. 2024. Androidinthewild: 大规模安卓设备控制数据集. 《神经信息处理系统进展》36 卷 (2024).

[25] Yunpeng Song, Yiheng Bian, Yongtao Tang, and Zhongmin Cai. 2023. Navigating Interfaces with AI for Enhanced User Interaction. arXiv preprint arXiv:2312.11190 (2023).

Yunpeng Song, Yiheng Bian, Yongtao Tang, 和 Zhongmin Cai. 2023. 利用 AI 导航界面以增强用户交互. arXiv 预印本 arXiv:2312.11190 (2023).

[26] Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, et al. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. arXiv preprint arXiv:2403.05530 (2024).

Gemini 团队，Petko Georgiev，Ving Ian Lei，Ryan Burnell，Libin Bai，Anmol Gulati，Garrett Tanzer，Damien Vincent，Zhufeng Pan，Shibo Wang 等。2024。Gemini 1.5: 解锁跨越数百万上下文标记的多模态理解。arXiv 预印本 arXiv:2403.05530 (2024)。

[27] Sagar Gubbi Venkatesh, Partha Talukdar, and Srini Narayanan. 2022. Ugif: Ui grounded instruction following. arXiv preprint arXiv:2211.07615 (2022).

Sagar Gubbi Venkatesh，Partha Talukdar，和 Srini Narayanan。2022。Ugif: 基于用户界面 (UI) 的指令跟随。arXiv 预印本 arXiv:2211.07615 (2022)。

[28] Junyang Wang, Haiyang Xu, Haitao Jia, Xi Zhang, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang, and Jitao Sang. 2024. Mobile-Agent-v2: Mobile Device Operation Assistant with Effective Navigation via Multi-Agent Collaboration. arXiv preprint arXiv:2406.01014 (2024).

Junyang Wang，Haiyang Xu，Haitao Jia，Xi Zhang，Ming Yan，Weizhou Shen，Ji Zhang，Fei Huang，和 Jitao Sang。2024。Mobile-Agent-v2: 通过多代理协作实现高效导航的移动设备操作助手。arXiv 预印本 arXiv:2406.01014 (2024)。

[29] Junyang Wang, Haiyang Xu, Jiabo Ye, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang, and Jitao Sang. 2024. Mobile-agent: Autonomous multi-modal mobile device agent with visual perception. arXiv preprint arXiv:2401.16158 (2024).

Junyang Wang, Haiyang Xu, Jiabo Ye, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang, 和 Jitao Sang。2024。Mobile-agent: 具备视觉感知的自主多模态移动设备代理。arXiv 预印本 arXiv:2401.16158 (2024)。

[30] Luyuan Wang, Yongyu Deng, Yiwei Zha, Guodong Mao, Qinmin Wang, Tianchen Min, Wei Chen, and Shoufa Chen. 2024. MobileAgentBench: An Efficient and User-Friendly Benchmark for Mobile LLM Agents. arXiv preprint arXiv:2406.08184 (2024).

Luyuan Wang，Yongyu Deng，Yiwei Zha，Guodong Mao，Qinmin Wang，Tianchen Min，Wei Chen，和 Shoufa Chen。2024。MobileAgentBench: 一个高效且用户友好的移动大型语言模型 (LLM) 代理基准测试。arXiv 预印本 arXiv:2406.08184 (2024)。

[31] 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. 2024. Qwen2-VL: Enhancing Vision-Language Model's Perception of the World at Any Resolution. arXiv preprint arXiv:2409.12191 (2024).

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，和 Junyang Lin。2024。Qwen2-VL: 提升视觉-语言模型 (Vision-Language Model) 对任意分辨率世界的感知能力。arXiv 预印本 arXiv:2409.12191 (2024)。

[32] Shuai Wang, Weiwen Liu, Jingxuan Chen, Weinan Gan, Xingshan Zeng, Shuai Yu, Xinlong Hao, Kun Shao, Yasheng Wang, and Ruiming Tang. 2024. GUI Agents with Foundation Models: A Comprehensive Survey. arXiv preprint arXiv:2411.04890 (2024).

Shuai Wang，Weiwen Liu，Jingxuan Chen，Weinan Gan，Xingshan Zeng，Shuai Yu，Xinlong Hao，Kun Shao，Yasheng Wang，和 Ruiming Tang。2024。基于基础模型的图形用户界面 (GUI) 代理: 全面综述。arXiv 预印本 arXiv:2411.04890 (2024)。

[33] Wenhao Wang, Zijie Yu, William Liu, Rui Ye, Tian Jin, Siheng Chen, and Yanfeng Wang. 2025. FedMobileAgent: Training Mobile Agents Using Decentralized Self-Sourced Data from Diverse Users. arXiv preprint arXiv:2502.02982 (2025).

Wenhao Wang，Zijie Yu，William Liu，Rui Ye，Tian Jin，Siheng Chen，和 Yanfeng Wang。2025。FedMobileAgent: 利用来自多样用户的去中心化自源数据训练移动代理。arXiv 预印本 arXiv:2502.02982 (2025)。

[34] Zhenhailong Wang, Haiyang Xu, Junyang Wang, Xi Zhang, Ming Yan, Ji Zhang, Fei Huang, and Heng Ji. 2025. Mobile-Agent-E: Self-Evolving Mobile Assistant for Complex Tasks. arXiv preprint arXiv:2501.11733 (2025).

Zhenhailong Wang，Haiyang Xu，Junyang Wang，Xi Zhang，Ming Yan，Ji Zhang，Fei Huang，和 Heng Ji。2025。Mobile-Agent-E: 面向复杂任务的自我进化移动助手。arXiv 预印本 arXiv:2501.11733 (2025)。

[35] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. Advances in neural information processing systems 35 (2022), 24824-24837.

Jason Wei，Xuezhi Wang，Dale Schuurmans，Maarten Bosma，Fei Xia，Ed Chi，Quoc V Le，Denny Zhou 等。2022。链式思维提示 (Chain-of-thought prompting) 激发大型语言模型的推理能力。神经信息处理系统进展 35 (2022)，24824-24837。

[36] Hao Wen, Yuanchun Li, Guohong Liu, Shanhui Zhao, Tao Yu, Toby Jia-Jun Li, Shiqi Jiang, Yunhao Liu, Yaqin Zhang, and Yunxin Liu. 2024. Autodroid: Llm-powered task automation in android. In Proceedings of the 30th Annual International Conference on Mobile Computing and Networking. 543-557.

Hao Wen，Yuanchun Li，Guohong Liu，Shanhui Zhao，Tao Yu，Toby Jia-Jun Li，Shiqi Jiang，Yunhao Liu，Yaqin Zhang，和 Yunxin Liu。2024。Autodroid: 基于大型语言模型 (LLM) 的安卓任务自动化。发表于第 30 届国际移动计算与网络会议论文集。543-557。

[37] Hao Wen, Hongming Wang, Jiaxuan Liu, and Yuanchun Li. 2023. Droidbot-gpt: Gpt-powered ui automation for android. arXiv preprint arXiv:2304.07061 (2023).

Hao Wen，Hongming Wang，Jiaxuan Liu，和 Yuanchun Li。2023。Droidbot-gpt: 基于 GPT 的安卓用户界面自动化。arXiv 预印本 arXiv:2304.07061 (2023)。

[38] Biao Wu, Yanda Li, Meng Fang, Zirui Song, Zhiwei Zhang, Yunchao Wei, and Ling Chen. 2024. Foundations and recent trends in multimodal mobile agents: A survey. arXiv preprint arXiv:2411.02006 (2024).

Biao Wu，Yanda Li，Meng Fang，Zirui Song，Zhiwei Zhang，Yunchao Wei，和 Ling Chen。2024。多模态移动代理的基础与最新趋势综述。arXiv 预印本 arXiv:2411.02006 (2024)。

[39] Zhiyong Wu, Zhenyu Wu, Fangzhi Xu, Yian Wang, Qiushi Sun, Chengyou Jia, Kanzhi Cheng, Zichen Ding, Liheng Chen, Paul Pu Liang, et al. 2024. Os-atlas: A foundation action model for generalist gui agents. arXiv preprint arXiv:2410.23218 (2024).

吴志勇, 吴振宇, 徐方志, 王奕安, 孙秋实, 贾成友, 程坎之, 丁子辰, 陈立恒, Paul Pu Liang 等. 2024. Os-atlas: 通用 GUI 代理的基础动作模型. arXiv 预印本 arXiv:2410.23218 (2024).

[40] Yifan Xu, Xiao Liu, Xueqiao Sun, Siyi Cheng, Hao Yu, Hanyu Lai, Shudan Zhang, Dan Zhang, Jie Tang, and Yuxiao Dong. 2024. AndroidLab: Training and Systematic Benchmarking of Android Autonomous Agents. arXiv preprint arXiv:2410.24024 (2024).

徐一凡, 刘晓, 孙雪桥, 程思怡, 余浩, 赖涵宇, 张书丹, 张丹, 唐杰, 董宇霄. 2024. AndroidLab: 安卓自主代理的训练与系统性基准测试. arXiv 预印本 arXiv:2410.24024 (2024).

[41] Yiheng Xu, Zekun Wang, Junli Wang, Dunjie Lu, Tianbao Xie, Amrita Saha, Doyen Sahoo, Tao Yu, and Caiming Xiong. 2024. Aguvis: Unified Pure Vision Agents for Autonomous GUI Interaction. arXiv preprint arXiv:2412.04454 (2024).

徐一恒, 王泽坤, 王俊立, 卢敦杰, 谢天宝, Amrita Saha, Doyen Sahoo, 余涛, 熊才明. 2024. Aguvis: 用于自主 GUI 交互的统一纯视觉代理. arXiv 预印本 arXiv:2412.04454 (2024).

[42] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2024. Tree of thoughts: Deliberate problem solving with large language models. Advances in Neural Information Processing Systems 36 (2024).

姚顺宇, 余典, 赵杰弗里, Izhak Shafran, Tom Griffiths, 曹源, Karthik Narasimhan. 2024. 思维树: 利用大型语言模型进行深思熟虑的问题解决. 神经信息处理系统进展 36 (2024).

[43] Chaoyun Zhang, Shilin He, Jiaxu Qian, Bowen Li, Liqun Li, Si Qin, Yu Kang, Minghua Ma, Qingwei Lin, Saravan Rajmohan, et al. 2024. Large Language Model-Brained GUI Agents: A Survey. arXiv preprint

arXiv:2411.18279 (2024).

张超云, 何世林, 钱嘉旭, 李博文, 李立群, 秦思, 康宇, 马明华, 林庆伟, Saravan Rajmohan 等. 2024. 大型语言模型驱动的 GUI 代理: 综述. arXiv 预印本 arXiv:2411.18279 (2024).

[44] Chi Zhang, Zhao Yang, Jiaxuan Liu, Yucheng Han, Xin Chen, Zebiao Huang, Bin Fu, and Gang Yu. 2023. Appagent: Multimodal agents as smartphone users. arXiv preprint arXiv:2312.13771 (2023).

张驰, 杨钊, 刘佳轩, 韩宇成, 陈昕, 黄泽彪, 傅斌, 余刚. 2023. Appagent: 作为智能手机用户的多模态代理. arXiv 预印本 arXiv:2312.13771 (2023).

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

张继文, 吴继豪, 滕一华, 廖明辉, 徐诺, 肖晓, 魏中宇, 唐杜宇. 2024. Android in the zoo: GUI 代理的动作-思维链. arXiv 预印本 arXiv:2403.02713 (2024).

[46] Jiayi Zhang, Chuang Zhao, Yihan Zhao, Zhaoyang Yu, Ming He, and Jianping Fan. 2024. MobileExperts: A Dynamic Tool-Enabled Agent Team in Mobile Devices. arXiv preprint arXiv:2407.03913 (2024).

张佳怡, 赵闯, 赵一涵, 余昭阳, 何明, 范建平. 2024. MobileExperts: 移动设备中的动态工具驱动代理团队. arXiv 预印本 arXiv:2407.03913 (2024).

[47] Li Zhang, Shihe Wang, Xianqing Jia, Zhihan Zheng, Yunhe Yan, Longxi Gao, Yuanchun Li, and Mengwei Xu. 2024. LlamaTouch: A Faithful and Scalable Testbed for Mobile UI Automation Task Evaluation. arXiv preprint arXiv:2404.16054 (2024).

张力, 王世和, 贾贤清, 郑志涵, 颜云鹤, 高龙曦, 李元春, 徐孟威. 2024. LlamaTouch: 一个真实且可扩展的移动 UI 自动化任务评测平台. arXiv 预印本 arXiv:2404.16054 (2024).

[48] Zhuosheng Zhang and Aston Zhang. 2023. You only look at screens: Multimodal chain-of-action agents. arXiv preprint arXiv:2309.11436 (2023).

张卓晟, Aston Zhang. 2023. 你只需看屏幕: 多模态动作链代理. arXiv 预印本 arXiv:2309.11436 (2023).

A ADDITIONAL LEARNGUI STATISTICS

A 附加的 LearnGUI 统计数据

Figure 8 illustrates the distribution of similarity scores across different dimensions in the LearnGUI-Offline dataset, enabling systematic analysis of how different types of similarity between demonstration and query tasks affect learning efficacy.

图 8 展示了 LearnGUI-Offline 数据集中不同维度的相似度分布，便于系统分析示范任务与查询任务之间不同类型相似度对学习效果的影响。

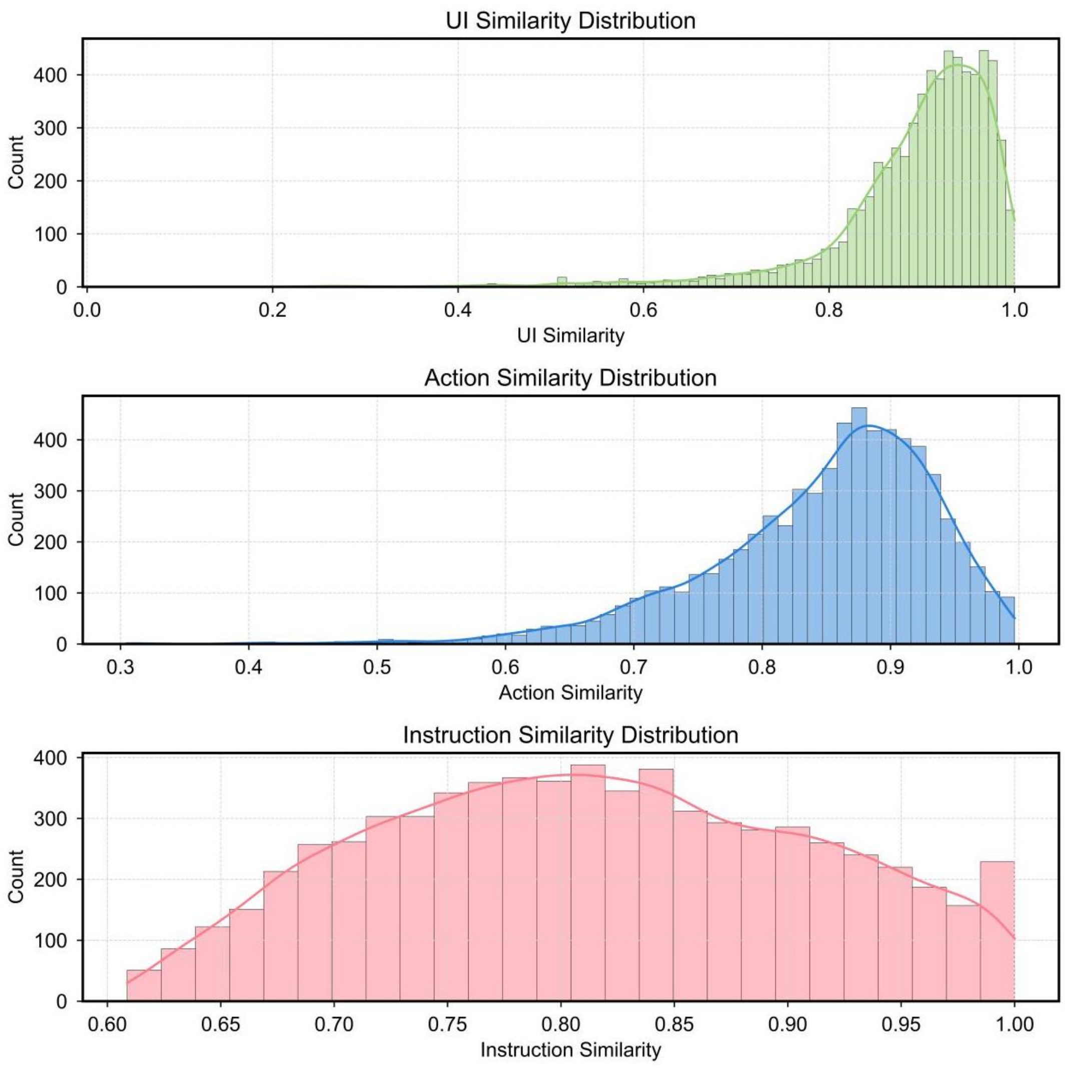


Figure 8: Distribution of instruction, UI, and action similarity scores in LearnGUI-Offline. The histograms show the distribution of similarity scores across three dimensions: instruction similarity (top), UI similarity (middle), and action similarity (bottom). These distributions enable systematic analysis of how different types of similarity between demonstration and query tasks affect learning efficacy.

图 8:LearnGUI-Offline 中指令、UI 和动作相似度分数的分布。直方图显示了三种维度的相似度分布: 指令相似度 (上)、UI 相似度 (中) 和动作相似度 (下)。这些分布支持系统分析示范任务与查询任务之间不同类型相似度对学习效果的影响。

B LEARNACT FRAMEWORK DETAILS

B LearnAct 框架细节

This section provides detailed descriptions of the components of our LearnAct framework, corresponding to the methods presented in Section 4 of the paper.

本节详细介绍了 LearnAct 框架的各个组成部分，对应论文第 4 节中提出的方法。

B.1 DemoParser Prompts

B.1 DemoParser 提示

We provide all of our prompt templates used in DemoParser for generating semantically descriptive action descriptions from demonstration data. These carefully designed prompts guide the vision-language model to produce structured knowledge that captures the essence of human demonstrations, as shown in Figures 9 and 10.

我们提供了 DemoParser 中用于从演示数据生成语义描述性动作描述的所有提示模板。这些精心设计的提示引导视觉语言模型生成结构化知识，捕捉人类演示的本质，如图 9 和图 10 所示。

Prompt 1: Intermediate Action Description

提示 1: 中间动作描述

System Prompt:

系统提示:

You are a mobile UI interaction analyst. Follow these rules: 1. Analyze the split-screen image (Before-action left, After-action right) 2. For click actions, a high-contrast red marker (white-bordered circle) shows the precise click location, with a green square surrounding it and a 'C' label at the top-right corner of the square indicating the click. 3. Output JSON with ONLY ONE 'action_description' field in this exact format: "[On/In] [Screen Name], [Action Details], to [Purpose]"

你是一名移动端 UI 交互分析师。遵循以下规则:1. 分析分屏图像 (左侧为动作前，右侧为动作后)2. 对于点击动作，使用高对比度红色标记 (白边圆圈) 显示精确点击位置，绿色方框环绕，方框右上角标有“C”标签表示点击。3. 输出 JSON，且仅包含一个“action_description”字段，格式严格为: “[在/于] [屏幕名称]，[动作细节]，以 [目的]”

Action Types: - click [element] (e.g., 'Search button') - swipe [up/down/left/right] - type [text] in [field] - press [back/home/enter]

动作类型:- 点击 [元素](例如，“搜索按钮”)- 滑动 [上/下/左/右] - 在 [字段] 中输入 [文本] - 按下 [返回/主页/回车]

Validation Rules: 1. Screen names should be 2-6 words 2. Keep purpose descriptions under 8 words 3. Never mention coordinates/IDs

验证规则:1. 屏幕名称应为 2-6 个词 2. 目的描述不超过 8 个词 3. 不得提及坐标/ID

MEMORY RECORDING RULES: If the current screen contains information relevant to the user's instruction that needs to be remembered for future steps, include a Memory part in your action description. The format should be: ”[On/In] [Screen Name], [Action Details], to [Purpose]. [Memory: important information for future steps]”

记忆记录规则: 如果当前屏幕包含与用户指令相关且需记忆以供后续步骤使用的信息，在动作描述中加入记忆部分。格式为: “[在/于] [屏幕名称]，[动作细节]，以 [目的]。[记忆: 后续步骤的重要信息]”

Memory should ONLY be added when: 1. The information is relevant to completing the user's instruction 2. The information will likely be needed in future steps 3. This specific information has NOT been recorded in previous action history entries

仅在以下情况下添加记忆:1. 信息与完成用户指令相关 2. 信息可能在后续步骤需要 3. 该信息未在之前的动作历史中记录过

Memory examples: 1. For a travel planning task: On Travel Blog, click 'Bali Beach Guide', to read article. [Memory: Guide mentions Kuta Beach has surfing lessons for $25/hour] 2. For a shopping task: In Product Details, click 'Add to Cart', to select item. [Memory: iPhone 13 Pro costs $999 with 128GB storage] 3. For a note-taking task: On Weather App, swipe down forecast, to view weekend. [Memory: Saturday will be rainy with 80% precipitation]

记忆示例:1. 旅行规划任务: 在旅游博客，点击“巴厘岛海滩指南”，以阅读文章。[记忆: 指南提到库塔海滩有每小时 25 美元的冲浪课程] 2. 购物任务: 在产品详情，点击“加入购物车”，以选择商品。[记忆:iPhone 13 Pro 售价 999 美元，存储 128GB] 3. 记笔记任务: 在天气应用，向下滑动查看预报，以查看周末天气。[记忆: 周六有 80% 降水概率，预计下雨]

Avoid using Memory for: 1. Obvious UI changes that don't contain task-relevant information 2. Information already captured in previous action steps 3. Generic observations not specific to the user's task objective

避免为以下情况使用记忆:1. 明显的 UI 变化且不含任务相关信息 2. 已在之前动作步骤中捕获的信息 3. 与用户任务目标无关的泛泛观察

Figure 9: Prompt template for intermediate action descriptions. The template guides DemoParser to generate standardized descriptions for intermediate actions, including detailed rules for memory annotations that capture important information observed during task execution.

图 9: 中间动作描述的提示模板。该模板指导 DemoParser 生成标准化的中间动作描述，包括捕捉任务执行中重要信息的记忆注释详细规则。

B.2 ActExecutor Prompts

B.2 ActExecutor 提示

We provide the prompt templates used by ActExecutor to make decisions based on current observations, action history, and demonstration knowledge. These prompts guide the vision-language model to select appropriate actions for task execution, as shown in Figure 11.

我们提供 ActExecutor 用于基于当前观察、动作历史和演示知识做出决策的提示模板。这些提示引导视觉语言模型选择适当动作以执行任务，如图 11 所示。

B.3 Algorithm Details

B.3 算法细节

We provide the detailed algorithms for the DemoParser and ActExecutor components of our LearnAct framework, which are the core computational processes enabling knowledge extraction and task execution.

我们提供 LearnAct 框架中 DemoParser 和 ActExecutor 组件的详细算法，这些是实现知识提取和任务执行的核心计算过程。

C ADDITIONAL EXPERIMENTAL RESULTS AND ANALYSES

C 额外实验结果与分析

This section provides additional experimental results and analyses that supplement the findings presented in Section 5 of the paper.

本节提供了补充论文第 5 节中研究结果的额外实验结果和分析。

C.1 Online Performance Comparisons

C.1 在线性能比较

Figures 12 and 13 provide detailed comparisons of model performance with and without LearnAct enhancement in online evaluation scenarios.

图 12 和图 13 详细比较了在在线评估场景中，模型有无 LearnAct 增强时的性能表现。

C.2 Case Studies of LearnAct Online Experiments

C.2 LearnAct 在线实验案例研究

We present three detailed case studies from our online experiments to provide concrete examples of how LearnAct leverages demonstration knowledge to solve tasks in unseen mobile applications. These case studies highlight different scenarios where demonstration knowledge proves particularly beneficial for task execution.

我们展示了三个来自在线实验的详细案例研究，具体说明 LearnAct 如何利用示范知识解决未见过的移动应用中的任务。这些案例突出了示范知识在任务执行中尤为有益的不同场景。

Prompt 2: Terminal Action Description - Standard Completion

提示 2: 终端动作描述 - 标准完成

System Prompt for standard completion:

标准完成的系统提示:

Determine the final task status. Output rules: 1. Use ONLY ONE ’action_description’ field 2. Format: ”[On/In] [Screen], complete task, [Reason]”

确定最终任务状态。输出规则:1. 仅使用一个’action_description’ 字段 2. 格式: “[在/于] [屏幕], 完成任务, [原因]”

Validation Rules: - Reason should be less than 10 words - Screen name must match previous context

验证规则:- 原因应少于 10 个词 - 屏幕名称必须与前文上下文匹配

Examples: 1. Basic completion: On Payment Screen, complete task, successfully submit order 2. Failure case: In Search Results, cannot complete task, no nearby Vivo mobile phone stores found

示例:1. 基本完成: 在支付屏幕，完成任务，成功提交订单 2. 失败案例: 于搜索结果，无法完成任务，未找到附近的 Vivo 手机店

Prompt 3: Terminal Action Description - With Answer

提示 3: 终端动作描述 - 带答案

System Prompt for completion with answer:

带答案完成的系统提示:

Determine the final task status with the given answer. Output rules: 1. Use ONLY ONE ’action_description’ field 2. Format: ”[On/In] [Screen], complete task, the answer is [answer]”

根据给定答案确定最终任务状态。输出规则:1. 仅使用一个’action_description’ 字段 2. 格式: “[在] [屏幕], 完成任务, 答案是 [answer]”

Validation Rules: - Screen name must match previous context - Use the exact answer provided in the TASK_COMPLETE action

验证规则:- 屏幕名称必须与之前的上下文匹配 - 使用 TASK_COMPLETE 操作中提供的准确答案

Examples: 1. Answer is a price: On Checkout Screen, complete task, the answer is ”$299.9”. 2. Answer is a list: On Payment Options Screen, complete task, the answer is ”google pay, check out with affirm, add credit/debit card”.

示例:1. 答案是价格: 在结账屏幕，完成任务，答案是“$299.9”。2. 答案是列表: 在支付选项屏幕，完成任务，答案是“google pay，使用 affirm 结账，添加信用/借记卡”。

Figure 10: Prompt templates for terminal action descriptions. The templates provide specific formats for both standard task completion and information retrieval tasks, ensuring consistent output structure across different task types.

图 10: 终端操作描述的提示模板。模板为标准任务完成和信息检索任务提供了具体格式，确保不同任务类型输出结构一致。

Algorithm 1 DemoParser Knowledge Generation Process

算法 1 DemoParser 知识生成过程

Require: Demonstration dataset $D = \left\{(i_k, s_k, a_k)_{k=1}^{N}\right\}$ where i_k is instruction, s_k is screenshot sequence, a_k is action sequence

需求: 演示数据集 $D = \left\{(i_k, s_k, a_k)_{k=1}^{N}\right\}$，其中 i_k 为指令，s_k 为截图序列，a_k 为操作序列

Ensure: Knowledge base K with semantically descriptive action descriptions

保证: 知识库 K 包含语义描述的操作说明

$K \leftarrow \varnothing$ ▷ Initialize empty knowledge base

$K \leftarrow \varnothing$ ▷ 初始化空知识库

for each demonstration (i, s, a) in D do

对 D 中的每个演示 (i, s, a) 执行

$d \leftarrow \varnothing$ - Initialize empty description sequence

$d \leftarrow \varnothing$ - 初始化空描述序列

for $j = 1$ to $|a|$ do

从 $j = 1$ 到 $|a|$ 循环

if $j < |a|$ then ▷ Intermediate action

```
    如果 $j < |a|$ 则 $\triangleright$ 中间操作
      Create visualization of action $a_j$ with before-after screenshots from $s_j$ and $s_{j+1}$
      使用 $s_j$ 和 $s_{j+1}$ 的前后截图创建操作 $a_j$ 的可视化
      $h \leftarrow$ Previous action descriptions $\{d_1, d_2, \ldots, d_{j-1}\}$
      $h \leftarrow$ 之前的操作描述 $\{d_1, d_2, \ldots, d_{j-1}\}$
      $d_j \leftarrow$ GenerateDescription $(i, a_j,$ visualization, $h)$ using prompt format detailed in Appendix B. 1
      $d_j \leftarrow$ 使用附录 B.1 中详细的提示格式生成描述 $(i, a_j,$ visualization, $h)$
      $d_j$ follows format: ”[On/In] [Screen],[Action], to [Purpose]” with optional memory
      $d_j$ 格式为:“[在] [屏幕]，[操作]，以 [目的]”可选记忆
    else - Terminal action
    否则 - 终端操作
      $h \leftarrow$ Complete action history $\{d_1, d_2, \ldots, d_{|a|-1}\}$
      $h \leftarrow$ 完成操作历史 $\{d_1, d_2, \ldots, d_{|a|-1}\}$
      $d_{|a|} \leftarrow$ GenerateFinalDescription $(i, s_{|a|}, h, a_{|a|})$ using prompt detailed in Appendix B.1
      $d_{|a|} \leftarrow$ 使用附录 B.1 中详细提示生成最终描述 $(i, s_{|a|}, h, a_{|a|})$
      $d_{|a|}$ follows format: ”[On/In] [Screen], complete task, [Reason/Answer]”
      $d_{|a|}$ 格式如下:“[在] [屏幕]，完成任务，[原因/答案]”
    end if
    结束条件判断
    Add $d_j$ to description sequence $d$
    将 $d_j$ 添加到描述序列 $d$
  end for
  循环结束
  Add $(i, a, d)$ to knowledge base $K$
```

将 (i, a, d) 添加到知识库 K

end for

循环结束

return K

返回 K

Prompt 4: Task Execution Prompt

提示 4: 任务执行提示

Role Definition:

角色定义:

You are a smartphone assistant to help users complete tasks by interacting with apps. I will give you a screenshot of the current phone screen.

你是智能手机助手，通过与应用交互帮助用户完成任务。我将提供当前手机屏幕的截图。

Example Tasks: [Only when demonstrations are available]

示例任务:[仅当有演示时提供]

Example 1: [Demonstration instruction] Steps taken in this example: Step-1: [Action] [Action Description] Step-2: [Action] [Action Description] ...

示例 1:[演示说明] 本例中执行的步骤: 步骤 1:[操作] [操作描述] 步骤 2:[操作] [操作描述] ...

Background: This image is a phone screenshot. Its width is [width] pixels and its height is [height] pixels. The user's instruction is: [instruction]

背景: 此图像为手机截图。宽度为 [width] 像素，高度为 [height] 像素。用户指令为:[instruction]

History operations: [Only when action history is available]

历史操作:[仅当操作历史可用时]

Before reaching this page, some operations have been completed. You need to refer to the completed operations to decide the next operation. These operations are as follow: Step-1: [Action] [Action Description] Step-2: [Action] [Action Description] ...

在到达此页面之前，已完成一些操作。您需要参考已完成的操作来决定下一步操作。这些操作如下: 步骤-1:[操作] [操作描述] 步骤-2:[操作] [操作描述] ...

Response requirements: Now you need to combine all of the above to decide just one action on the current page. You must choose one of the actions below:

响应要求: 现在您需要结合以上所有内容，在当前页面仅决定一个操作。您必须从以下操作中选择一个:

”SWIPE[UP]”: Swipe the screen up. ”SWIPE[DOWN]”: Swipe the screen down. ”SWIPE[LEFT]”: Swipe the screen left. ”SWIPE[RIGHT]”: Swipe the screen right. ”CLICK[x, y]”: Click the screen at the coordinates (x, y) .x is the pixel from left to right and y is the pixel from top to bottom ”TYPE[text]”: Type the given text in the current input field. ”PRESS_BACK”: Press the back button. ”PRESS_HOME”: Press the home button. ”PRESS_ENTER”: Press the enter button. ”TASK_COMPLETE[answer]”: Mark the task as complete. If the instruction requires answering a question, provide the answer inside the brackets. If no answer is needed, use empty brackets ”TASK_COMPLETE[]”.

“SWIPE[UP]”: 向上滑动屏幕。“SWIPE[DOWN]”: 向下滑动屏幕。“SWIPE[LEFT]”: 向左滑动屏幕。“SWIPE[RIGHT]”: 向右滑动屏幕。“CLICK[x, y]”: 点击屏幕坐标，(x, y) .x 为从左到右的像素，y 为从上到下的像素。“TYPE[text]”: 在当前输入框中输入指定文本。“PRESS_BACK”: 按返回键。“PRESS_HOME”: 按主页键。“PRESS_ENTER”: 按回车键。“TASK_COMPLETE[answer]”: 标记任务完成。如果指令要求回答问题，请在括号内提供答案。如不需答案，使用空括号“TASK_COMPLETE[]”。

Response Example: Your output should be a string and nothing else, containing only the action type you choose from the list above. For example: ”SWIPE[UP]” ”CLICK[156, 2067]” ”TYPE[Rome]” ”PRESS_BACK” ”PRESS_HOME” ”PRESS_ENTER” ”TASK_COMPLETE[1h30m]” ”TASK_COMPLETE[]”

响应示例: 您的输出应为字符串且仅包含您从上述列表中选择的操作类型。例如: “SWIPE[UP]”“CLICK[156, 2067]”“TYPE[Rome]”“PRESS_BACK”“PRESS_HOME”“PRESS_ENTER”“TASK_COMPLET

Figure 11: Task execution prompt template. This comprehensive prompt directs ActExecutor to generate actions based on current observations, action history, and retrieved demonstrations, with explicit formatting requirements to ensure consistent action outputs.

图 11: 任务执行提示模板。该综合提示引导 ActExecutor 基于当前观察、操作历史和检索到的示范生成操作，具有明确的格式要求以确保操作输出一致。

Algorithm 2 ActExecutor Task Execution Process

算法 2 ActExecutor 任务执行流程

Require: User instruction i , Knowledge base K , Maximum steps T

```
输入: 用户指令 $i$ , 知识库 $K$ , 最大步骤数 $T$
 Ensure: Task execution trajectory
输出: 任务执行轨迹
    $t \leftarrow 0 \triangleright$ Initialize time step
  $t \leftarrow 0 \triangleright$ 初始化时间步
    $h \leftarrow \varnothing$ - Initialize action history
  $h \leftarrow \varnothing$ - 初始化操作历史
    $\mathcal{D} \leftarrow$ KnowSeeker $(i, K) \triangleright$ Retrieve relevant demonstrations
  $\mathcal{D} \leftarrow$ KnowSeeker $(i, K) \triangleright$ 检索相关示范
    while $t < T$ and not IsTaskComplete do
  while $t < T$ 且任务未完成 do
        $o_t \leftarrow$ GetObservation() - Obtain current screenshot
      $o_t \leftarrow$ GetObservation() - 获取当前截图
        $P_t \leftarrow$ ConstructPrompt $(i, o_t, h, \mathcal{D}) \triangleright$ Construct decision prompt
      $P_t \leftarrow$ ConstructPrompt $(i, o_t, h, \mathcal{D}) \triangleright$ 构建决策提示
        $a_t \leftarrow f_{LLM}(P_t)$ - Generate action via LLM
      $a_t \leftarrow f_{LLM}(P_t)$ - 通过大语言模型生成操作
        $d_t \leftarrow$ GenerateDescription $(i, a_t, o_t, h) \triangleright$ Generate action description
      $d_t \leftarrow$ 生成描述 $(i, a_t, o_t, h) \triangleright$ 生成动作描述
        $h \leftarrow h \cup \{(a_t, d_t)\} \triangleright$ Update action history
      $h \leftarrow h \cup \{(a_t, d_t)\} \triangleright$ 更新动作历史
        ExecuteAction $(a_t) \triangleright$ Execute action in environment
      执行动作 $(a_t) \triangleright$ 在环境中执行动作
        $t \leftarrow t + 1$ - Increment time step
        $t \leftarrow t + 1$ - 时间步进
    end while
    结束循环
    return $\{(a_0, d_0), (a_1, d_1), \ldots, (a_{t-1}, d_{t-1})\}$
    返回 $\{(a_0, d_0), (a_1, d_1), \ldots, (a_{t-1}, d_{t-1})\}$
```

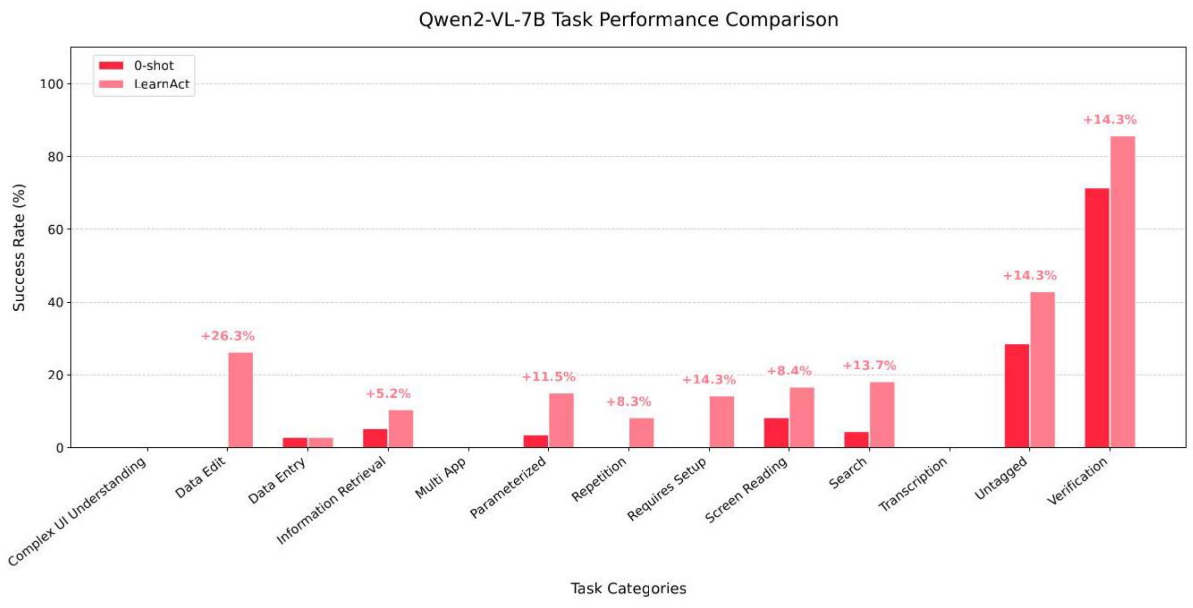


Figure 12: Detailed performance comparison of Qwen2-VL-7B with and without LearnAct on LearnGUI-Online. The figure shows the task success rates of Qwen2-VL-7B baseline versus Qwen2-VL-7B enhanced with LearnAct across different task dimensions in the LearnGUI-Online benchmark.

图 12:Qwen2-VL-7B 在 LearnGUI-Online 上使用和不使用 LearnAct 的详细性能比较。图中展示了 Qwen2-VL-7B 基线模型与增强了 LearnAct 的 Qwen2-VL-7B 在 LearnGUI-Online 基准测试中不同任务维度的任务成功率。

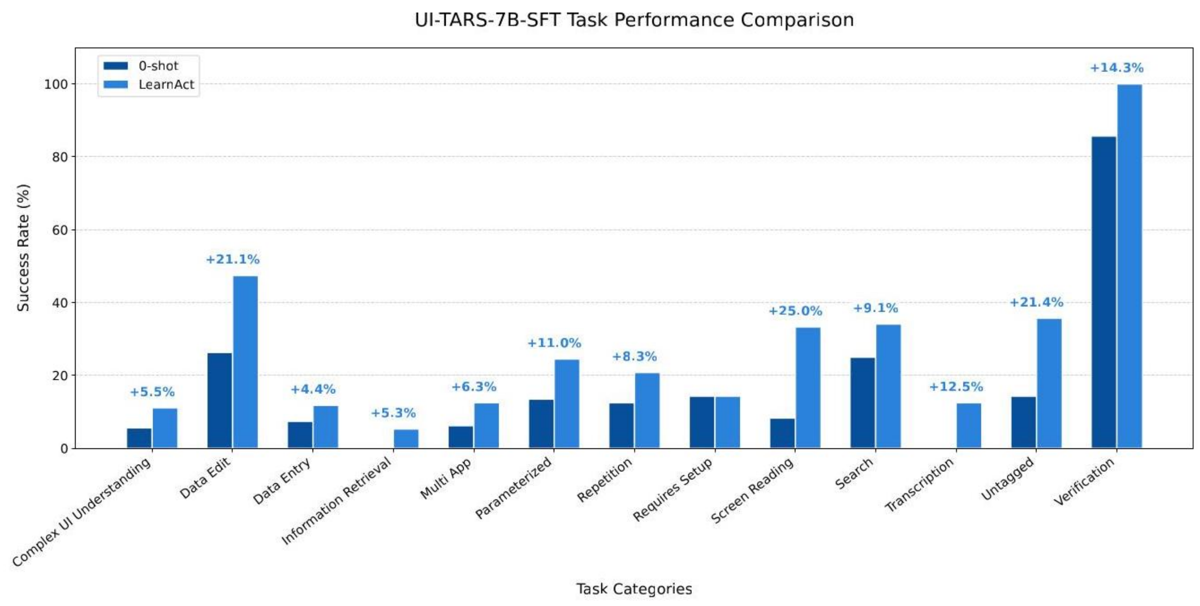


Figure 13: Detailed performance comparison of UI-TARS-7B-SFT with and without LearnAct on LearnGUI-Online. The figure presents a comprehensive breakdown of task success rates for UI-TARS-7B-SFT baseline versus UI-TARS-7B-SFT enhanced with LearnAct across multiple task dimensions in the LearnGUI-Online benchmark.

图 13:UI-TARS-7B-SFT 在 LearnGUI-Online 上使用和不使用 LearnAct 的详细性能比较。图中全面展示了 UI-TARS-7B-SFT 基线模型与增强了 LearnAct 的 UI-TARS-7B-SFT 在 LearnGUI-Online 基准测试中多个任务维度的任务成功率细分情况。

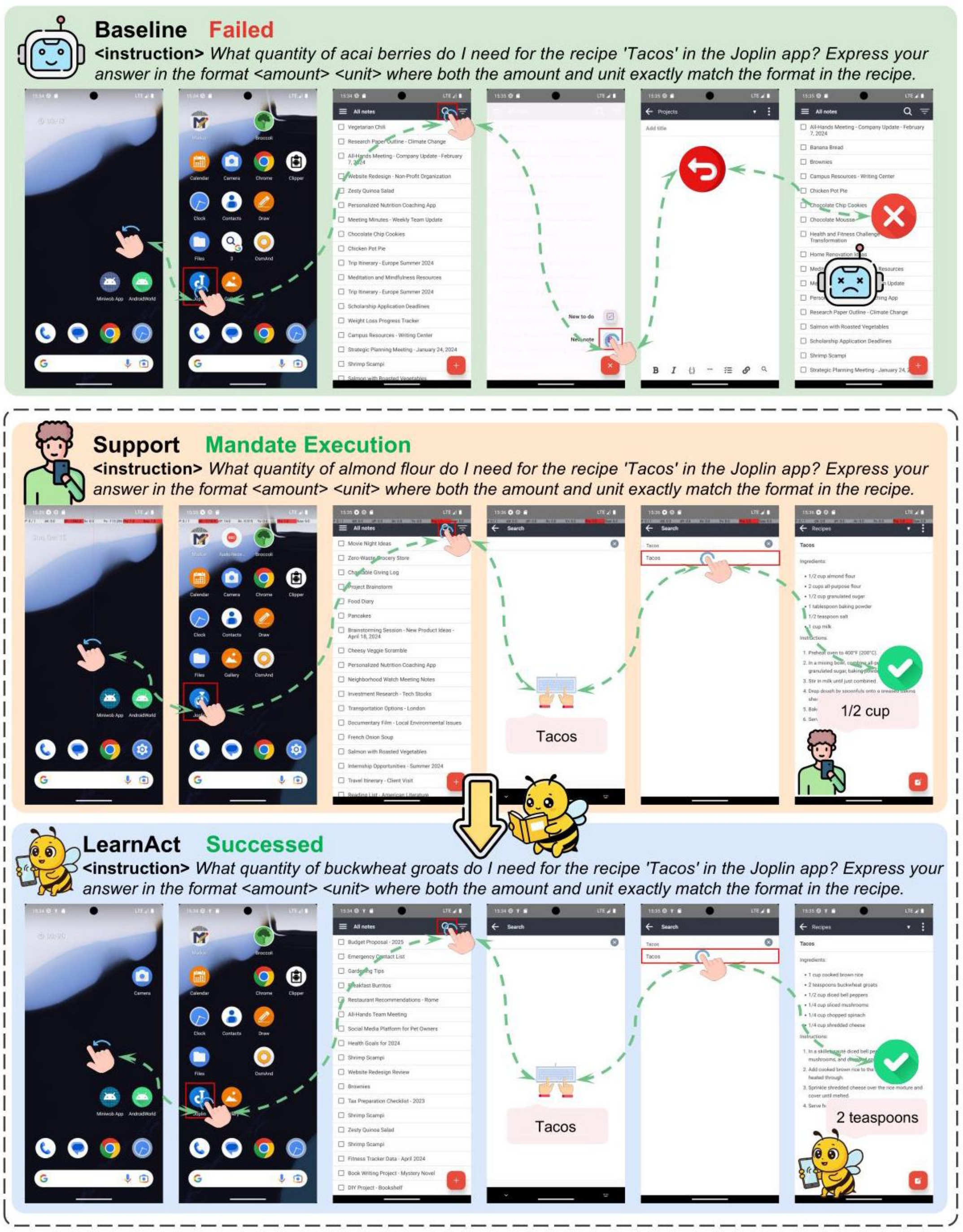

Baseline Failed
<instruction> What quantity of acai berries do I need for the recipe 'Tacos' in the Joplin app? Express your answer in the format <amount> <unit> where both the amount and unit exactly match the format in the recipe.
Support Mandate Execution
<instruction> What quantity of almond flour do I need for the recipe 'Tacos' in the Joplin app? Express your answer in the format <amount> <unit> where both the amount and unit exactly match the format in the recipe.
Tacos
1/2 cup
LearnAct Successed
<instruction> What quantity of buckwheat groats do I need for the recipe 'Tacos' in the Joplin app? Express your answer in the format <amount> <unit> where both the amount and unit exactly match the format in the recipe.
Tacos
2 teaspoons

Figure 14: UI-TARS-7B-SFT with LearnAct vs. Baseline in NotesRecipeIngredientCount Task. Task template: ”What quantity of {ingredient} do I need for the recipe ’{title}’ in the Joplin app? Express your answer in the format <amount> <unit> without using abbreviations.”

图 14:UI-TARS-7B-SFT 使用 LearnAct 与基线模型在 NotesRecipeIngredientCount 任务中的对比。任务模板:“在 Joplin 应用中,配方‘{title}’中 {ingredient} 的用量是多少?请以 <amount> <unit> 格式回答,单位不使用缩写。”

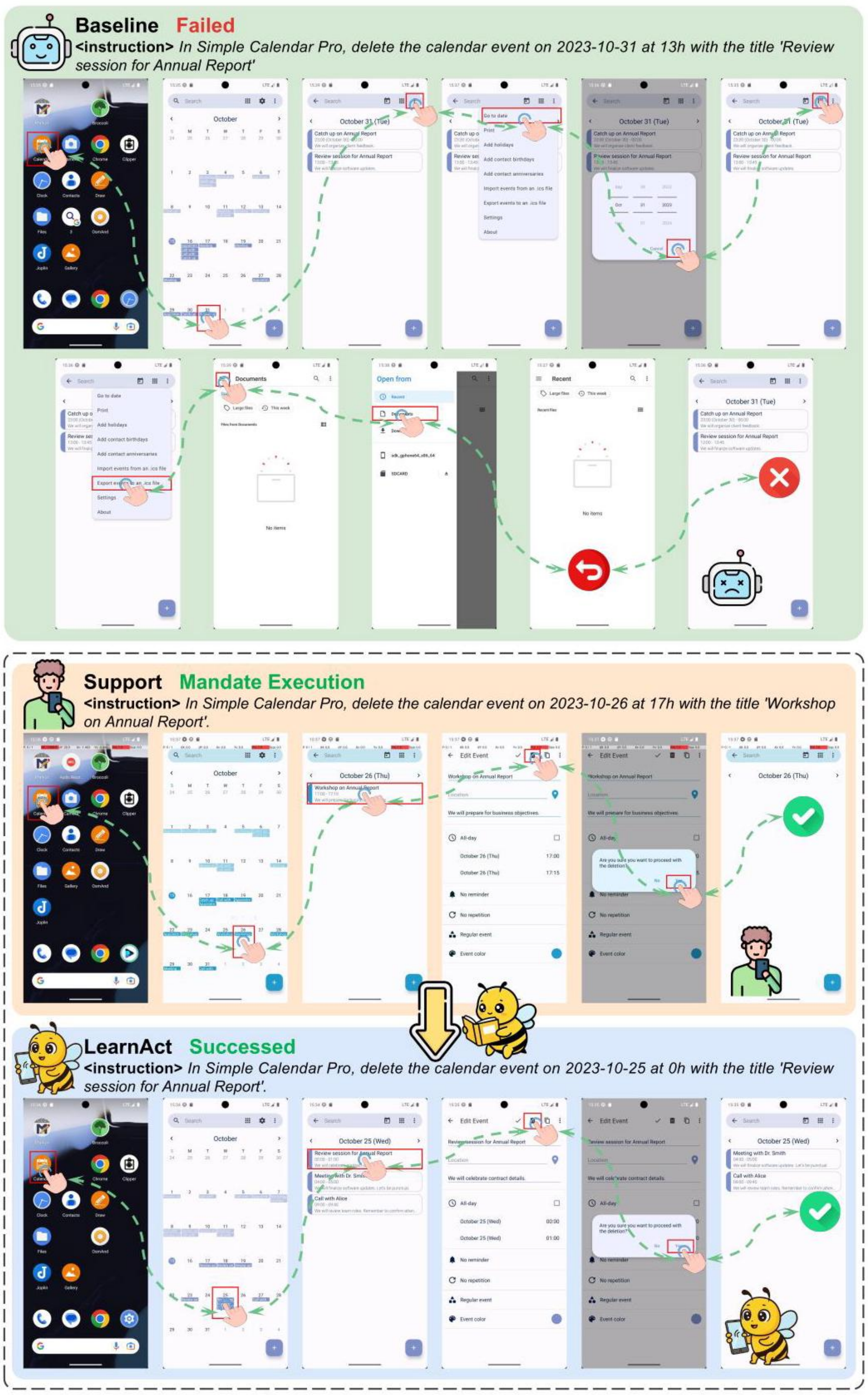

Baseline Failed
<instruction> In Simple Calendar Pro, delete the calendar event on 2023-10-31 at 13h with the title 'Review session for Annual Report'
Support Mandate Execution
<instruction> In Simple Calendar Pro, delete the calendar event on 2023-10-26 at 17h with the title 'Workshop on Annual Report'.
LearnAct Successed
<instruction> In Simple Calendar Pro, delete the calendar event on 2023-10-25 at 0h with the title 'Review session for Annual Report'.

Figure 15: UI-TARS-7B-SFT with LearnAct vs. Baseline in SimpleCalendarDeleteOneEvent Task. Task template: "In Simple Calendar Pro, delete the calendar event on {year}-{month}-{day} at {hour}h with the title '{event_title}'"

图 15:UI-TARS-7B-SFT 使用 LearnAct 与基线模型在 SimpleCalendarDeleteOneEvent 任务中的对比。任务模板: “在 Simple Calendar Pro 中，删除 {year} 年 {month} 月 {day} 日 {hour} 时，标题为 ‘{event_title}’ 的日历事件。”

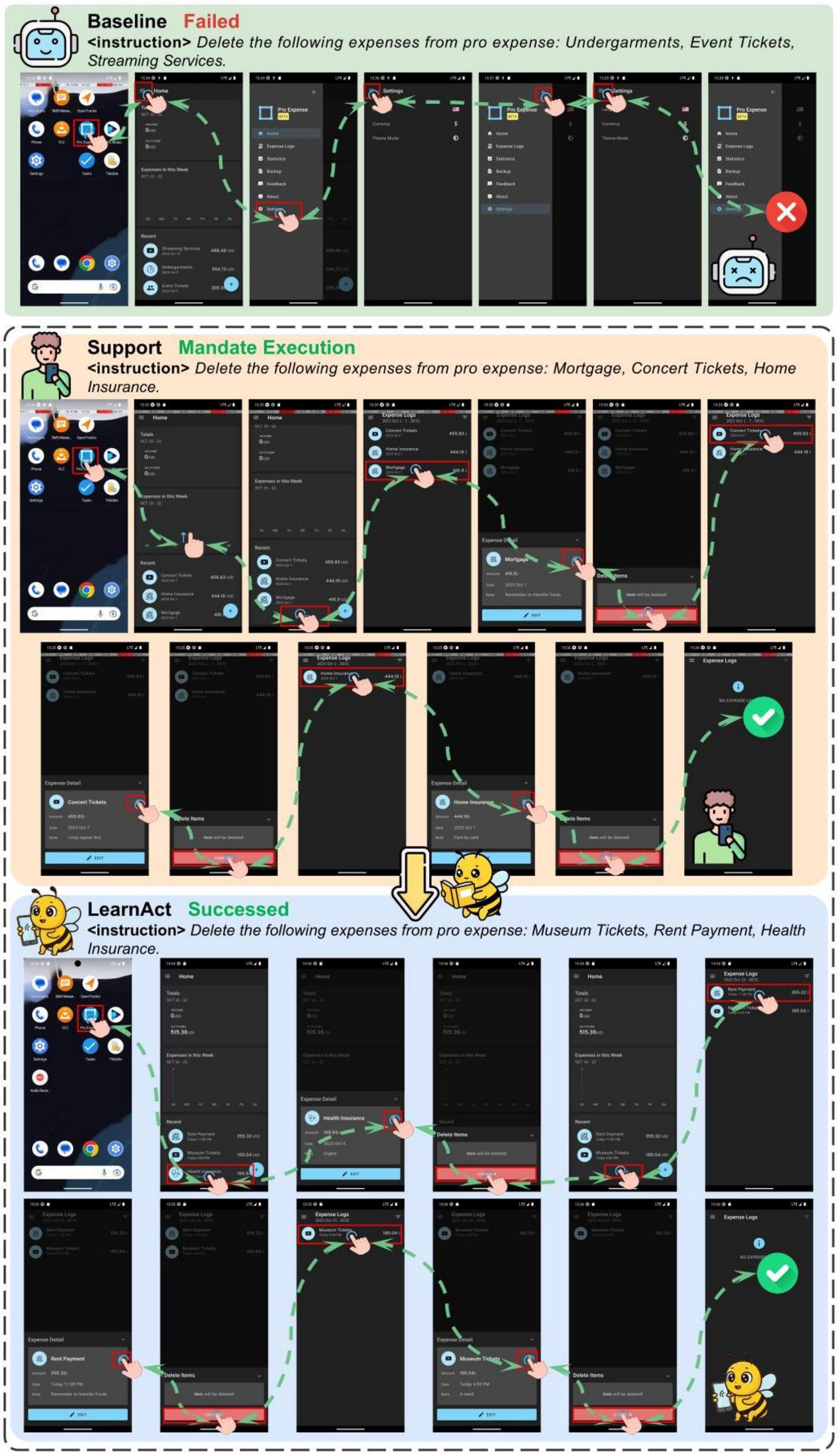

Baseline Failed
<instruction> Delete the following expenses from pro expense: Undergarments, Event Tickets, Streaming Services.
Support Mandate Execution
<instruction> Delete the following expenses from pro expense: Mortgage, Concert Tickets, Home Insurance.
LearnAct Successed
<instruction> Delete the following expenses from pro expense: Museum Tickets, Rent Payment, Health Insurance.

Figure 16: Qwen2-VL-7B with LearnAct vs. Baseline in ExpenseDeleteMultiple Task. Task template: ”Delete the following expenses from arduia pro expense: {expenses}.”

图 16:Qwen2-VL-7B 使用 LearnAct 与基线模型在 ExpenseDeleteMultiple 任务中的对比。任务模板:“从 arduia pro expense 中删除以下费用:{expenses}。”