GUI Agent 调研报告

简介

功能　能够像人类一样，通过“看”屏幕、“操作”鼠标和键盘来使用计算机软件。

意义　改变使用计算机的方式：通过提升计算机自动化水平，提高人的工作效率。

工作流程　感知（Perception）、规划（Planning）与行动（Action）

GUI Agnet 工作流程

感知

屏幕截图/视觉输入　Agent 实时获取屏幕的图像。

多模态大模型（MLLM）　利用像 GPT-4o、Claude 3.5 Sonnet 或 Gemini 这样的多模态大模型，Agent 能够理解屏幕上的内容。

元素识别　它需要识别按钮、输入框、图标、菜单栏的位置。这通常结合了 OCR（文字识别）和目标检测技术。有些先进的 Agent 还会读取系统的辅助功能树（Accessibility Tree）来辅助定位。

规划

任务拆解　用户给出一个模糊指令（例如：“帮我把在这个 Excel 表里的销售数据做成图表并发给客户”）。Agent 需要将这个指令拆解为一系列步骤：打开 Excel -> 选中数据 -> 插入图表 -> 打开邮件 -> 撰写邮件 -> 粘贴图表 -> 发送。

逻辑推理　如果弹出了一个意外的错误窗口，Agent 需要像人一样判断是点击“确定”还是“取消”，而不是直接卡死。

行动

模拟操作 Agent 将决策转化为具体的键鼠操作，例如 click(x=200, y=500)（在坐标处点击）或 type(“Hello”)（输入文字）。

反馈循环 操作完成后，Agent 会再次截图，确认操作是否成功，然后进行下一步。

与传统自动化（RPA/脚本）的区别

特性	传统自动化 (RPA/Selenium)	GUI Agent (AI)
依赖基础	依赖底层代码（HTML ID, DOM 结构）或固定坐标	依赖视觉图像，像人眼一样看
适应性	极差。一旦 UI 改版或按钮位置移动，脚本就会报错	强。按钮换了位置或颜色，AI 依然认得那是“提交”按钮
通用性	专用的。针对特定软件编写特定脚本	通用的。同一个 Agent 可以学着用 Excel，也可以学着用 Photoshop
指令理解	只能执行预设的死命令	可以理解自然语言指令（模糊指令）
处理异常	遇到未预设的弹窗通常会崩溃	可以根据弹窗内容尝试解决或绕过

发展现状

产品/项目	开发方/机构	地址	Stars	发布时间	备注
Claude Computer Use	Anthropic	platform.claude.com…	闭源	2024 年 10 月	性能最佳
Operator	OpenAI	openai.com/index/introducing-operator	闭源	2025 年 1 月 23 日	性能第一梯队
UFO²	Microsoft	github.com/microsoft/UFO	7.8k	2025 年 5 月 6 日	Win 系统深度集成
UI-TARS-desktop	ByteDance	github.com/bytedance…	19.8k	2025 年 1 月 23 日	原生 GUI 桌面应用
Open Interpreter	Open Interpreter	github.com/openinterpreter…	61k	2023 年	star 数最高
self-operating-computer	OthersideAI	github.com/OthersideAI…	10k	2023 年	多模态操作代理
OS-Copilot	上海 AI Lab	github.com/OS-Copilot…	1.7k	2024 年 2 月	通用 OS 代理框架
ShowUI	新加坡国立大学	github.com/showlab…	1.6k	2024 年 11 月	UI 视觉理解
Cradle	智源研究院	github.com/BAAI-Agents…	2.4k	2024 年 3 月	通用控制框架
OpenCUA	香港大学	github.com/xlang-ai…	594	2025 年 8 月	开源 CUA 实现
Agent-S	Simular AI	github.com/simular-ai…	8.6k	2025 年 10 月	搜索与学习机制
AppAgent	腾讯	github.com/Tencent…	6.3k	2023 年 12 月	移动端多模态
MobileAgent	阿里	github.com/X-PLUG…	6.6k	2025 年 8 月	移动端多模态
AutoGLM	智谱华章	github.com/zai-org…	2.2k	2025 年 12 月 9 日	移动端多模态

发展现状 · OpenAI · Operater

Computer Use Agent (CUA)/Operator 是 OpenAI 推出的计算机使用智能体。

Benchmark type	Benchmark	Computer use (universal interface)		Web browsing agents	Human
		OpenAI CUA	Previous SOTA	Previous SOTA	
Computer use	OSWorld	38.1%	22.0%	-	72.4%
Browser use	WebArena	58.1%	36.2%	57.1%	78.2%
	WebVoyager	87.0%	56.0%	87.0%	-

- WebArena 利用离线自托管开源网站来模仿电子商务、在线商店内容管理 (CMS)、社交论坛平台等真实场景。
- WebVoyager 测试模型在亚马逊、GitHub 和谷歌地图等在线实时网站上的性能。

发展现状 · 微软 · UFO²

发布时间 微软于 2025 年 5 月 6 日正式开源 UFO²，这是业内首个深度集成 Windows 操作系统的 AgentOS 桌面智能体平台

论文 https://arxiv.org/abs/2504.14603

项目地址 https://github.com/microsoft/UFO (7.8k stars)

现有 CUA 缺点 多停留在概念/原型阶段、缺乏 OS 深度集成、规模化应用受限

基准测试 1 Windows Agent Arena (WAA)：154 个真实的 Windows 任务，涵盖 15 个应用程序（Office、Edge、文件资源管理器、VS Code 等）。UFO²准确率为 30.5%。

基准测试 2 OSWorld（Windows）：49 个跨应用程序任务，涉及 Office 365、浏览器和系统实用程序。UFO²准确率为 32.7%。

发展现状 · 字节跳动 · UI-TARS-desktop

项目地址　https://github.com/bytedance/UI-TARS-desktop (19.8k)

论文　https://arxiv.org/abs/2501.12326

简介　UI-TARS Desktop 是一款桌面应用程序，基于 UI-TARS 模型提供原生 GUI Agent。支持本地和远程计算机及浏览器操作。

MCP 能力　绘制一个月的杭州天气图

Codeact 能力　修复 Git 使用时的错误

Reasearch 能力　深度调研字节跳动开源模型，给出调研报告。

Ai-coding 能力　编写代码以重现用户界面。

Ai-broswer 能力　打开视频网站并播放视频。

发展现状 · Claude Computer Use

```
import anthropic
client = anthropic.Anthropic()
response = client.beta.messages.create(
    model="claude-sonnet-4-5",  # or another compatible model
    tools=[
        {"type": "computer_20250124",  # 第一版 computer use tool 发布时间：2024年10月
          "name": "computer"},
        {"type": "text_editor_20250728",
          "name": "str_replace_based_edit_tool"},
        {"type": "bash_20250124",
          "name": "bash"}],
    messages=[{"role": "user", "content": "Save a picture of a cat to my
desktop."}]
)
print(response) # https://platform.claude.com/docs/en/agents-and-tools/tool-use/
computer-use-tool
```

发展现状 · 智谱华章 · **AutoGLM**

发布时间　2025 年 12 月 9 日，智谱开源其核心 AI Agent 模型 AutoGLM。

功能　以多模态方式理解手机屏幕内容，并通过自动化操作帮助用户完成任务。系统通过 ADB（Android Debug Bridge）来控制设备，以视觉语言模型进行屏幕感知，再结合智能规划能力生成并执行操作流程。

项目地址　https://github.com/zai-org/Open-AutoGLM

应用示例　在电脑端用自然语言控制手机实现“打开淘宝搜索无线耳机”

发展现状 · 其他

Open Interpreter https://github.com/openinterpreter/open-interpreter (61k stars)

self-operating-computer https://github.com/OthersideAI/self-operating-computer (10k stars)

OS-Copilot https://github.com/OS-Copilot/OS-Copilot (1.7k stars) https://arxiv.org/pdf/2402.07456

MobileAgent https://github.com/X-PLUG/MobileAgent (6.6k stars) https://arxiv.org/abs/2508.15144

ShowUI https://github.com/showlab/ShowUI (1.6k stars) https://arxiv.org/abs/2411.17465

Cradle https://github.com/BAAI-Agents/Cradle (2.4k stars) https://arxiv.org/abs/2403.03186

OpenCUA https://github.com/xlang-ai/OpenCUA (594 stars) https://arxiv.org/abs/2508.09123

Agent-S https://github.com/simular-ai/Agent-S (8.6k stars) https://arxiv.org/abs/2510.02250

AppAgent https://github.com/TencentQQGYLab/AppAgent (6.3k stars) arxiv:2312.13771

Benchmark · OSWorld

简介　这是目前评估 GUI Agent 能力的权威测试环境[1]

作用　提供真实的计算机环境和一系列任务，旨在全面评估智能体的任务执行能力

Rank	Model	Developer	Steps	Date	Success Rate
1	Claude Sonnet 4.5	Anthropic	100	2025-10-31	**62.9%**
2	Claude Sonnet 4.5	Anthropic	50	2025-10-31	58.1%
3	DeepMiner-Mano-72B	Mininglamp Tech	100	2025-10-31	53.9%
4	UI-TARS-2-2509	ByteDance Seed	100	2025-10-14	53.1%
5	opencua-72b-preview	HKU & Moonshot	100	2025-10-01	45.0%
6	opencua-72b-preview	HKU & Moonshot	50	2025-10-01	44.9%

Table 1: GUI Agent Leaderboard

[1] https://os-world.github.io/

技术原理

UI-TARS

简介 UI-TARS 是字节跳动推出的原生 GUI 智能体模型，以纯截图为输入实现类人交互（如键盘、鼠标操作），采用端到端设计，在 10 余个 GUI 智能体基准测试中实现 SOTA 性能。

自主智能体 Autonomous Agents（自主智能体）指的是任何具备感知、规划、决策并执行行动以实现目标的 AI 系统。它们可以通过 API、命令行、文本对话、物理机械臂等多种方式与外界交互。

GUI Agents GUI Agents（图形用户界面智能体）是 Autonomous Agents（自主智能体）的一个子集（Subset）或一种具体化的应用形态。

GUI Agnets 分类

基于文本的 基于文本的 GUI Agents 依赖 HTML、DOM 等结构化文本作为输入，适配性受限

基于视觉的 基于视觉的 GUI Agents 以纯截图为输入，泛化性更强

模块化的 模块化 GUI Agents 依赖拆分的功能模块（如独立的感知、推理模块）和人工优化，泛化性、稳定性弱

端到端的 端到端 GUI Agents（如 UI-TARS）将所有能力整合为单一模型，数据驱动自主学习，跨平台适配性和复杂任务处理能力更优

原生的 基于视觉的、端到端的 GUI Agents，称为 Native GUI Agents

GUI Agents 的四个发展阶段

阶段 1：规则型代理（Rule-based Agents） 完全依赖人工预定义规则，通过匹配用户指令调用固定 API 或模拟简单动作。需直接访问系统底层权限（如 API、HTML 代码），仅能在高度结构化环境中运行。代表作品：RPA 系统、DART、WoB、Roscript、FLIN。

阶段 2：模块化代理框架（Agent Framework） 以 GPT-4、GPT-4o 等基础模型为核心，将感知、推理、记忆等功能拆分为独立模块，通过人工设计的工作流和提示词（Prompt）协调模块协作。代表作品：AutoGPT、LangChain、MMNavigator、SeeAct。

阶段 3：原生代理模型（Native Agent Model） 数据驱动的端到端架构，将感知、动作、推理、记忆能力整合为单一模型，无需人工拆分模块，通过大规模数据自主学习交互逻辑。代表作品：UI-TARS、Aguvis、OS-Atlas、ShowUI。

阶段 4：主动与终身学习代理（Active and Lifelong Agent，展望阶段） 当前未完全实现，目标是最小化人工干预，代理可自主发现任务、探索环境、评估执行结果，并通过自我奖励机制持续优化能力。

Native GUI Agnets 的四个核心能力

感知（Perception） 不仅能识别按钮、文本框等静态 GUI 元素，还能解析元素布局、空间关系、功能属性，以及界面状态的动态变化（如按钮点击后的视觉反馈、页面加载后的内容更新）。

行动（Action） 具备统一的动作空间（如点击、输入、滚动、拖拽等），能标准化不同平台（桌面、移动、Web）的语义等效动作；同时能精准 “接地”（Grounding），即根据感知结果定位元素坐标，确保动作执行的准确性。

推理（Reasoning） 融合两种推理模式 ——System 1（快速直觉推理）用于简单常规操作（如点击熟悉的确认按钮），System 2（深思熟虑推理）用于复杂任务，包含任务分解、长期一致性维护、里程碑识别、试错、反思纠错等模式。

记忆（Memory） 存储任务相关信息，支撑持续决策和经验复用。短期记忆存储当前任务的上下文（如已执行的动作、界面当前状态），支持实时决策调整；长期记忆存储历史交互经验、任务操作规律、背景知识（如不同软件的通用操作逻辑），支持跨任务知识迁移。

感知能力

挑战 数据稀缺：GUI 专用截图远少于通用场景图像，需针对性构建数据集。信息密度高：界面包含数百个元素（如小图标、文本框），需精准识别并理解其空间关系与功能。精度要求高：需捕捉元素细微状态变化（如按钮点击反馈），支持后续精准交互。

GUI 数据集采集 通过专用解析工具，采集截图及元数据（元素类型、边界框、文本内容、层级关系）。统一存储为 “截图 + 元素框 + 元素元数据” 的结构化格式，支撑多任务训练。

五大核心训练任务 通过 “从局部到整体、从静态到动态” 的任务设计，全面强化感知能力：

1. 元素描述：生成 GUI 元素的结构化描述，解决小元素识别难题。
2. 密集字幕：对整个界面生成全景描述，整合所有元素的描述信息，避免局部认知偏差。
3. 状态转换字幕：对比连续截图，识别界面视觉变化，支撑动态场景理解。
4. 问答：构建多样化问答数据，强化模型的视觉推理能力（如 “找出右上角的提交按钮” “当前选中的文本框是什么功能”）。
5. 标记点关联：在截图中为元素添加 markers，提升后续动作定位的准确性。

行动能力

统一动作空间　将不同设备的语义等效动作统一

环境	Action	定义
通用	Click(x, y)	点击坐标 (x, y)。
	Drag(x1, y1, x2, y2)	从 (x1, y1) 拖拽到 (x2, y2)。
	Scroll(x, y, direction)	在 (x, y) 处按指定方向滚动。
	Type(content)	输入指定内容。
	Wait()	暂停片刻。
	Finished()	标记任务已完成。
	CallUser()	请求用户干预。
桌面端	Hotkey(key)	按下指定快捷键。
	LeftDouble(x, y)	在 (x, y) 处双击左键。
	RightSingle(x, y)	在 (x, y) 处单击右键。
移动端	LongPress(x, y)	在 (x, y) 处长按。
	PressBack()	按下“返回”按钮。
	PressHome()	按下“主页”按钮。
	PressEnter()	按下“回车”键。

表 2　不同平台下的统一动作空间

行动能力

大规模动作轨迹数据支撑 整合自研标注数据集（人工执行任务记录轨迹）和标准化开源数据（如 MM-Mind2Web、GUIAct 等），覆盖 Web、移动、桌面多平台。

数据类型		定位 (Grounding)		多步操作 (MultiStep)	
		元素 (Ele.)	元素/图	轨迹 (Trace)	平均步数
开源数据	网页 (Web)	14.8M	6.7	6.4k	7.1
	移动端 (Mobile)	2.5M	4.6	145k	9.6
	桌面端 (Desktop)	1.1M	6.2	0	0
本文 (Ours)		-	7.5	-	14.9

表 3 定位和多步操作轨迹数据的基本统计，对比了 UI-TARS 的标注数据集与不同平台（网页、移动端和桌面端）的开源数据。我们报告了元素数量 (Ele.) 和操作轨迹数量 (Trace)。

提升接地（Grounding）能力 将 GUI 元素描述与空间坐标配对，生成大规模接地数据集（如 “右上角红色提交按钮” 对应具体屏幕坐标）。

推理能力

数据来源 从 MINT、OmniCorpus 等数据集筛选、提纯 600 万条高质量 GUI 教程（含文本和图像），覆盖各类软件、网页的操作逻辑。

任务分解 将复杂目标拆分为可执行的子任务（如 “订机票” 拆分为选出发地、日期、筛选价格等）

长期一致性 全程对标初始目标，避免执行中偏离；

里程碑识别 感知 intermediate 目标完成（如 “已选好日期”），平滑推进后续步骤

试错 在模糊场景中假设并验证操作（如不确定按钮功能时尝试点击并观察反馈）

反思 分析过往动作错误，调整后续策略（如点错按钮后反思原因，重新定位目标）。

记忆能力

挑战 一方面，在实际部署中，Agent 常因误判界面、点击无效按钮陷入循环，却不会反思调整，导致任务卡壳。另一方那个面，现有训练数据多是 “理想化无错轨迹”（标注者刻意避免错误），模型没机会学习 “如何从错误中恢复”。

修正错误记忆 针对模型生成的错误动作轨迹，标注 “错误点 + 修正方案”，构建 “错误轨迹 - 正确轨迹” 配对样本（如点错按钮后，标注 “应重新定位目标按钮”）。

训练方式 通过有监督微调（SFT），让模型记住错误类型及应对策略，避免重复踩坑。

训练方式

底座模型 Qwen-2-VL 7B/72B

训练数据量 总计约 500 亿 tokens，涵盖感知、grounding、动作轨迹、反思调优等全类型数据。

持续预训练（Continual Pre-training） 训练数据为除反思调优数据外的全量数据（含感知任务数据、动作轨迹数据、GUI 教程数据等）。

退火阶段（Annealing Phase） 训练数据为筛选感知、grouding、动作轨迹、反思调优数据中的优质子集，剔除噪声数据。输出模型为 UI-TARS-SFT（有监督微调后的基础模型）。

DPO 阶段（Direct Preference Optimization） 训练数据来自在线轨迹引导的“错误 - 修正”反思配对样本（负样本 + 正样本）。输出模型为 UI-TARS-DPO（最终部署的优化模型）。

实验

感知能力 在 VisualWebBench 上，Qwen2-VL-7B 得分 73.3，而 UI-TARS-7B 得分 79.7

Grounding 能力 在 ScreenSpot 上，OS-Atlas-7B 得分 82.5，而 UI-TARS-7B 得分 89.5

离线 Agent 能力 在 Multimodal Mind2Web 上，Aguvis-7B 得分 64.2，而 UI-TARS-7B 得分 73.1

在线 Agent 能力 在 OSWorld 上，Aguvis-7B 得分 14.8，而 UI-TARS-7B 得分 18.7

UFO

UFO²: The Desktop AgentOS

现有工作缺点 仅通过视觉感知和在固定动作空间行动，未充分利用操作系统和应用内部的 API[1]

中央 HostAgent 解析用户意图、分解子任务、管理应用生命周期、调度 AppAgent、维护全局状态[2]

AppAgent 每个 AppAgent 专门用于特定应用（Excel, Outlook 等）

感知 结构化元数据 + OmniParser-v2（视觉解析）

[1] https://arxiv.org/pdf/2504.14603

[2] 从目前的纯人工操作计算机，到未来的端到端 GUI Agents，本文的工作处于两者中间，可行性更高。对于未来的端到端 GUI Agents，感知完全依赖截图，行动完全依赖键鼠，难度很大。而 UFO，感知和行动依赖底层 API，所以可行性更高。

Agent-S

Agent-S1

感知　结合视觉输入（截图）感知环境变化，以及可访问性树（标注唯一元素 ID）实现精准的 element grounding。

性能　OSWorld 得分 20.58（模型采用 GPT-4o），仅 GPT-4o（无智能体）得分为 11.21[1]

[1] https://www.simular.ai/articles/agent-s

Agent-S2

现有方法缺点 单一通用模型难以适配多任务

感知 Agent S2 仅以原始屏幕截图作为输入，无需结构化的辅助数据。

分层规划（Hierarchical Planning） 根据环境反馈，动态地改进计划，而非遵循固定的行动。

Mixture of Grounding 将每次交互发送给最合适的模型。例如，对于电子表格，使用 structural grounding 模型；对于按钮，使用 visual grounding 模型。

OSWorld Agent S2 在 50 步评估中达到了 34.5% 的准确率（基于 Claude 3.7）

从 Google 云端硬盘下载图片，然后使用 GIMP 进行压缩

启动 GIMP	下载图片	在 GIMP 中打开图片	调整图片大小	导出压缩图片
1. 点击“活动 (Activities)”按钮。 2. 点击“取消”按钮。 3. 点击“X”按钮关闭。 4. 点击“活动”按钮。 5. 在搜索栏输入“GIMP”。 6. 点击 GIMP 应用程序图标	7. 按 Ctrl + Alt + T。 8. 在终端输入 wget	9. 按 Ctrl + O 打开文件。 10. 点击 GIMP 中的“文件”菜单。 11. 点击 GIMP 主窗口。 12. 点击“文件”菜单 13. 点击“打开…” 14. 点击“下载”文件夹 15. 点击用户文件夹 16. 点击名为 ‘uc?exp… 的文件 17. 点击“打开”按钮。	18. 点击“图像 (Image)”菜单 19. 点击“缩放图像 (Scale Image)…” 20. 在宽度输入框中输入 2000 21. 点击“缩放 (Scale)”按钮。	22. 按 Alt + F，然后按 E 打开… 23. 点击“导出为 (Export As)…” 24. 点击“桌面”文件夹 25. 输入 compressed.jpeg 26. 点击“导出”按钮。 27. 在质量字段中输入 60。 28. 点击“导出”按钮

Agent-S3

Computer-use agents (CUAs) 现有方法：单步错误累积、反馈延迟、路径分支多、环境噪声（UI 变化、弹窗）干扰，导致同一代理在不同尝试中表现差异显著。

本文解决方案 提出 Behavior Best-of-N（bBoN） 框架，并行生成多个轨迹再由 VLM 法官进行比较选择最优轨迹

Agent S3 Agent S3 基于 Agent S2 改进，移除了分层规划（hierarchical planning）并新增了代码智能体（coding agent）。OSWorld：62.6%

效果 Agent S3 w/ bBoN + GPT5 在 OSWorld 基准上实现 69.9% 的 100 步成功率，大幅超越此前 59.9% 的 SOTA，接近人类 72% 的水平

ReAct

智能体与环境交互

时间步 t 时，智能体先从环境接收“观察结果”$o_t \in \mathcal{O}$（比如知识库返回的搜索结果、游戏里看到的物品位置）；

智能体根据“上下文”c_t，遵循某个“策略” π 选择一个“动作”$a_t \in \mathcal{A}$（比如继续搜索、拿起物品）；

上下文c_t是从任务开始到当前步的所有历史信息集合，包括之前的观察o_1到 o_{t-1}、之前的动作a_1到 a_{t-1}，再加上当前的观察 o_t，相当于智能体的 “记忆”。

“策略 $\pi(a_t|c_t)$” 本质是 “根据上下文 c_t 选动作 a_t” 的映射关系（$c_t \to a_t$）

OSWORLD

OSWorld: 在真实计算机环境中对多模态代理进行基准测试

与计算机的交互方式 图形用户界面 (graphical user interfaces, GUI) 和命令行界面 (command line interfaces, CLI)

OSWorld 的交互 论文[1]2.2.1 节明确指出，智能体的动作用于操纵键盘和鼠标。说明即使是 CLI 操作也需要通过键鼠转化。

OSWorld 的观测空间 屏幕截图。可选 ally 树。

总任务数 369 个（Ubuntu）。其中：73% 为单应用，27% 为多应用

应用程序 LibreOffice (Calc, Writer and Impress), Thunderbird, VLC Media Player, Chrome, VS Code, GIMP

[1]https://arxiv.org/pdf/2404.07972

单应用任务

关联应用	典型任务指令
OS（Ubuntu）	“我想在当前系统安装 Spotify，能帮我吗?”
LibreOffice Calc（表格）	“检查‘含重复项的姓名’列，将唯一姓名按原顺序填入‘唯一姓名’列”
LibreOffice Impress（演示）	“每次启动 Impress 都会用双屏显示，我只想用一个屏幕展示演示内容，该怎么设置?”
LibreOffice Writer（文档）	“将桌面的 screenshot 1.png 复制到光标所在位置”
Chrome（浏览器）	“清理亚马逊可能保存的追踪数据，确保浏览隐私”
VLC（媒体播放）	“调整设置，让我在看 PDF 讲义时能用快捷键暂停 VLC 视频，无需切换应用”
Thunderbird（邮件）	“创建‘促销’本地文件夹，设置过滤器自动将主题含‘折扣’的收件箱邮件移入该文件夹”
VS Code（编程 IDE）	“修改 VS Code 设置，禁用 Python 缺失导入的错误提示”
GIMP（图像编辑）	“将图片背景设为透明”

OSWorld

测评方式 OSWorld 通过向大模型发送指令（即提示词），让大模型实现操作计算机的能力。

提示词 你是一个智能体，你需要遵循我的指令并按要求执行桌面计算机任务。在每一步中，你将获得一张图像作为观察输入，即计算机屏幕的截图，你需要根据该图像预测计算机的动作。每次返回一行或多行 Python 代码以执行操作。你需要根据对当前画面的观察自行指定坐标。当你认为任务无法完成时，返回 FAIL。当你认为任务已完成时，返回 DONE。

Open Interpreter

使用方式

```
from interpreter import interpreter
# 大模型接受根据自然语言描述的需求，生成代码，agent执行代码以完成需求
interpreter.chat("绘制 META 的标准化股价")
```

interpreter.chat

interpreter/core/core.py

```
class OpenInterpreter:
  def chat(self, message):
    # 大模型接受根据自然语言描述的需求，生成代码，agent执行代码以完成需求
    self._streaming_chat(message)
```

interpreter._streaming_chat

interpreter/core/core.py

```
class OpenInterpreter:
  def _streaming_chat(self, message):
    # 大模型接受根据自然语言描述的需求，生成代码，agent执行代码以完成需求
    self._respond_and_store()
```

interpreter._respond_and_store

```
# interpreter/core/core.py
from .respond import respond
class OpenInterpreter:
  def _respond_and_store(self):
    # 大模型接受根据自然语言描述的需求，生成代码，agent执行代码以完成需求
    respond(self)
```

respond

大模型接受根据自然语言描述的需求，生成代码，agent 执行代码以完成需求

```
# interpreter/core/respond.py
def response(interpreter):
  if interpreter.messages[-1]["type"] == "code":
    # 系统提示词+用户需求描述，即大模型的完整输入
    system_message = interpreter.system_message
    messages_for_llm = [rendered_system_message] + messages_for_llm

    # 大模型根据提示词，生成代码
    interpreter.llm.run(messages_for_llm)

    # 执行代码
    interpreter.computer.run(language, code)
```

系统提示词

```
# interpreter/core/core.py
from .default_system_message import default_system_message
class OpenInterpreter:
  system_message=default_system_message
```

系统提示词

```
# interpreter/core/default_system_message.py
default_system_message = f"""

You are Open Interpreter, a world-class programmer that can complete any goal by executing code.
For advanced requests, start by writing a plan.
When you execute code, it will be executed **on the user's machine**. The user has given you **full and complete
permission** to execute any code necessary to complete the task. Execute the code.
You can access the internet. Run **any code** to achieve the goal, and if at first you don't succeed, try again and
again.
You can install new packages.
When a user refers to a filename, they're likely referring to an existing file in the directory you're currently
executing code in.
Write messages to the user in Markdown.
In general, try to **make plans** with as few steps as possible. As for actually executing code to carry out that plan,
for *stateful* languages (like python, javascript, shell, but NOT for html which starts from 0 every time) **it's
critical not to try to do everything in one code block.** You should try something, print information about it, then
continue from there in tiny, informed steps. You will never get it on the first try, and attempting it in one go will
often lead to errors you cant see.
You are capable of **any** task.

User's Name: {getpass.getuser()}
User's OS: {platform.system()}""".strip()
```

系统提示词

你是 Open Interpreter，一位世界级的程序员，能够通过执行代码完成任何目标。

对于复杂的需求，请先制定一个计划。

当你执行代码时，代码会在用户的计算机上运行。用户已授予你完全且完整的权限，可以执行完成任务所需的任何代码。请执行该代码。

你可以访问互联网。运行任何代码来实现目标，如果一开始没有成功，就反复尝试。

你可以安装新的软件包。

当用户提到文件名时，他们很可能指的是你当前执行代码所在目录中的现有文件。

用 Markdown 格式向用户发送消息。

一般来说，尽量制定步骤尽可能少的计划。至于实际执行代码来完成该计划，对于有状态的语言（如 Python、JavaScript、Shell，而不是像 HTML 这种每次都从零开始的语言），关键是不要试图在一个代码块中完成所有事情。你应该先尝试做一些事情，打印相关信息，然后再根据这些信息，分小步骤继续进行。你不可能一次就成功，试图一步到位往往会导致无法发现的错误。

你有能力完成任何任务。

OS Mode

操作系统模式 是一种高度实验性的模式,允许 Open Interpreter 通过鼠标和键盘进行视觉控制操作系统。

它提供了一种像 GPT-4V 这样的多模态 LLM，内置了必要的工具来截取显示屏的屏幕截图，并与文字和图标等屏幕元素进行交互。

它将尝试使用最直接的方法来实现目标，例如在 Mac 上使用聚光灯来打开应用程序，以及使用 URL 中的查询参数来打开包含额外信息的网站。

进度 操作系统模式是一项正在进行中的工作

启用方式 `interpreter --os`[1]

[1]https://docs.openinterpreter.com/guides/os-mode

OS-Copilot

OS-Copilot: Towards Generalist Computer Agents with Self-Improvement

现有智能体缺点 仅用于单个应用，如网络浏览器、命令行终端、《我的世界》游戏和数据库。

Computer Agents 与操作系统中的各种元素进行交互，包括网络、代码终端、文件、多媒体以及各种第三方应用程序。

OS-Copilot 提供通用交互接口，整合了操作系统操作的常见方式，包括 Python 代码解释器、bash 终端、鼠标/键盘控制以及 API 调用。

FRIDAY 通过自主学习来控制新应用程序。

Claude 3.5 Computer Use

The Dawn of GUI Agent: A Preliminary Case Study with Claude 3.5 Computer Use

Claude 3.5 Computer Use Anthropic 推出的首款支持公开测试的 API-based GUI 智能体

交互方式 纯视觉观察（截图），不使用 HTML 元数据或软件 API

工具集 1. 计算机工具：键鼠操作（点击 / 拖拽 / 快捷键）、截图；2. 编辑工具：文件查看 / 创建 / 替换；3. Bash 工具：Linux 命令执行

测试框架 Computer Use OOTB[1]

[1]https://github.com/showlab/computer_use_ootb

Claude 3.5 Computer Use 的系统提示词

系统概述

你可使用一组函数（工具）与沙盒化计算环境交互。

可用函数（工具）

1. Computer Tools：使用鼠标和键盘与计算机交互并截取屏幕截图。
2. Bash Tools：在 Bash 终端中执行命令。
3. Computer Tools：查看、创建和编辑文件。

系统能力

可通过 apt 或 pip 安装应用程序和 Python 包。

Tool Schema

作用 是 JSON 格式的文本，作为 LLM 系统提示词的一部分，指导 LLM 输出操作电脑的指令。

Computer Tool Schema 如下 json 文件是 LLM 系统提示词的一部分，告诉 LLM 如何使用 Computer Tools[1]

```
{
  "properties": {
    "action": {
      "description": "可执行的动作有：打字、单击、移动鼠标……",
      "enum": ["key", "type", "..."],
    },
    "coordinate": {
      "description": "(x, y)：x（距离左边缘的像素数）和 y（距离上边缘的像素数），即鼠标需要移动到的坐标。",
    },
}
```

[1] https://arxiv.org/pdf/2411.10323

Cradle

Cradle: Empowering Foundation Agents Towards General Computer Control

交互方式 屏幕截图作为输入,键盘和鼠标用于交互[1]

OSWorld 得分 7.81

[1]https://baai-agents.github.io/Cradle/

Fara

Fara-7B: An Efficient Agentic Model for Computer Use

论文 2025 年 11 月 24 日发表 https://arxiv.org/pdf/2511.19663

代码 2025 年 12 月 11 日更新 https://github.com/microsoft/fara

模型 2025 年 11 月 24 日上传 https://huggingface.co/microsoft/Fara-7B

任务 Web（网页应用任务），而非操作系统（Desktop/OS)任务。

动作空间

动作	描述
按键	按指定顺序按下按键（例如 **CTRL+C**）。
输入	在坐标 (x, y) 处输入字符串。
移动鼠标	移动光标悬停在坐标 (x, y) 上。
左键单击	在坐标 (x, y) 处单击鼠标左键。
滚动	滚动鼠标滚轮。
访问链接	访问指定的 URL。
网络搜索	使用指定查询进行网络搜索。
历史回退	返回上一页。
记忆	记忆信息以供将来参考。
等待	等待指定的秒数。
终止	结束当前任务。

Holo1.5

Holo1.5 - Open Foundation Models for Computer Use Agents

博客 https://www.hcompany.ai/blog/holo-1-5[1]

任务 UI element localization + UI VQA[2]

视觉定位 在 CU 智能体中，VLM 接受两个输入：自然语言任务描述（打开 Spotify 应用）和计算机界面屏幕截图。输出为截图的坐标（“点击 X，Y”）。

视觉理解 除定位外，CUA 还需正确理解截图。这项能力通过 VQA 任务进行评估。模型会被问及关于界面的自然语言问题，例如“当前激活的标签页是什么?”或“用户是否已登录?”，并且必须根据软件的视觉状态给出正确的答案。

---

[1]模型位于 https://huggingface.co/collections/Hcompany/holo15

[2]现有大模型同样能完成这两个任务。只是 Holo1.5 针对这两个任务进行了专门的后训练，性能有微小提升。

Surfer 2

Surfer 2: The Next Generation of Cross-Platform Computer Use Agents

论文 2025 年 10 月 24 日 https://arxiv.org/pdf/2510.19949

博客 2025 年 11 月 25 日 https://www.hcompany.ai/blog/surfer-2

平台 网页+桌面+手机

模型 不微调模型，而是依赖现有前沿模型。

感知 屏幕截图。不依赖 SoM、DOM。

Agent Architecture

设计理念 将规划能力与执行能力解耦。Orchestrator 负责规划，Navigator 负责执行。

架构 Orchestrator + Navigator(policy+Localizer) + Validator

Orchestrator 将用户任务分解子任务，委派给 Navigator。

Localizer 定位任务需要根据用自然语言描述的需求，输出截图图像中找出相关元素（按钮、文本段的）的精确坐标。本文依赖 Holo1.5 模型进行定位。

Validator 子任务完成后，验证器会检查最新的屏幕截图，判断该子任务是否成功。

性能 OSWorld(`pass@1`)得分 60.1%

GUI Agent 选型

参选 GUI Agent

UI-TARS-desktop　字节跳动出品。最近更新时间：2025 年 12 月 11 日。平台???

UFO²　微软出品。仅支持 Windows。

Open Interpreter　star 数最高。但 OS Mode 仍处于开发阶段。

OpenAI Operator　闭源。仅对 OpenAI 美国 pro 订阅者开放。

Surfer 2　闭源。

Fara-7B　微软出品。仅支持 Web。

Claude Computer Use　闭源。

运行环境

3 分 兼容 Windows, Ubuntu, Web

2 分 仅支持 Web

1 分 仅支持 Windows 或仅支持 Ubuntu

性能

3 分　第一梯队

2 分　第二梯队

1 分　第三梯队

社区

3 分　近 1 个月内更新

2 分　近 6 个月内更新

1 分　2025 年未更新

可访问性

开源　3 分

闭源　0 分

UI-TARS-desktop (字节跳动)

总分 11 分

运行环境 (2 分) 不支持 Ubuntu

性能 (3 分) UI-TARS-1.5 OSworld(100 steps) = 42.5[1]

社区 (3 分) 最近更新日期：2025 年 12 月 11 日

[1] https://github.com/bytedance/UI-TARS

UFO²

运行环境 (2 分)　不支持 Ubuntu

cua