

OpenCUA: Open Foundations for Computer-Use Agents

論文情報

- **タイトル:** OpenCUA: Open Foundations for Computer-Use Agents
- **著者:** Xinyuan Wang, Bowen Wang, Dunjie Lu, Junlin Yang, Tianbao Xie, Junli Wang, Jiaqi Deng, Xiaole Guo, Yiheng Xu, Chen Henry Wu, Zhennan Shen ほか30名 (XLANG Lab, The University of Hong Kong、Moonshot AI、Stanford University、University of Waterloo、Carnegie Mellon University)
- **arXiv ID:** 2508.09123
- **公開日:** 2025年8月12日 (最終更新: 2025年10月4日 v3)
- **分野:** 人工知能 (cs.AI) 、コンピュータビジョン (cs.CV)
- **GitHub:** <https://github.com/xlang-ai/OpenCUA>
- **プロジェクトページ:** <https://opencua.xlang.ai/>

概要

本論文は、Computer-Use Agent (CUA : コンピュータ使用エージェント) の研究における透明性の欠如という重大な問題に取り組むものです。既存の最先端CUAシステム (Anthropic Claude、OpenAI Operatorなど) は、そのトレーニングデータ、アーキテクチャ、開発プロセスの詳細が非公開であり、これが技術進歩を妨げ、安全性に関する懸念を引き起こしています。

OpenCUAは、この問題を解決するための完全なオープンソースフレームワークを提供します。具体的には、22,625件のヒューマンアノテーションによるデスクトップタスクを含む**AgentNetデータセット**、クロスプラットフォーム対応の**AgentNet Toolアノテーションツール**、データ処理パイプライン、トレーニングレシピ、そして評価ベンチマークを公開しています。

最大規模の**OpenCUA-72B**モデルは、OSWorld-Verifiedベンチマークにおいて**45.0%**の成功率を達成し、オープンソースモデル中で最高性能 (SOTA) を確立しました。これは、Claude 4 Sonnet (41.5%) を上回る性能です。

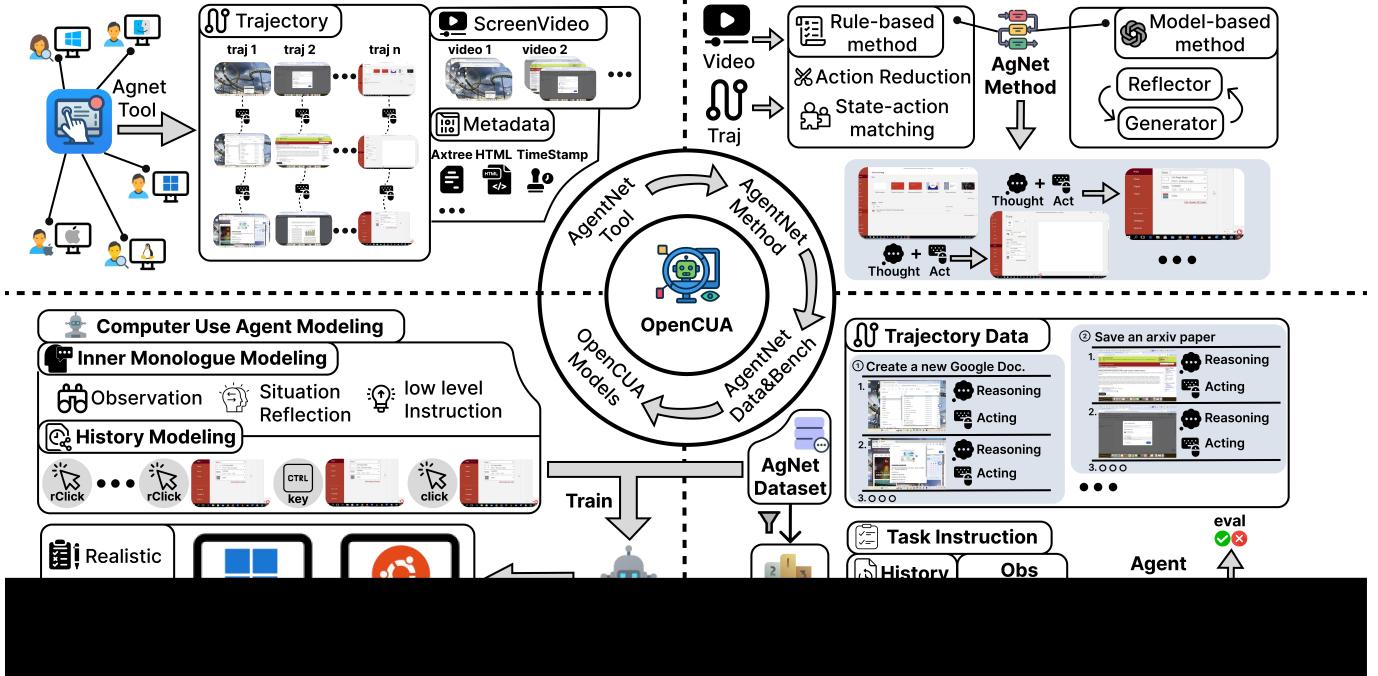


図1: OpenCUA フレームワークの概要。左上 : AgentNet Toolによるユーザーインタラクションのキャプチャ、右上 : 状態-アクションペアへの変換、右下 : AgentNet データセットとベンチマーク、左下 : 学習されたOpenCUA モデル

先行研究との関係

CUAベンチマークとデータセット

コンピュータ使用エージェントの評価は、主に実際のソフトウェア環境でエージェントを実行する**実行レベルベンチマーク**によって行われています：

1. デスクトップベンチマーク

- OSWorld : Linux、Windows、macOSを対象とした多段階ワークフロー
- WindowsAgentArena : Windows中心の154タスク
- AgentStudio : 汎用エージェント構築ツールキット
- UI-Vision : プロフェッショナルソフトウェアと空間推論を評価

2. Webベンチマーク

- WebArena、VisualWebArena : 自己ホストまたはライブサイトでの動的コンテンツナビゲーション
- Mind2Web : 情報検索を含む複雑なWebタスク
- WorkArena : 実務的なWebワークフロー

3. トレーニングデータパイプライン

- OmniAct、SeeClick、UGround : クロスデバイスグラウンドティングとアクションログ
- AgentTrek、TonGUI : チュートリアルからトラジェクトリへの変換
- Aguvis : 既存GUIデータセットの統合

AgentNetの独自性: 従来のデータセットと比較して、AgentNetは以下の点で優れています：

- **規模:** 22,625トラジェクトリ (平均18.6ステップ/タスク)
- **多様性:** 140以上のアプリケーション、190以上のWebサイト

- **マルチプラットフォーム:** Windows (12K)、macOS (5K)、Ubuntu (5K)
- **リアリズム:** ユーザーの実際のコンピュータ環境からキャプチャ
- **複雑性:** マルチアプリワークフロー、プロフェッショナルツール、稀な機能を含む

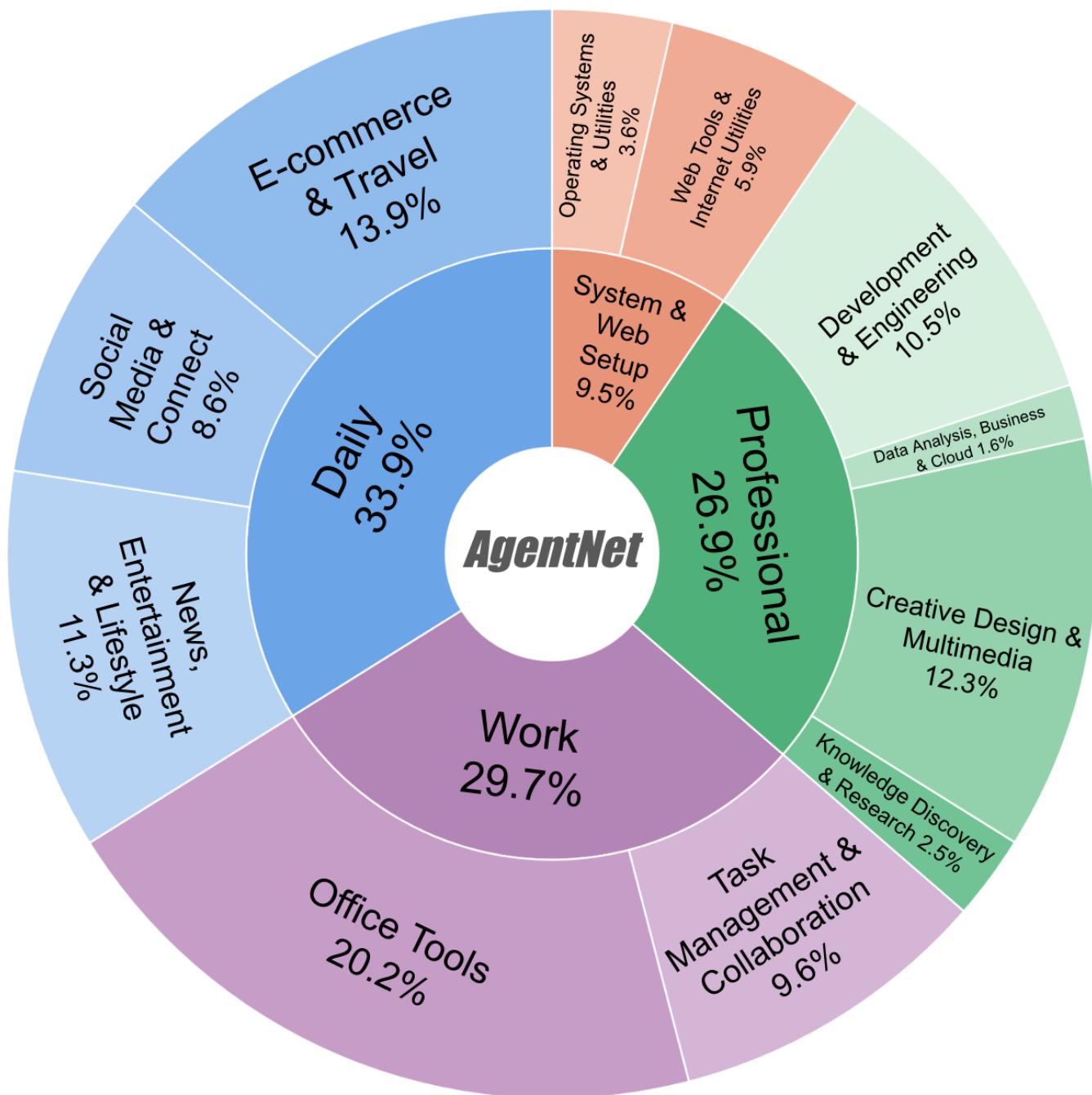


図2: AgentNetデータセットのドメイン分布

CUAフレームワークとモデル

先行研究は3つのカテゴリに分類されます：

1. **テキストベース:** DOM木やアクセシビリティラベルを使用して象徴的なコマンドを発行 (WebGPT、Lemur)
2. **ビジョン中心:**
 - グラウンドディング特化型：自然言語参照をバウンディングボックスや座標クリックと関連付け (UGround、OSAtlas)
 - エンドツーエンド：スクリーンショットを直接アクションシーケンスに変換 (Aguvis、UITars、OpenAI Operator、Claude Computer Use)

3. エージェントフレームワーク: LLMを特殊化されたビジョンエンコーダ、階層的プランナー、エピソードメモリなどでラップ (Agents、CAMEL)

OpenCUAの立ち位置: ビジョン中心のエンドツーエンドアプローチを採用しつつ、**Reflective Long Chain-of-Thought (反省的長CoT) 推論**という新しい手法を導入し、従来手法 (Aguvis、UITars) を大幅に改善しています。

主要な貢献

1. AgentNet Tool : クロスプラットフォームアノテーションツール

従来のデータ収集手法の課題を解決するため、OpenCUAは**AgentNet Tool**を開癆しました。このツールは以下の特徴を持ちます：

- **非侵入的記録:** ユーザーのワークフローを中断せずにバックグラウンドで動作
- **マルチモーダルキャプチャ:**
 - スクリーンビデオ (720p~4K)
 - マウスとキーボード信号
 - アクセシビリティツリー (Axtree)
- **クロスプラットフォーム:** Windows、macOS、Ubuntu対応
- **検証機能:** アノテータが軌跡を確認、編集、提出可能

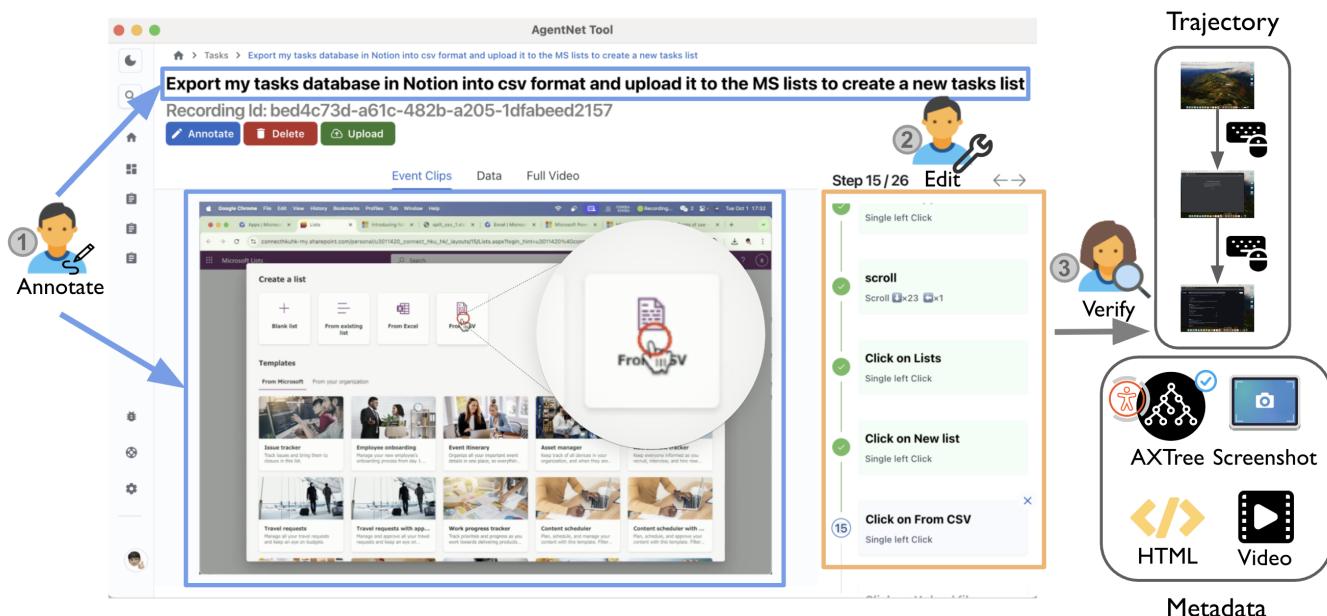


図3: AgentNet Toolのアノテーションと検証インターフェース

実装詳細:

- マウス/キーボードトラッキング: DuckTrack、OpenAdaptベース
- スクリーン録画: OBS Studio
- Axtree解析: OSWorldフレームワーク

アノテーションパイプライン:

1. **多様性と複雑性の重視:** 200以上のアプリ/サイトリストを提供し、複雑なワークフローを奨励
2. **品質基準:** 15ステップ以上のタスクを要求 (5ステップ未満は拒否)
3. **品質ラベル:** rejected、ok、good、excellentの4段階評価

4. プライバシー保護: 多層プライバシー保護メカニズム
5. データ分割: Windows/macOSとUbuntuを分離 (OSWorldとの漏洩防止)

2. データ処理パイプライン：コンパクトな状態-アクションペアの構築

生のヒューマンデモンストレーションは、高頻度で冗長なアクション（1タスクで数千の低レベルアクション）を含みます。OpenCUAは以下の技術でこれを解決します：

(1) アクション削減 (Action Reduction)

ルールベース手法で密なアクション信号を圧縮：

- マウス移動: クリック/ドラッグの前提条件として扱い、開始/終了位置のみ保持
- スクロール: 単一方向アクションにマージし、ホイールカウントを累積
- キー入力: 遠続したキープレスをテキスト入力文字列にマージ
- 修飾キー: CTRL+Cなどをホットキーアクションに抽象化
- ジェスチャ: ドラッグやダブルクリックなどのマルチステップジェスチャを統合

最終的に、PyAutoGUIアクション空間に適合（表1参照）。

(2) 状態-アクションマッチング (State-Action Matching)

各アクション\$a_i\$を、アクション発生直前のシステム状態\$s_i\$とペアリング：

- キーフレーム抽出: スクリーン録画から代表的なフレームを抽出
- 情報漏洩防止: マウスクリックの場合、マウスがすでにボタン上にある状態を避けるため、マウス移動の開始時点までバッファトラック
- 視覚的差異検出: L1距離閾値0.99で最後の視覚的に異なるフレームを検索
- 終了フレーム: 最終アクション後に終了フレームと終了アクションを追加

結果として、軌跡は以下の形式になります： $\tau = (\langle s_0, a_0 \rangle, \langle s_1, a_1 \rangle, \dots, \langle s_T, a_T \rangle)$

ここで、\$I\$はタスク命令、\$s_i\$はスクリーンショット、\$a_i\$はPyAutoGUIアクションです。

3. Reflective Long CoT推論の合成

OpenCUAの最も重要な貢献の1つは、反省的長Chain-of-Thought (CoT) 推論の合成手法です。

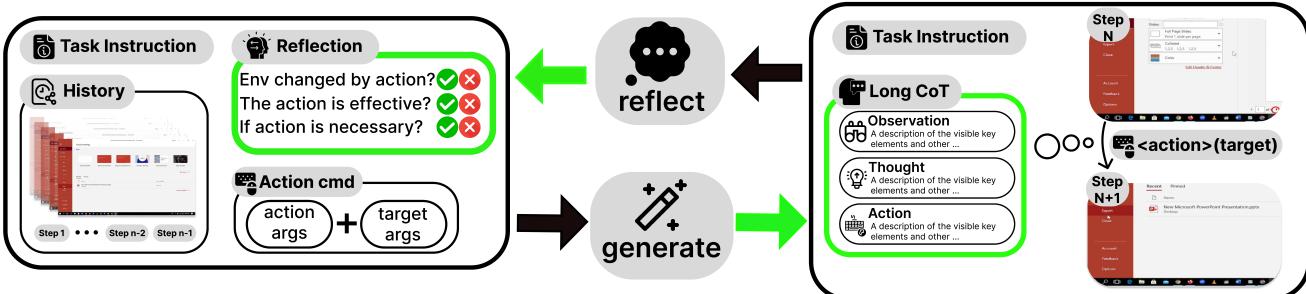


図4: Reflective Long CoT合成パイプライン : GeneratorとReflectorが推論コンポーネントを反復生成・検証

3レベルCoT階層構造:

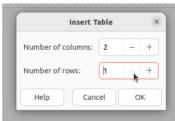
- L3 (Observation + Thought + Action) : 最も詳細

- 画面の視覚的・テキスト的要素の文脈的観察
 - 反省的推論
 - 実行可能アクション
- **L2 (Thought + Action)** : バランス型 (推奨)
 - 状態遷移の分析
 - 過去のステップの想起
 - エラー訂正
 - 次のアクションの計画
 - 実行可能アクション
 - **L1 (Action)** : 最も簡潔
 - 実行可能アクションのみ
- 反省増強 (Reflection Augmentation) :**
- 従来手法 (Aguvis、ActRe) を拡張し、3つのコンポーネントで構成：
1. **Reflector (反省器)** :
 - 各ステップの正確性と冗長性を検査
 - アクション前後のスクリーンショットを比較
 - アクションコードとCoTの正確性を検証
 - エラー/冗長な場合：理由を詳述し、トレーニング時に無視
 - 正しい場合：アクションがもたらす状態変化を説明
 2. **Generator (生成器)** :
 - 完全なエージェントコンテキストに基づいてCoTを生成
 - 入力：過去の反省、アクション履歴、タスク目標、スクリーンショット、アクションコード
 - 視覚的手がかり:
 - マウスアクション座標に赤色マーカー
 - ズームイン画像パッチ (V^* にインスペクタ)
 - 座標関連アクションのグラウンディング精度を向上
 3. **Summarizer (要約器)** :
 - 曖昧なユーザー記述目標を明確で整合性のあるタスク目標に洗練
 - 各軌跡を整合性、効率性、難易度でスコアリング

使用モデル: Claude-3-7-Sonnet-20250219

Task: In the "Features" slide, insert a table with 5 rows and 2 columns.

Step 8: Click on the "Number of rows:" input field to ensure it's selected for editing.



Error:
Model mistakenly click the wrong place changing the rows from 2 to 1, instead of 5

Step 9 Long CoT

Reflection
I can see that we've made progress in setting up the table for the Features slide. The number of columns is correctly set to 2, which aligns with our goal. However, the number of rows is currently set to 1, but we need 5 rows according to our goal.

Memory
Looking at the previous actions, we've already:
 1. Navigated to slide 3 (the Features slide)
 2. Activated the content area for editing
 3. Opened the Insert menu and selected "Table..."
 4. Set the number of columns to 2
5. Set the number of rows to 1 (though we need to change this to 5)

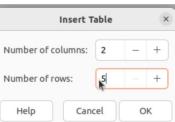
Plan & Reasoning
The current state shows that we're in the Insert Table dialog with the number of rows field active and showing "1". Since we need 5 rows instead of 1, we need to change this value. There are a few ways to do this:
 1. We could click the increase button multiple times to increment from 1 to 5
 2. We could delete the current value and type "5" directly
3. We could select the current value and replace it with "5"

Prediction
After selecting the "1", the next logical step would be to type "5" to replace it, and then click the OK button to create the table with our desired dimensions (5 rows and 2 columns).

Step 9: Click and drag across the "1" in the Number of rows field to select it for replacement.



Step 10: Type "5" to replace the selected "1" in the Number of rows field.



✓ Correctly resolved former mistake

図5: Reflective Long CoTの例。モデルは過去の誤りを特定し、後続ステップで訂正

効果: セクション6の分析により、この反省モジュールがパフォーマンス向上の重要な駆動力であることが実証されました。

4. コンテキストエンコーディングとテスト時推論

コンテキストエンコーディング:

1. テキスト履歴: 構造化された内部モノローグフレームワーク

- ダイアログスタイル履歴形式
- L1 CoT (Action) で過去ステップを表現 (トークン効率的)
- メモリコンポーネントを含む

2. 視覚履歴: マルチ画像スクリーンショット履歴

- デフォルト: 3枚のスクリーンショット (パフォーマンスと効率のバランス)
- テキスト要約より情報損失が少ないロスレス視覚表現

テスト時推論フォーマット:

- 学習時: L1、L2、L3の混合
- 推論時: L2 CoTフォーマットを採用 (より豊富な推論内容)
- L2により、Pass@\$n\$成功率が大幅に向上

5. トレーニングデータミックスチャード

CoTデータミックスチャード:

3レベルのCoTすべてを混合してトレーニング:

- L1: アクションとの直接的な接続が最も強い

- **L2:** 計画と予測を含み、L1の予測アクションに直接影響
- **L3:** 有用なスクリーンショット知覚情報を含むが、無関係な要素も含まれる可能性

この混合により、異なるレベルの接続を強化。

グラウンディング、計画、一般SFTデータの混合:

汎用CUA基盤モデルには、以下のデータタイプを混合：

1. グラウンディングデータ:

- ShowUI、UGround
- AXTree構造から解析した189K境界ボックスサンプル

2. 計画と推論データ:

- UbuntuおよびWindows/macOSデモンストレーション
- タスク命令拡張サンプル

3. 一般SFTデータ (Kimi Teamより) :

- テキストデータ：命令フォロー、数学的推論、長コンテキスト理解
- ビジョンデータ：OCR、ビジョンQA

驚くべきことに、GUI環境に直接関連しない一般データも、エージェントの全体的なパフォーマンス向上に寄与します。

3つのCUAトレーニング戦略:

1. Stage 2のみ (限定リソース用) :

- 一般VLMを特化CUAに適応
- データ混合：70% CUA（計画:グラウンディング = 4:1）+ 30%一般SFT
- Qwen2-VLで30Bトークン、Kimi-VL-A3Bで20Bトークン

2. Stage 1 + Stage 2 (より多くのリソース用) :

- **Stage 1:** グラウンディングと理解強化
 - グラウンディング軌跡、チュートリアル、状態遷移キャプション、一般VLタスク、一般テキストSFT
 - Qwen2.5-VL-32Bで40Bトークン
- **Stage 2:** CUA計画に焦点
 - 45%計画 + 20%グラウンディング + 残り一般データ
 - 結果：OpenCUA-32B
- Qwen2.5-VL-72Bも同様の戦略で学習（より多くのデータ）

3. Joint Training (汎用VLM + CUA能力用) :

- ドメイン間でバランスの取れたデータ混合
- マルチ画像軌跡データを3エポック学習
- データ比率：20%計画 + 20%グラウンディング + 60%一般
- Qwen2.5-VL-7Bベースで200Bトークン予算

- 結果：OpenCUA-7B（7Bスケールで最高のオープンソースCUA）

実験結果

実験設定

モデル:

- Kimi-VL-A3B (MoE、16B総パラメータ、3Bアクティブ)
- Qwen2-VL-7B-Instruct
- Qwen2.5-VL-7B/32B/72B-Instruct

注意: M-Ropeは実装されておらず、1D RoPEを使用。Kimi-VL-A3Bのチャットテンプレートとトークナイザーに統一。

評価ベンチマーク:

1. オンラインエージェント評価:

- **OSWorld-Verified**: 369タスク (Ubuntu中心)、15/50/100ステップ予算
- **WindowsAgentArena (WAA)** : 154 Windowsタスク (4つのクロックタスクはAPI制限で除外)

2. オフラインエージェント評価:

- **AgentNetBench**: 100代表的タスク (Windows/macOS)

3. GUIグラウンディング評価:

- OSWorld-G (564サンプル)
- ScreenSpot-V2 (モバイル、デスクトップ、Web)
- ScreenSpot-Pro (高解像度デスクトップ)
- UI-Vision (プロフェッショナルソフトウェア、空間推論)

評価設定:

- 解像度：1920×1080
- L2 CoTフォーマット (Thought + Action)
- Temperature : 0 (決定論的デコーディング)
- OSWorld-Verifiedは3回実行の平均

メイン結果

1. OSWorld-Verifiedオンライン評価

モデル	15ステップ	50ステップ	100ステップ
Claude Sonnet 4.5	55.3%	60.7%	61.4%
OpenCUA-72B	38.6%	43.9%	45.0% ★
Claude 4 Sonnet	34.0%	46.7%	44.3%
OpenCUA-32B	29.7%	34.1%	34.8%

モデル	15ステップ	50ステップ	100ステップ
OpenCUA-7B	22.5%	26.9%	27.3%
Qwen2.5-VL-72B	3.3%	4.4%	5.0%
Kimi-VL-A3B	1.1%	1.4%	1.7%

主要な発見:

1. プロプライエタリモデルがリードするも、差は縮小
 - Claude Sonnet 4.5: 61.4% (100ステップ)
 - OpenCUA-72B: 45.0% (オープンソース最高、Claude 4 Sonnetの41.5%を上回る)
2. OpenCUA手法は異なるアーキテクチャとサイズに適用可能
 - 4つのモデルアーキテクチャで検証 (MoEと密アーキテクチャ)
 - アクティブパラメータ数: 3B~72B
 - すべてでベースモデルから大幅改善
 - モデルサイズとともにパフォーマンスがスケール
3. ステップ制限の効果
 - 15→50ステップ: 頗著な改善
 - 50→100ステップ: 改善は小さい
 - 理由:
 - ほとんどのタスクは15~50アクションで完了
 - エラー認識、回復、停止判断がまだ不十分
 - 幻覚と反復ループが追加ステップを浪費
4. OpenCUAモデルはPass@\$n\$で著しく高いスコア
 - OpenCUA-32B: 34.2% (Pass@1) → **45.6%** (Pass@3)
 - OpenCUA-72B: 45.0% (Pass@1) → **53.2%** (Pass@3)
 - 将来のポストトレーニング、ランキング、マルチエージェント手法の余地が大きい

2. AgentNetBenchオンライン評価

モデルを2グループに分類:

- **Zero-shot**: AgentNetで学習していない (Qwen2.5-VL、Aguvis、OpenAI CUA)
- **Fine-tuned**: OpenCUA-7B、OpenCUA-32B

主要な発見:

1. モデルサイズとともにスケール: Zero-shotグループでパフォーマンスがサイズと相関
2. OpenAI CUAは未見タスクで汎化良好: 特に終了状態検出とコンテンツベースアクションで優秀
3. オフラインベンチマークはオンラインベンチマークとランキング相関: OpenCUA-32B > OpenAI CUA > Qwen2.5-VLモデル
4. 座標アクションパフォーマンスはグラウンドディングパフォーマンスを反映: OpenCUA-32BがOpenCUA-7Bを上回る

3. GUIグラウンディング評価

モデル	OSWorld-G	ScreenSpot-V2	ScreenSpot-Pro	UI-Vision
OpenCUA-72B	59.2%	67.0%	60.8%	37.3% ★
OpenCUA-32B	58.5%	68.0%	55.3%	33.5%
OpenCUA-7B	51.2%	57.0%	50.0%	28.6%
Qwen2.5-VL-72B	50.4%	67.0%	52.8%	31.2%
Qwen2.5-VL-32B	50.2%	66.0%	51.7%	30.3%

主要な発見:

1. **OpenCUA-72BとOpenCUA-32Bが1位:** すべての主流GUIグラウンディングベンチマークで最高
 - UI-Visionで37.3% (SOTA)
 - ScreenSpot-Proで60.8%
 - 理由 : (i) Stage-1での大規模グラウンディングコーパス、(ii) 大きなパラメータスケール
 2. **Joint-trainingがOpenCUA-7Bを強化:** 大規模グラウンディングデータの注入により競争力のあるスコア
 3. **Qwen2.5-VLのピクセル予算の利点:**
 - 最大ピクセル数 : 12,845,056 (Qwen2-VLとKimi-VL-A3Bの829,440に対して)
 - 高解像度ScreenSpot-Proで強い結果
 4. **グラウンディングだけでは不十分:**
 - Qwen2.5-VL-32BはOSWorld-GとScreenSpot-V2でOpenCUA-7B/A3Bに匹敵または上回る
 - しかしOSWorldでは大幅に劣る (19.9%/23.0% vs 5.0%未満)
 - 結論 : 堅実なグラウンディングは必要だが不十分。高レベルの計画と反省的推論が信頼性の高いタスク完了を実現
 5. **ドメイン相関:** OSWorld-G (Ubuntu) スコアはオンラインOSWorld評価と相関が高い
- 4. データスケーリングによるパフォーマンススケーリング**

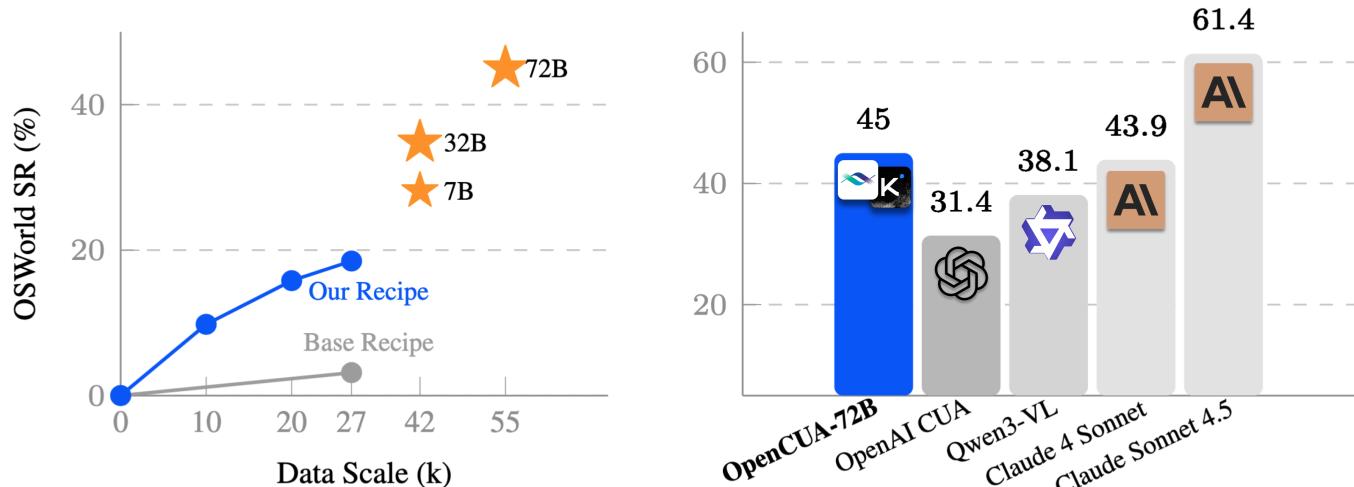


図6: データ量増加によるパフォーマンススケーリング

クロスドメインデータの効果:

- 7K Ubuntu単独: 9.8%
- 7K Ubuntu + 14K Win&Mac: 大幅改善
- 10K Ubuntu + 17K Win&Mac: **18.5%**
- 異なるプラットフォームからのデータでも負の転移なし

ドメイン内/外データスケールの影響:

- Ubuntuデータ 3K→10K: 平均72%改善
- Win/Macデータ 3K→14K: 平均125%改善
- データ量とエージェントパフォーマンスに強い正の相関

詳細分析

1. テスト時計算量スケーリングによるモデル性能上限分析

Pass@\$n\$評価 (温度0.1) :

ステップ予算	Pass@1	Pass@4	Pass@8	Pass@16
15ステップ	16.9%	24.4%	29.8%	34.6% (+104%)
30ステップ	17.9%	25.9%	31.8%	37.2%
50ステップ	18.4%	26.6%	33.2%	39.2% (+113%)

主要な発見:

1. Pass@1とPass@16の間に顕著なパフォーマンスギャップ
2. \$n\$が大きいほど、ステップ予算増加からの利益が大きい
3. 分散の要因 :
 - 異なる実行で異なる解決策を選択
 - 軽微な省略または余分な操作（「保存」クリックの忘れなど）
 - 環境動的性：CAPTCHA、マシン変動性、ネットワーク遅延

2. エージェントモデルのロバスト性不足

温度0でのPass@N評価でも、初期状態の微小な変動（システム日付など）が大きく異なる軌跡を生成：

- Pass@1 (温度0) : 20.10%
- Pass@16 (温度0) : 38.60%
- 18%以上の絶対差

結論: 決定論的設定でも、一見些細な要因（時間的コンテキストなど）が多段階推論に重大な影響を与える可能性。

3. クロスプラットフォーム学習が汎化を改善

ドメイン間のパフォーマンスギャップは存在するが：

- Ubuntuデータで学習：OSWorldで優れる
- Windows/macOSデータで学習：WindowsAgentArenaとAgentNetBenchで優れる

興味深いことに、WAAでのギャップはOSWorldより狭い → **アプリケーションレベルの知識はOSを超えて部分的に転移可能。**

4. L2推論フォーマットが最高の推論パフォーマンス

L1、L2、L3の混合で学習したが、推論時のベストフォーマットは？

CoTフォーマット	OSWorld成功率 (15ステップ)
L2 (Thought + Action)	最高
L1 (Action)	L2より低い
L3 (Obs + Thought + Action)	L2より低い

理由:

- L2 CoTは高品質（計画と反省を含む）→ より良い意思決定
- L3は無関係な要素を多く含む可能性 → モデルを誤導

Aguvis/UITarsとの違い: 彼らの結論はL1 > L2だったが、OpenCUAの高品質L2 CoTではL2が優れる。

5. 適度な視覚履歴画像数と簡潔なテキスト履歴が最適

視覚履歴: スクリーンショット数（1、3、5）でアブレーション

結果：

- 1枚より複数枚で大幅改善（GUIエージェントは状態変化観察に完全に視覚に依存）
- 3→5枚：限界利益、~3Kコンテキストトークン増加、収束遅延
- **結論:** 3枚のスクリーンショットが最適

テキスト履歴: L1 vs L2履歴（同じ3画像設定）

結果：

- L2履歴は利益なし、幻覚を導入する可能性
- トレーニング効率も低下
- **結論:** L1 CoT + 3画像をデフォルト設定

6. CoTミックスチャーがL2のみより優れる

L2推論が推論時にベストなら、L2のみで学習すべき？

実験: OpenCUA-7Bと同じレシピだが、L1/L2/L3混合をL2のみに置換

結果：

- L2のみ: 13.1%
- ミックスチャー (OpenCUA-7B) : **23.0%**

結論: 異なるレベルのCoTを混合してトレーニングすることで、異なるレベルの接続を強化し、より良いパフォーマンスを実現。

7. 一般ドメインテキストデータがエージェントパフォーマンスに正の効果

35%の一般テキストデータを使用 vs なし (2400ステップ、同等のエージェントデータトークン)

結果：

- 一般テキストデータがわずかにエージェントパフォーマンスを改善
- 完全に異なる一般ドメインからのテキストデータがエージェントモデルのパフォーマンスを損なわない
- むしろパフォーマンス向上に寄与

理由: 一般テキストデータがエージェントモデルの汎化と命令理解を支援する可能性。

8. Reflective Long CoTがエラー訂正改善により大幅にパフォーマンスを向上

Qwen2-VL-7B、14K Win&Mac + 3K Ubuntu軌跡でアブレーション：

- Reflective Long CoTなし (Aguvis CoT) : 11.5%
- **Reflective Long CoT: 15.3%**

推測: 反省的推論はエラー訂正に焦点を当てており、改善は自己訂正能力の向上に起因。

制限事項

著者は永続的な課題を認識しています：

1. **新規シナリオでのタスク知識不足:** 全く新しいタスクでは知識不足
2. **複雑なUIでの高精度グラウンディングエラー:** 密集したUIでの正確な座標予測が困難
3. **アクション反復と終了誤判定:** 同じアクションの繰り返しや、タスク完了の誤判断
4. **長期タスクの失敗:** 多段階タスクで途中で失敗
5. **環境ロバスト性の制限:** 環境の微小な変動に敏感

今後の研究方向

論文から示唆される今後の研究方向：

1. **ポストトレーニング手法:** Pass@3で53.2%を達成することから、リランкиングやアンサンブル手法の余地
2. **長期タスクの計画:** 100ステップを超える超長期タスクの計画能力向上
3. **環境ロバスト性:** 初期条件の微小な変動への頑健性向上

4. エラー回復メカニズム: より良いエラー検出と回復戦略
5. マルチモーダル統合: テキストとビジョンのより効果的な統合

コード例とプロンプト

インストール

```
conda create -n opencua python=3.10
conda activate opencua
pip install -r requirement.txt
```

モデルのダウンロード

```
from huggingface_hub import snapshot_download

# OpenCUA-7Bをダウンロード
snapshot_download(
    repo_id="xlangai/OpenCUA-7B",
    local_dir="OpenCUA-7B"
)

# OpenCUA-32Bをダウンロード
snapshot_download(
    repo_id="xlangai/OpenCUA-32B",
    local_dir="OpenCUA-32B"
)

# OpenCUA-72Bをダウンロード
snapshot_download(
    repo_id="xlangai/OpenCUA-72B",
    local_dir="OpenCUA-72B"
)
```

モデルの使用

```
from transformers import AutoModelForCausalLM, AutoTokenizer
import torch

# モデルとトークナイザーの読み込み
model_path = "OpenCUA-7B"
model = AutoModelForCausalLM.from_pretrained(
    model_path,
    torch_dtype=torch.float16,
    device_map="auto"
)
tokenizer = AutoTokenizer.from_pretrained(model_path)
```

```

# タスク命令とスクリーンショット
task_instruction = "Open a new browser tab and search for 'machine
learning'"
# screenshots: PIL Image objectsのリスト（最大3枚の履歴）

# 推論 (L2 CoTフォーマット)
# モデルは "Thought: ... Action: ..." という形式で出力

```

アクション空間 (PyAutoGUI)

主要なアクションタイプ：

- `click(x, y, button='left')`: 座標(x, y)でクリック
- `type_text(text)`: テキストを入力
- `hotkey(*keys)`: ホットキーを実行 (例 : 'ctrl', 'c')
- `scroll(clicks, direction)`: スクロール
- `drag(start_x, start_y, end_x, end_y)`: ドラッグ
- `success()`: タスク成功で終了
- `fail()`: タスク失敗で終了

技術的詳細

アーキテクチャの変更点

標準的なQwen2-VL/Qwen2.5-VLアーキテクチャからの変更：

1. **M-RoPEなし**: 1D RoPEを使用
2. **Kimi-VLトークナイザー**: Kimi-VL-A3Bと同じチャットテンプレートとトークナイザーを採用
3. **標準transformersローディング推奨**: vLLMサポートは開発中

トレーニング詳細

OpenCUA-7B (Joint Training) :

- ベースモデル：Qwen2.5-VL-7B
- トレーニングデータ：200Bトークン
- データ混合：20%計画 + 20%グラウンディング + 60%一般
- バッチサイズ：512
- 学習率：最大 2.5×10^{-5} (最小 3×10^{-6})
- GPU：128 × A100
- 期間：8日間
- ベストチェックポイント：ステップ14,600

OpenCUA-32B (Stage 1 + Stage 2) :

- ベースモデル：Qwen2.5-VL-32B
- **Stage 1:**
 - データ：35Bトークン（一般テキスト、ビジョン、グラウンディング）
 - バッチサイズ：3,584
 - 学習率： 3×10^{-5}

- GPU : 224 × A100
- チェックポイント : ステップ1,200
- **Stage 2:**
 - データ : 60Bトークン (軌跡 + 一般 + グラウンディング)
 - 18K Win&macOS + 20K Ubuntu軌跡
 - バッチサイズ : 512
 - 学習率 : 2.5×10^{-5}
 - GPU : 128 × A100
 - 最終モデル : ステップ4,700

OpenCUA-72B (Stage 1 + Stage 2、データ増強) :

- ベースモデル : Qwen2.5-VL-72B
- アノテーション軌跡 + 8K o3+Jediロールアウト軌跡
- **Stage 1:**
 - データ : 250Bトークン
 - バッチサイズ : 600
 - 学習率 : $2.5 \times 10^{-5} \rightarrow 1.5 \times 10^{-5}$ に減衰
 - GPU : 480 × A100
- **Stage 2:**
 - データ : 16Bトークン
 - 効率的な情報密度の高いCoTフォーマットに変換
 - 学習率 : $1.5 \times 10^{-5} \rightarrow 2 \times 10^{-6}$ に減衰
 - GPU : 480 × A100

データセット統計

AgentNet Dataset:

- **総タスク数:** 22,625
 - Windows: 12,000
 - macOS: 5,000
 - Ubuntu: 5,000
- **平均ステップ数:** 18.6ステップ/タスク
- **解像度範囲:** 720p~4K
- **カバレッジ:**
 - 140以上のアプリケーション
 - 190以上のWebサイト
- **品質分布:**
 - Excellent: 高複雑度、マルチアプリワークフロー
 - Good: 適度な複雑度
 - OK: 基本的なタスク
 - Rejected: 品質基準未達

AgentNetBench:

- 100代表的タスク (Windows/macOS)
- ゴールドスタンダードアクション付き
- オフライン評価でオンラインメトリクスを近似

- オンラインベンチマークとの相関を検証済み

まとめ

OpenCUAは、コンピュータ使用エージェント研究における透明性のギャップに対処する包括的なオープンソースフレームワークです。主要な貢献は以下の通りです：

1. **AgentNet Tool**: クロスプラットフォーム、非侵入的なアノテーションインフラストラクチャ
2. **AgentNetデータセット**: 22,625タスク、3つのOS、200以上のアプリ/サイト
3. **Reflective Long CoT**: エラー検出と訂正を含む革新的な推論合成手法
4. **スケーラブルなトレーニングレシピ**: 3つの戦略 (Stage 2のみ、Stage 1+2、Joint Training)
5. **SOTA性能**: OpenCUA-72BがOSWorld-Verifiedで45.0% (オープンソース最高)
6. **完全な公開**: ツール、データセット、コード、ベンチマーク、モデルすべてをオープンソース化

OpenCUAは、明確なデータスケーリング法則とクロスドメイン汎化能力を実証し、オープンソースとプロプライエタリシステム間のギャップを縮小しました。すべてのコンポーネントを公開することで、コミュニティがこれらのエージェントの能力、制限、リスクを体系的に調査できるようになります。

参考リンク

- **論文**: <https://arxiv.org/abs/2508.09123>
- **GitHub**: <https://github.com/xlang-ai/OpenCUA>
- **プロジェクトページ**: <https://opencua.xlang.ai/>
- **OSWorld-Verified**: <https://os-world.github.io/>
- **Hugging Face Models**:
 - <https://huggingface.co/xlangai/OpenCUA-7B>
 - <https://huggingface.co/xlangai/OpenCUA-32B>
 - <https://huggingface.co/xlangai/OpenCUA-72B>

Sources

- [GitHub - xlang-ai/OpenCUA](#)
- [OpenCUA: Open Foundations for Computer-Use Agents](#)
- [Introducing OSWorld-Verified](#)