Interpreting Vision Grounding in Vision-Language Models: A Case Study in Coordinate Prediction

Clement Neo
Nanyang Technological University
Digital Trust Centre, Singapore

Yongsen Zheng†
Nanyang Technological University
Digital Trust Centre, Singapore

Kwok-Yan Lam
Nanyang Technological University
Digital Trust Centre, Singapore

Luke Ong
Nanyang Technological University

Abstract

Vision-language models increasingly power autonomous agents that require precise spatial actions, from computer-use agents clicking interface elements to robots grasping objects. We present the first mechanistic analysis of computer-use models, using UI-TARS 1.5 on a controlled task where models must click colored squares in grid images. We discover a systematic failure mode where the model misclicks approximately 50% of the time, often targeting locations exactly one patch below the correct target despite high confidence. Through activation patching, layer-wise analysis, and coordinate probing, we reveal that failures stem from biased late-layer selection: the model simultaneously maintains accurate representations of both correct and incorrect locations yet systematically outputs wrong coordinates. Our analysis identifies strong patching effects at specific token positions in the final layers, with probes successfully detecting the systematic downward bias. Our work establishes coordinate prediction as a tractable testbed for multimodal interpretability and provides insights for improving spatial grounding reliability in deployed vision-language agents.

1 Introduction

Vision-language models (VLMs) are rapidly being deployed as autonomous agents in high-stakes applications where spatial precision matters. Computer-use agents execute financial transactions by clicking specific interface elements, and robotic systems perform delicate manipulations by grasping particular objects [Brohan et al., 2023]. However, the reliability and mechanism of coordinate prediction in these models remains poorly understood, despite having direct implications for deployed autonomous systems. Computer-use agents that misclick interface elements can cause significant errors, while robotic systems with poor spatial grounding pose safety risks.

Hence, we study the task of coordinate prediction in a state-of-the-art computer-use model, UI-TARS 1.5 [Qin et al., 2025], under a controlled synthetic setting: clicking on a colored square with a plain background. The image input is a 476×476 image partitioned into a 17×17 grid of 28px×28px patches containing a single colored square on a uniform background, with the text instruction being "Click on the <COLOR> square." (where COLOR is the corresponding color of the square). On this dataset of 1,445 images spanning all grid locations and five colors, the model misclicks approximately

†Corresponding author: yongsen.zheng@ntu.edu.sg.

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Mechanistic Interpretability Workshop.

50% of the time, with errors concentrated at exactly one patch below the correct target and with high token-level confidence.

To examine where this error arises, we use three techniques: (1) layer-wise token probabilities at positions immediately before each coordinate digit, also known as the logit lens, (2) activation patching of residual stream states across layers and token positions, and (3) linear probes trained on hidden states to recover ground-truth and predicted coordinates. Through our analysis, we find that the model encodes accurate ground-truth coordinates in middle layers but fails to decode them correctly, coordinate predictions become highly confident in final layers despite systematic errors, and activation patching reveals a two-stage computational process with late-layer transformations at the `<|box_start|>` token.

2 Background

Vision-Language Model Architecture Vision-language models follow a standard three-component architecture. A vision encoder divides images into spatial patches and extracts features $F \in \mathbb{R}^{N \times d_{vis}}$, a cross-modal adapter A transforms these into visual token embeddings $V = A(F) \in \mathbb{R}^{N \times d_{LM}}$, and an autoregressive model processes combined visual and text tokens to generate outputs $y = \text{LM}([V_S; T_q])$.

Computer-Use Agents We study UI-TARS 1.5 7B, a state-of-the-art computer-use model. The model is essentially a vision-language model trained to call actions like "click(x,y)" or "type('abc')" based on screenshots. Given an instruction like "click on the blue square", UI-TARS analyzes the visual input, often reasoning about what it observes, then generates specific action commands such as "click(154,154)." (Refer to Figure 1a for an illustration.)

Concretely, UI-TARS emits actions as function calls like `click(x,y)` under a fixed grammar:

```
Thought: <MODEL_RATIONALE>
Action: click(start_box='<|box_start|>(154, 154)<|box_end|>')<|im_end|>
```

where immediately preceding the first digit, both `<|box_start|>` and `(` are single tokens. The numerical digits are also tokenized as individual tokens.

Interpretability Challenges in Multimodal Models Mechanistic interpretability of vision-language models faces two key challenges compared to text-only models. First, visual information exhibits extreme redundancy across tokens. Visual objects are typically represented in a distributed manner across multiple image patches, with semantic content redundantly encoded in many tokens. This redundancy is so robust that models can maintain performance even when visual tokens are randomly scrambled [Qi et al., 2025], making it difficult to identify which specific tokens are crucial for particular predictions.

Second, vision-language models process sequences with hundreds or thousands of tokens per image, far exceeding typical text sequence lengths. A single screenshot might be divided into 576 patches, each becoming a separate token, creating sequences much longer than most language model inputs. This scale makes comprehensive interpretability analysis computationally prohibitive [Neo et al., 2025].

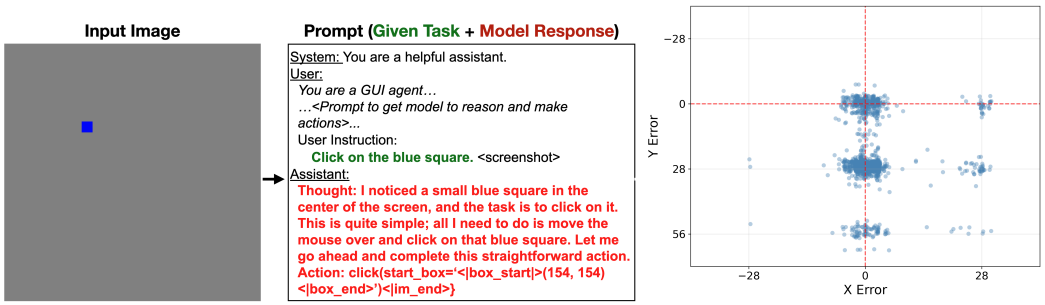
Computer-use tasks offer a more tractable setting because coordinate prediction requires committing to exact pixel locations rather than approximate descriptions. This constraint forces clearer connections between visual patches and numerical outputs, providing more direct pathways for mechanistic analysis.

Hence, we investigate the mechanisms underlying coordinate prediction in vision-language models: How does a model process visual information to determine where to click?

3 Experimental Setup

We design a synthetic task to address the interpretability challenges outlined above. Our setup uses 476×476 pixel images divided into a 17×17 grid of 28×28 pixel patches, with each image containing

a single colored square positioned at a specific grid location against a uniform gray background. This ensures that the colored square occupies exactly one visual token. The model is then prompted to ‘‘`Click on the {COLOR} square`’’. Refer to Figure 1a below for an illustration.



(a) An overview of the image and task

(b) Click Error Distributions

Figure 1: In our experimental setup, UI-TARS 1.5 fails ≈50% of the time on our synthetic grid task, with systematic bias toward clicking one patch below the target (negative y-errors).

We generate a dataset of 1,445 images spanning 5 colors (red, blue, green, yellow, orange) across all 17×17 grid positions, with a 80/20 train/test split for probe training.

Model Evaluation. We then evaluate UI-TARS 1.5 7B on this dataset. We use deterministic decoding (T=0) and count the prediction as correct if and only if the predicted coordinate lands in the square. To compare against the grid, each target cell (i, j) is mapped to its pixel-center $(c_x(i), c_y(j)) = (14 + 28i,\ 14 + 28j)$, and a prediction is counted correct if and only if its predicted coordinate lands in that cell, i.e., $|x - c_x(i)| \leq 14$ and $|y - c_y(j)| \leq 14$.

In this setup, we find that the model gets the coordinates wrong more than 50% of the time. The clicks are clustered around the center of patches, and that it is often clicking the patch directly below the square (Figure 1b). In particular, across all 1,445 examples, it clicks on the correct patch in 413 examples (28.6%), one patch down in 837 examples (57.9%), and two patches down in 76 examples (6%).

This raises the following questions: How does the model compute the (x, y) to be clicked, and can we trace where the model gets it wrong? In the next section, we attempt to investigate this using per-layer probability trajectories, activation patching, and probing.

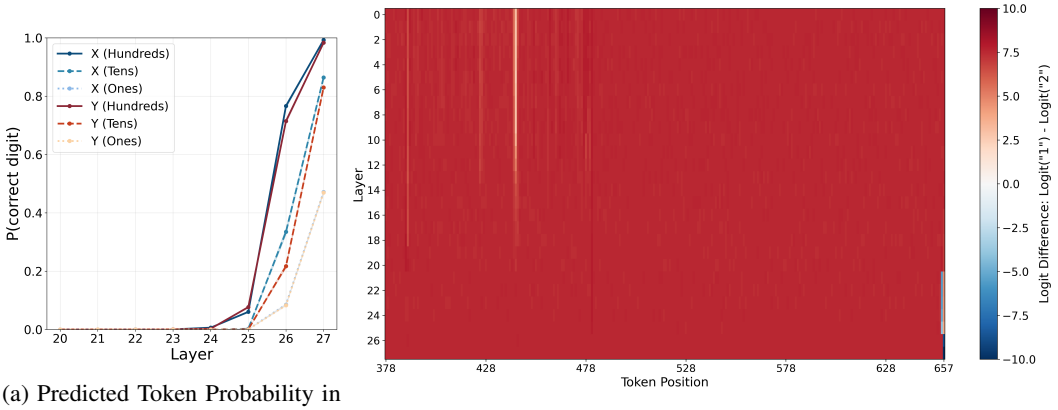
4 Mechanistic Interpretability Experiments

In this section, we use several interpretability techniques to investigate how the model performs coordinate prediction.

Token Probability over Layers. How confident is the model when it predicts incorrect coordinates? To investigate this, we analyze how the probability of predicting coordinate digits evolves across the model's layers. We conduct forward passes on our synthetic grid dataset and extract hidden states at each layer. At each position before a coordinate digit, we apply layer normalization and the language modeling head to the hidden state immediately before each coordinate digit to compute the probability of the next predicted digit. We do this across our 5-color dataset.

Figure 2a shows how the probability of the predicted digit consistently spikes to high confidence in the final two layers, even when the predictions are wrong. This points to a follow-up question: at which layers is the target patch encoded, and how is it rendered into coordinate tokens?

Activation Patching. Given the model's high confidence in incorrect coordinates, we next investigate where in the model these coordinates are computed. We use activation patching [Vig et al., 2020] to identify which components causally influence coordinate prediction. We generate two images: a source with the blue square at $(154, 154)$ (first digit "1") and a target at $(238, 238)$ (first digit "2"). During a forward pass on the source image, we systematically replace activations at each residual



(a) Predicted Token Probability in the Last Layers

(b) Activation Patching Heatmap

Figure 2: We find that token probabilities spike to high confidence in final layers despite incorrect predictions, and that patching effects concentrate at <|box_start|> in late layers.

stream position with corresponding activations from the target image, measuring the change in logit difference Logit("1") − Logit("2").

Our results are in Figure 2b. First, we observe strong patching effects at the <|box_start|> token in layers 21-25, suggesting this position stores coordinate information. Second, patching effects peak at the final "(" token in the last two layers, though this is expected since the model directly decodes coordinates from this position. Combined with our earlier finding that coordinate probabilities spike in these same final layers, this may suggest a two-stage process: the model computes and stores coordinates at <|box_start|>, then retrieves this information when generating outputs. If so, we should be able to extract full coordinate information from the <|box_start|> representations using probes.

Training Probes To test our hypothesis that coordinates are stored at <|box_start|>, we train linear probes to extract (x, y) coordinates from hidden states from our 5-color dataset at both <|box_start|> and (token positions across all layers. Each probe predicts the normalized coordinates $(x/500, y/500)$, with hyperparameters selected via grid search (Appendix A). We train three types of probes: (1) ground truth coordinates (X_{gt}, Y_{gt}), (2) the model's predicted coordinates (X_{pred}, Y_{pred}), and (3) the binary class of if the model will click one patch below target. We report Mean Average Error for the coordinates probe, and balanced accuracy for the binary class probe.

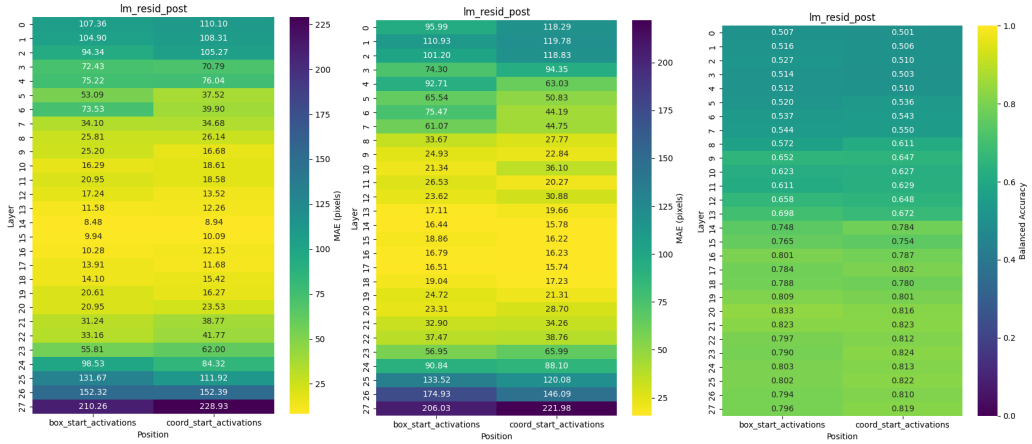
Our findings are in Figure 3. First, we find that linear probes accurately recover ground truth coordinates (MAE < 10 pixels) around layer 14, but achieve only 16-20 pixel MAE for predicted coordinates, which is beyond the 14-pixel threshold for correct patch selection.

Second, probe performance degrades in layers 20-27, where activation patching shows strong effects. This may suggest that the activations are transformed from from generic coordinate representations into task-specific representations.

Lastly, the probes can detect the "one patch below" error with 80% balanced accuracy in late layers. This suggests that the downward error is encoded in the model's representations.

5 Discussion & Conclusion

The results are surprising: The middle layers carry an accurate estimate of the target location, yet the final decoding stages output the wrong coordinates. We see this because linear probes at <|box_start|> recover ground-truth coordinates with lower error than they recover the model's own output coordinates, yet token probabilities surge to high confidence in the last one or two layers even when the digits are wrong. Furthermore, activation patching shows that the largest causal effect at <|box_start|> appears in late layers (e.g., 21–25) after the layers where the probe reads out ground-truth coordinates best (roughly 12–16).



(a) MAE (Y_{gt}) (b) MAE (Y_{pred}) (c) Balanced Acc. (Error)

Figure 3: The model simultaneously encodes both correct (a) and predicted (b) coordinates, and as well as the systematic "one-patch-below" error (c).

We consider several hypotheses for how this failure arises. The model may employ two-stage computation where middle layers (12-16) form accurate coordinate representations that later undergo a corrupting transformation at <|box_start|> in layers 20-25, where we observe strong patching effects.

A possible non-mechanistic explanation for this behavior is action-grounding priors from pre-training. Computer-use models learn on real UIs where clickable elements exhibit systematic spatial relationships—buttons in vertical lists, downward-stacking menus, and text baseline alignment causing clicks to statistically fall below visual centers. The systematic downward error might reflect a learned prior that "clickable elements appear below salient features," confidently applied even when inappropriate for our synthetic task. This would explain both the high confidence and the specific directional bias.

We hope to continue building on this work to better understand how the mid-layer coordinate state is transformed into a task-specific state, and to pinpoint how the model ends up choosing the wrong patch to click on.

Lastly, while our synthetic grid task is simplified, we believe that it establishes computer-use models as a tractable testbed for multimodal interpretability research. Our finding that models can simultaneously represent both correct and incorrect spatial information opens new questions about how these representations compete during inference. As computer-use agents become more prevalent, understanding these basic failure modes provides a foundation for improving reliability in deployed systems.

Open Questions and Future Work Why does the model select biased coordinates despite accurate representations? Is the downward error learned or architectural? Future work should test if such failures persist in realistic interfaces, and if so, solve such failures without harming performance.

References

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alex Herzog, Jasmine Hsu, Brian Ichter, Alex Irpan, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Lisa Lee, Tsang-Wei Edward Lee, Sergey Levine, Yao Lu, Henryk Michalewski, Igor Mordatch, Karl Pertsch, Kanishka Rao, Krista Reymann, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Pierre Sermanet, Jaspiar Singh, Anikait Singh, Radu Soricut, Huong Tran, Vincent Vanhoucke, Quan Vuong, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Jialin Wu, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *arXiv preprint arXiv:2307.15818*, 2023.

Clement Neo, Luke Ong, Philip Torr, Mor Geva, David Krueger, and Fazl Barez. Towards interpreting visual information processing in vision-language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=chanJGoa7f.

Jianing Qi, Jiawei Liu, Hao Tang, and Zhigang Zhu. Beyond semantics: Rediscovering spatial awareness in vision-language models, 2025. URL https://arxiv.org/abs/2503.17349.

Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, Wanjun Zhong, Kuanye Li, Jiale Yang, Yu Miao, Woyu Lin, Longxiang Liu, Xu Jiang, Qianli Ma, Jingyu Li, Xiaojun Xiao, Kai Cai, Chuang Li, Yaowei Zheng, Chaolin Jin, Chen Li, Xiao Zhou, Minchao Wang, Haoli Chen, Zhaojian Li, Haihua Yang, Haifeng Liu, Feng Lin, Tao Peng, Xin Liu, and Guang Shi. Ui-tars: Pioneering automated gui interaction with native agents, 2025. URL https://arxiv.org/abs/2501.12326.

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Simas Sakenis, Jason Huang, Yaron Singer, and Stuart Shieber. Causal mediation analysis for interpreting neural nlp: The case of gender bias. *arXiv preprint arXiv:2004.12265*, 2020.

A Probe Hyperparameters

We extract representations from both <|box_start|> and (token positions across all layers. Each probe uses a dual-output architecture to simultaneously predict normalized coordinates $(x/500, y/500)$. We do a train/test split of 80/20 and report test results in the main body. We train for up to 100 epochs with early stopping (patience=15), employing grid search over learning rates $(10^{-4}, 10^{-3}, 10^{-2})$, batch sizes (32, 64, 128), and for MLP probes, hidden dimensions (256, 512, 1024). Performance is evaluated using mean absolute error (MAE) on held-out data. We then report our results on the best performing probe. We also note that we trained MLP probes in addition to linear probes, but did not find substantially better results.