Learning to Act on Screens: VLM-RL for Real-World GUI Automation

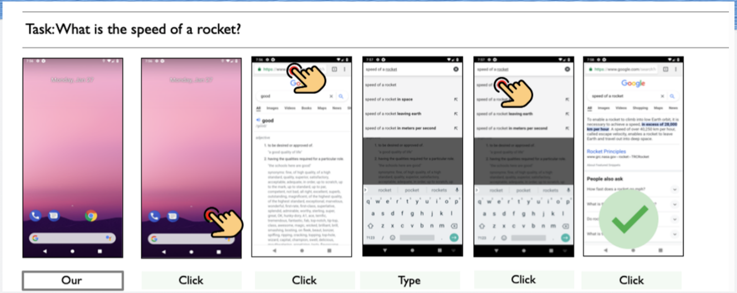
Fang Sun

Nov 7, 2025

Project Summary

- Led an end-to-end effort to train a **1.3B VLM** as a **general-purpose GUI agent**
- Built from **700,000 Android GUI interaction trajectories** (offline dataset)
- Used vLLM for high-throughput inference and DeepSpeed ZeRO-3 for memory-efficient fine-tuning
- Designed an **Offline-PPO** algorithm with stochastic-aware advantage estimation to handle GUI non-stationarity and interface variability
- Achieved **53.8% task success**, outperforming:
  - Supervised baseline: **+29.5% absolute**
  - GPT-4V: **8.3%**
  - CogAgent: **38.5%**

How the Agent Works: From Perception to Action



- Task: ”What is the speed of a rocket?”
- The agent observes the **entire screen** at each step.
- Selects actions sequentially: **Click** → **Click** → **Type** → **Click** → **Click**.
- ~~Learns **why** actions move the task~~ forward, not where buttons are.
- Enables **generalization** across UI layouts and devices.

Initiative: What Sparked This Project

- Many internal workflows (reporting, form submission, testing) relied on **rule-based GUI scripts**.
- These scripts were **fragile**: small UI layout changes caused repeated failures.
- Even advanced **VLMs (e.g., GPT-4V)** could **see** the interface but **could not decide the next action**.
- Early supervised fine-tuning attempts **overfit to pixel positions** instead of learning task strategies.
- I recognized the core challenge was not perception, but **decision-making under UI variability**.

Initiative: What I Did and The Immediate Impact

- I reframed GUI automation as a **sequential decision problem** and proposed using **reinforcement learning**.
- Designed a **safe offline RL pipeline** to avoid real-time interaction risks.
- Initiated and coordinated collection of **700k Android GUI interaction trajectories**.
- Led a staged training plan: **imitation learning** $\rightarrow$ **advantage-weighted offline RL**.
- **Direct Result:** The team shifted from **script-based automation** to a **learning-based agentic framework**, enabling models that **act**, not just **see**.

Innovation: Offline-PPO for GUI Action Learning

- We cannot safely or efficiently **explore online** in GUI environments:
  - Wrong clicks can lose progress, break workflows, or corrupt state
  - Real systems do not allow trial-and-error at high frequency
- We use an **advantage-weighted behavioral cloning** objective to learn action preferences:

$$\max_{\pi} \mathbb{E}_{(s,a)\sim D}\left[\exp\left(\frac{A(s,a)}{\beta}\right)\log \pi(a|s)\right] - \lambda \cdot \mathrm{KL}(\pi \parallel \pi_{\text{behavior}})$$

- Interpretation:
  - **Good actions** (positive advantage) get **higher probability**
  - **Bad / irrelevant actions** are **down-weighted**
  - KL regularization prevents the policy from drifting away from **human-like behavior**
- Result: We transform a passive VLM into a **policy that chooses actions**, without performing any online exploration.

Innovation: Stochastic-Aware Advantage Estimation

- GUI environments are **visually and functionally non-stationary**:
  - UI layout shifts with resolution or window scaling
  - Theme packs, OS versions, or app updates modify colors or widget shapes
  - Dynamic pop-ups introduce unexpected states
- Naive advantage estimation would **overfit** to specific screen appearances or trajectories.
- Our solution adds **stochastic smoothing and clipping**:
  - **Temporal smoothing** stabilizes advantages across similar states
  - **Clipping** prevents extremely large advantages from dominating learning
  - **State augmentation** increases tolerance to layout perturbation
- Effect:
  - The agent **generalizes across UI variations** rather than memorizing pixel layouts
  - The model becomes **robust to unseen screen states** and dynamic interface changes

Implementation Process & Collaboration

- **I initiated and led the project end-to-end**, from problem framing to prototypes to full training pipeline.
- **Solo-led core development**:
  - Built dataset pipeline (collection, filtering, trajectory formatting)
  - Implemented imitation learning and Offline-PPO training loop
  - Designed action-token interface and runtime execution layer
- **Collaborative inputs where needed**:
  - Advisor feedback on algorithmic strategy
  - Occasional engineering support for environment instrumentation
- Result: A **fully functional, reproducible pipeline** for training agentic GUI VLMs.

Implementation: Dataset Construction

- Collected **human demonstration trajectories** from real GUI workflows
- Each step contains:
  - Screenshot state s_t
  - Task instruction (goal)
  - Action token a_t (click / drag / type / confirm)
  - Reward signal r_t representing task progress
- Stored as transitions:

$$(s_t, a_t, r_t, s_{t+1})$$

- Filtered out:
  - Idle cursor motion
  - Accidental clicks
  - Dead-end trajectories
- Result: **Clean dataset reflecting intentional expert behavior**.

Implementation: Imitation Warm-Start & Offline RL Refinement

Step 1: Supervised Imitation (Behavior Cloning)

- Model learns to **imitate expert actions** directly from demonstrations
- Prevents unstable "random exploration" behavior

Step 2: Offline-PPO Policy Refinement

$$\max_{\pi} \mathbb{E}_{(s,a)\sim D}\left[\exp\left(\frac{A(s,a)}{\beta}\right)\log \pi(a|s)\right] - \lambda \, \mathrm{KL}(\pi \parallel \pi_{\text{behavior}})$$

- **Increase** probability of actions that advance task progress
- **Reduce** probability of suboptimal actions
- **Constrain policy** near demonstrated behavior for stability & safety

Implementation: Action Execution Runtime & Training Infrastructure

Action Execution Layer

- Converts predicted tokens to executable actions:
- `CLICK(x,y)`, `SELECT`, `TYPE("text")`, `CONFIRM`
- Decouples policy learning from UI rendering to **portable across GUIs**

Training Infrastructure

- vLLM: High-throughput inference during evaluation loops
- DeepSpeed ZeRO-3: Sharded training for the 1.3B parameter policy model

Outcome

The system trains efficiently on **commodity multi-GPU hardware** and outputs a **robust, multi-step GUI automation agent**.

Insights: Why Previous Approaches Struggled

- **Visual Non-Stationarity is the Core Challenge**
  - UI layouts change over time (position, size, color, pop-ups)
  - Supervised VLMs **memorize** training layouts $\rightarrow$ fail when screens shift
- **The Problem Was Not Recognition — It Was Decision Adaptation**
  - Models could see elements correctly
  - They did not know what to do next when context changed
- **Key Insight:** GUI automation must be framed as **policy decision-making**, not screen captioning.

Insights: Limitations of Imitation Learning

- **Imitation Learning Reproduces Patterns, Not Strategy**
  - Behavior cloning copies human actions seen during training
  - Breaks down in new or unseen states
- GUI tasks require **credit assignment across multiple steps**
  - E.g., clicking a menu now enables success several steps later
- **Offline Reinforcement Learning was required** to learn:
  - **Why** actions matter (not just what they look like)
  - How to recover from unfamiliar or shifted UI states
- **Key Insight:** GUI automation requires **policy reasoning, not action mimicry**.

Insights: What Enabled Generalization

- **Advantage-Weighted Offline PPO Improved Policy Quality**
  - Increased probability of actions that reliably advanced task progress
  - Worked entirely offline — no risky or costly real-time exploration
- **Decoupling Actions from Pixel Coordinates Was Critical**
  - Represented actions as semantic UI intents (e.g., “confirm”, “open menu”)
  - Dramatically reduced brittleness across applications and screen layouts
- **Key Insight:** Stable generalization comes from

$$\textbf{(offline credit assignment)} + \textbf{(semantic action abstraction)}.$$

Iteration: Refining the Approach

- **Policy Drift During Offline RL**
  - Early policies deviated from human behavior to unstable “random clicking”
  - Solution: **KL regularization** + **advantage normalization**
  - Result: Stable updates that stay close to expert intent
- **Ambiguity in Multi-Element Screens**
  - Model misclicked when multiple similar buttons appeared (e.g., multiple “Next” buttons)
  - Solution: **Semantic action abstraction** + goal-grounded UI prompts
  - Result: Better element selection precision
- **Overfitting to Frequent UI Layouts**
  - Model performed well on common states, poorly on rare or unseen screens
  - Solution: **UI theme / resolution randomization** + synthetic pop-ups
  - Result: Improved generalization across interface variation

Future Improvements: Increasing Velocity & Efficiency

- More Systematic Evaluation Signals
  - Added mid-trajectory checkpoints to diagnose where reasoning failed
  - Going forward: Integrate **step-level success scores** automatically during data logging
- Smarter Dataset Curation
  - Next iteration would **actively mine failure states** for targeted offline fine-tuning
  - Reduces training time by focusing on high-impact cases
- Faster Iteration Loops
  - Early loops required full training runs to observe behavior changes
  - Would adopt **agent replay + on-device lightweight rollouts** for faster debugging
- Scalable Action Abstraction Library
  - Future improvement: A reusable **semantic UI action vocabulary** across apps
  - Reduces retraining effort when transferring to new GUI environments

Impact

Quantitative Results

Model	Success
GPT-4V	8.3%
BC (Supervised VLM)	24.3%
CogAgent	38.5%
Ours (Offline-RL VLM)	**53.8%**

- **+29.5% absolute** over supervised
- **5–6×** higher than GPT-4V prompting
- **1.4×** over state-of-the-art agent

Qualitative Outcomes

- Generalizes to **unseen** UI layouts
- Robust to **pop-ups, themes, resizing**
- Previously manual workflows ■ **fully automated**

Key Insight

- GUI automation is a **policy decision problem**, not just visual recognition
- Offline RL enables safe and scalable improvement **without online trials**

Thank You for the opportunity!