Vision UI: Desktop GUI Grounding for Evidence-First User Activity Narration

Jared Young
Machine Learning and Deep Learning Final Project

***Abstract*—Vision UI is a local-first machine learning systems project for transforming desktop screenshot streams and native operating-system context into evidence-linked accounts of user activity. The project began as a literature-driven survey of GUI grounding and action understanding, then implemented the central conclusion of that survey: faithful narration depends first on grounding user actions to interface elements that were actually visible on screen. This report presents the resulting system as an evidence-first pipeline that captures screenshots, records native context when available, extracts OCR and heuristic interface proposals, applies optional learned grounding components, serializes per-frame JSON artifacts, and produces deterministic session summaries plus qualitative human-readable narratives. The main quantitative contribution is the learned GUI grounding progression; narration is systems-oriented and demonstrated qualitatively through saved evidence artifacts. The benchmark story is a historical progression rather than a single replacement result. The best crop-level model remains a MobileNetV3-Large role-family classifier with `0.7767` raw accuracy, `0.7533` thresholded accuracy, and `0.6221` thresholded macro-F1. Grounding top-1 accuracy improved from the heuristic combined baseline of `0.1643`, which detector-only matched, to classifier-only `0.1956`, detector-fusion `0.1978`, v3 reranker-shadow `0.3418`, transformer listwise `0.3424`, geometry calibrated MLP `0.3509`, click-visual union MLP `0.3620`, raw hybrid gate `0.3817`, hybrid plus visual-candidate gate `0.3939`, recall-push visual-candidate reranker v2 `0.3966`, and finally the five-predictor recall-push hybrid gate `0.4450`. The best result reaches `0.4533` on held-out UI-Vision, `0.4434` on ScreenSpot-Pro, and selected-predictor recall `0.6906` over the combined `1,881`-example evaluation. On held-out UI-Vision, detector-only grounding remained at `0.1000`, matching the heuristic baseline with `0.0000` improvement. The report therefore argues that Vision UI is a working local-first evidence pipeline with deterministic narration demonstrated qualitatively, while the learned grounding stack is a meaningful experimental progression whose best ensemble remains offline until harmful switches and high-confidence wrong pockets are reduced.**

***Index Terms*—GUI grounding, desktop understanding, screenshot streams, evidence-first narration, computer-use agents, multimodal learning**

I. Introduction

Modern desktop activity leaves behind many partial traces: screenshots, window titles, click coordinates, OCR text, clipboard events, shell commands, and application logs. None of those traces is, by itself, a reliable account of what the user did. A screenshot archive shows what was visible but not which target was intended. A click log records coordinates but not the semantic element under the pointer. A shell history may preserve a command but not the visual workflow that led to it. For troubleshooting, incident reconstruction, accessibility review, and personal memory systems, the useful artifact is a narrative that remains tied to inspectable evidence.

Vision UI addresses this problem by treating desktop activity understanding as an evidence-first grounding and narration task. The system captures screenshots and available native context, transforms each frame into standalone JSON evidence, and builds higher-level steps and session narratives only after the underlying evidence has been saved. This design deliberately separates the truth layer from the prose layer. The narrator is not asked to invent a story from pixels alone; it consumes a sequence of OCR spans, GUI element proposals, window metadata, input observations, reconstructed commands, facts, episodes, and confidence markers.

The central technical problem is GUI grounding. Given a frame S_t, native context C_t, and an interaction such as a click or focus event, the system must decide which visible interface element is the target. The broader Vision UI task can be written as

$$(S_{1:T}, C_{1:T}) \rightarrow (E_{1:T}, A_{1:T}, N), \tag{1}$$

where $E_{1:T}$ is grounded evidence, $A_{1:T}$ is a sequence of user-level actions, and N is a natural-language narrative. In practice, the quality of $A_{1:T}$ and N depends heavily on the quality of $E_{1:T}$. If a click is attributed to a neighboring pane, an oversized panel, or the wrong row of a list, the downstream action text becomes misleading even when the sentence is fluent.

This report continues the earlier Vision UI topic and dataset survey. That survey selected five papers as the research basis for the project: SeeClick, UI-Vision, DeskVision, ScreenSpot-Pro, and Sharingan [1], [2], [3], [4], [5]. Together, those papers showed that GUI grounding remains a bottleneck, that desktop-specific supervised data now exists, that professional high-resolution interfaces are still difficult, and that temporal narration depends on strong frame-level evidence. The final repo implements this argument as a local-first pipeline with an experimental grounding stack rather than as a monolithic end-to-end agent.

The report makes four contributions. First, it synthesizes the five-paper survey into a concrete desktop activity understanding design. Second, it documents a working local-first system that preserves per-frame JSON evidence, supports batch

and live workflows, and keeps deterministic narration as the authoritative default. Third, it evaluates a learned grounding progression composed of heuristic proposals, a MobileNetV3-Large role-family classifier, detector-fusion ablations, click-conditioned proposals, MLP and transformer rerankers, and hybrid ensemble gates [6], [7], [8], [9], [10], [11], [12]. Fourth, it reports both positive and negative results: reranking and ensemble selection materially improve top-1 grounding, but detector-only grounding still fails the runtime gate and the best hybrid remains an offline experimental profile. The main quantitative contribution is the learned GUI grounding progression; the narration contribution is systems-oriented and evaluated qualitatively through saved evidence artifacts.

II. Survey

The literature most relevant to Vision UI sits between three areas: single-screen GUI understanding, visual GUI grounding, and temporal action reconstruction. Screen understanding work on widget captioning, screen summarization, screenshot question answering, pixel-level UI semantics, screenshot parsing, and UI-specialized vision-language modeling motivates the need for a strong per-frame semantic layer [13], [14], [15], [16], [17], [18]. GUI grounding and desktop perception work then narrows the problem from generic captioning to locating actionable interface targets [1], [19], [2], [3], [4]. Finally, UI action-sequence, screencast, bug-recording, and dense-video-captioning work motivates the downstream goal of reconstructing user activity over time [20], [21], [22], [23], [5], [24], [25], [26], [27]. However, Vision UI's specific bottleneck is not general captioning. It is deciding what element a user interacted with on dense desktop screens and then preserving that decision as evidence for downstream narration. The final project therefore continues the initial survey rather than replacing it: the same five anchor papers define the problem, dataset strategy, and evaluation logic.

A. SeeClick: Grounding as the Bottleneck

SeeClick is the conceptual anchor because it frames GUI grounding as a core limitation of visual GUI agents rather than as a solved preprocessing step [1]. The paper's key lesson for Vision UI is that fluent language generation does not compensate for weak localization. If the system cannot ground an instruction or event to the correct visual target, later action descriptions will be unreliable. SeeClick also supports Vision UI's local-first orientation because it emphasizes visual grounding without requiring HTML or accessibility metadata to be present.

The strength of SeeClick is its clear problem framing. It makes a persuasive case that screenshot-based GUI agents require precise target grounding before they can be trusted to act. Its limitation for this project is scope: the benchmark context is broader than desktop-only professional workflows, and the goal is agent action rather than evidence-preserving narration. Vision UI uses SeeClick primarily as the reason to prioritize click-to-element attribution before narrative generation.

B. UI-Vision: Desktop Supervision

UI-Vision is the closest supervised benchmark match for this project [2]. It is desktop-centric, spans 83 applications across six domains, includes roughly 450 demonstrations and about 8.2k query-label pairs, and evaluates perception and interaction tasks such as element grounding, layout grounding, and action prediction. Its reported difficulty is important: the surveyed result notes best tested scores of only 25.5% element grounding accuracy, 30.8 IoU layout grounding, and 19.7% click prediction recall. These values make desktop GUI grounding a credible course-project target because the task is useful, hard, and not already saturated.

The strength of UI-Vision is its alignment with Vision UI's operating point: real desktop applications, dense screenshots, action trajectories, and human labels. Its limitation is that it remains an offline benchmark rather than a local evidence system. It does not solve screenshot capture, OS context collection, privacy-preserving raw evidence, deterministic narration, or user-facing inspection. Vision UI uses UI-Vision as the primary supervised source while adding the surrounding local system that the benchmark does not provide.

C. DeskVision: Region Semantics at Scale

DeskVision contributes a different ingredient: scale [3]. Its 54,855 desktop screens and 303,622 region annotations show that desktop region semantics can be collected at a broader scale than a small local project could manually label. For Vision UI, DeskVision is most valuable as an auxiliary or future weak-supervision source. Region captions are not the same as click-aligned target labels, but they can teach reusable priors about buttons, document regions, lists, input fields, and panels.

DeskVision's strength is breadth across operating systems and applications. Its limitation is that caption-like region supervision is weaker than direct click-target supervision for the final task. In the implemented repo, DeskVision remains a survey-supported future scaling path rather than the source of the final metrics. The results still connect to DeskVision because several later experiments show exactly why proposal priors matter: increasing candidate recall can help, but only when the ranker or gate can handle the added ambiguity.

D. ScreenSpot-Pro: Professional Stress Testing

ScreenSpot-Pro provides the stress-test perspective [4]. It contains 1,581 expert-annotated instructions from 23 applications, five industries, and three operating systems. The average target occupies only 0.07% of the screenshot area, which makes it an appropriate external check for whether a model can survive dense professional interfaces. Vision UI uses ScreenSpot-Pro in that spirit: not as an easy benchmark, but as the external full-run evaluation beside held-out UI-Vision.

The strength of ScreenSpot-Pro is authenticity. Its examples resemble the dense, high-resolution desktops that make local troubleshooting and security review difficult. Its limitation is that it is evaluation-oriented and relatively small for training. This makes it a poor source for fitting many local project

decisions, but an excellent check against overfitting to UI-Vision.

E. Sharingan: From Frames to Action Sequences

Sharingan connects frame-level grounding to the downstream goal of desktop action narration [5]. It extracts user action sequences from desktop recordings and reports strong operation-type matching with VLM-based approaches on ACT ONE and ACT REAL. Its relevance is not that Vision UI copies Sharingan's architecture. Rather, Sharingan supports the project thesis that temporal explanation is fragile unless the underlying frame evidence is reliable.

The strength of Sharingan is that it moves beyond single screenshots and asks whether actions can be recovered as sequences. Its limitation for Vision UI is that it is centered on video-level extraction rather than evidence-first JSON artifacts, local privacy boundaries, or deterministic target attribution. Vision UI borrows the downstream motivation: better grounding should eventually improve operation recovery and narrative faithfulness, although this report does not directly measure that downstream effect.

F. Survey Synthesis

The survey supports three design conclusions. First, grounding is the most defensible learning target. The repo already has capture, serialization, deterministic narration, and local GUI review; the most damaging remaining errors are target attribution errors. Second, desktop-specific data is sufficient for a meaningful project. UI-Vision gives a primary supervised source, while ScreenSpot-Pro gives a harder external benchmark and DeskVision gives a credible future scaling route. Third, none of the surveyed works fully matches the Vision UI system goal. SeeClick is an agent benchmark, UI-Vision is an offline benchmark, DeskVision is a region-captioning dataset, ScreenSpot-Pro is an evaluation suite, and Sharingan is a recording-to-action method. Vision UI's niche is the combination of desktop grounding, native OS context, deterministic local artifacts, privacy-aware evidence handling, and session narration.

III. Dataset

The final implementation uses UI-Vision as the annotated dataset for training and held-out evaluation. ScreenSpot-Pro is the external validation and stress-test benchmark for professional high-resolution GUI grounding. The benchmark claims in this report therefore come from UI-Vision, ScreenSpot-Pro, and the saved metrics artifacts produced by the implementation.

A. Implemented Data Sources

Table II summarizes the data actually used for the final report. UI-Vision supplied the primary annotated examples used to train and evaluate the role-family classifier and grounding stack. ScreenSpot-Pro supplied the main external validation set, allowing the project to test whether improvements observed on UI-Vision transferred to denser professional interfaces. The presentation demo run is a prerecorded local capture collected through the Vision UI GUI application and replayed in the notebook. It is not a generated dataset or benchmark; it is a qualitative artifact used to show that the local capture, analysis, evidence JSON, GUI inspection, and narration workflow operates end to end.

B. Evaluation Scope

The held-out UI-Vision evaluation contains `300` examples. ScreenSpot-Pro contributes `1,581` examples. The combined full run therefore contains `1,881` grounding examples. The training pool contains `5,189` UI-Vision-derived examples for the role-family classifier and learned grounding experiments. The label space is deliberately small and portable: `input`, `button`, `list_item`, `document_content`, and `other`. This label design follows the survey thesis. It is broad enough to transfer across desktop applications, but specific enough to improve action semantics inside the deterministic Vision UI pipeline.

IV. Methodology

Vision UI is built around a simple constraint: every per-frame analysis must be serializable to standalone JSON, and session narration must consume those serialized events rather than hidden in-memory state. This rule makes the system inspectable. It also prevents learned components from silently replacing the evidence layer.

A. System Pipeline

The pipeline has six stages. First, the capture layer records screenshots and native context such as active window metadata, click coordinates, pointer position, and platform-specific focus hints when available. Second, OCR, implemented through the Tesseract engine in the local pipeline, and visual heuristics produce text spans and interface-region proposals [28]. Third, the optional learned stack adds role-family predictions, detector candidates, click-conditioned proposals, reranker scores, and hybrid-gate selections. Fourth, the frame analyzer serializes the combined observations, including provenance and confidence fields, into JSON. Fifth, a session layer reconstructs higher-level facts, commands, security findings, episodes, and joined user steps. Sixth, deterministic narration and the local GUI consume the saved artifacts.

The architecture is local-first by design. OCR may be missing, active-window APIs may differ by operating system, and global click capture may require permissions. Therefore, the system degrades gracefully: missing providers reduce confidence and detail, but do not prevent JSON output or deterministic narration. The learned stack follows the same principle. Model outputs are additive fields, not schema-breaking replacements.

B. Evaluation Protocol and Model Selection

All quantitative evaluation in this report is scoped to crop classification, click grounding, and predictor-selection diagnostics. Top-1 grounding accuracy counts an example correct

TABLE I
COMPARISON OF THE FIVE ANCHOR WORKS FROM THE VISION UI SURVEY. REPORTED METRICS ARE PAPER-SPECIFIC AND SHOULD NOT BE INTERPRETED AS A SINGLE LEADERBOARD.

Work	Main focus	Headline reported result	Strengths	Limitations	Direct value for Vision UI
SeeClick [1]	Screenshot-only GUI agent with grounding pretraining	53.4% average on ScreenSpot; better grounding improves downstream task performance	Establishes grounding as the critical visual GUI bottleneck; works without HTML or accessibility trees	Not desktop-only; benchmark is broader and simpler than professional desktop use	Justifies why Vision UI should invest in grounding before narration
UI-Vision [2]	Offline desktop benchmark for perception and interaction	Best reported scores are still low: 25.5% element grounding accuracy, 30.8 IoU layout grounding, 19.7% click recall	Best human-annotated desktop benchmark among surveyed works; includes trajectories and dense boxes	Offline benchmark, not continuous user monitoring; no evidence-first narration target	Primary supervised source and strongest desktop-centric evaluation reference
DeskVision [3]	Large-scale desktop region captioning and GUI understanding	GUIExplorer reaches 72.02 element accuracy on DeskVision-Eval and 82.86 average on ScreenSpot	Large desktop scale; balanced across operating systems; rich region captions and good transfer	Region captions are weaker than interaction traces; auto-generated data may be noisier than dense manual labels	Best auxiliary data source for weak supervision and role-family priors
ScreenSpot-Pro [4]	Grounding on high-resolution professional computer use	Best end-to-end model gets 18.9%; ScreenSeekeR reaches 48.1%	Realistic professional difficulty; expert annotation; authentic screen complexity	Small benchmark relative to training datasets; aimed at evaluation, not full pipeline design	Best external stress test for real-world grounding robustness
Sharingan [5]	VLM-based extraction of action sequences from desktop recordings	GPT-4o DF reaches 0.83/0.81 recall/precision on ACT ONE operation type and 0.82/0.70 on ACT REAL	Closest to Vision UI's temporal goal; shows action extraction can be replayable	Focuses on video-level action extraction, not element grounding or evidence-first JSON design	Supports the claim that better frame understanding should improve downstream narrative steps

TABLE II
IMPLEMENTED DATA USAGE FOR THE FINAL VISION UI REPORT.

Source	Scale	Key annotations/artifacts	Use in final project
UI-Vision [2]	83 desktop applications, 6 domains, about 450 videos, about 8.2k query-label pairs	Dense UI boxes, labels, layout regions, keyframes, OCR-linked content, and action trajectories	Primary annotated training source and main held-out evaluation source for the classifier, rerankers, and hybrid gates
ScreenSpot-Pro [4]	1,581 instructions, 23 applications, 5 industries, 3 operating systems	Expert grounding boxes on authentic high-resolution professional screens	External validation and stress-test benchmark used to judge transfer beyond UI-Vision
Presentation demo run	22 captured frame events; 21 validated facts	Prerecorded Vision UI GUI capture, local screenshots, JSON evidence, GUI overlays, and narration artifacts	Qualitative notebook demonstration only; not used as training data or benchmark evidence

when the selected target box overlaps the gold target at IoU ≥ 0.5, following the object-detection convention of overlap-based target correctness [29]. Candidate recall asks whether any candidate in a proposal pool reaches that threshold. Selected-predictor recall asks whether the predictor chosen by a hybrid gate had a correct candidate available. The role-family classifier uses the five portable labels `input`, `button`, `list_item`, `document_content`, and `other`.

The final classifier and learned grounding runs use `5,189` UI-Vision-derived training examples, a `300`-example UI-Vision validation/held-out split, and final evaluation over those `300` UI-Vision examples plus the `1,581`-example ScreenSpot-Pro external benchmark. Thresholded classifier metrics use the selected classifier confidence threshold `0.55`; predictions below that threshold are suppressed rather than forced into a public role-family label. On the held-out classifier evaluation, `20/300` predictions were suppressed.

Hybrid gates are trained on predictor-output caches rather than raw gold boxes. Their inputs are inference-safe features such as predictor scores, margins, ranks, box shape, cross-predictor agreement, candidate variant, and pool mode. Gold labels are used to train and evaluate the gate target, but no

TABLE III
FINAL EVALUATION SOURCES.

Source	Size	Role in final report
UI-Vision held-out split	300	Primary in-domain desktop grounding benchmark and classifier evaluation split
ScreenSpot-Pro	1,581	External stress test for professional high-resolution GUI grounding
Combined full run	1,881	Aggregate top-1 grounding accuracy used for the progression table
UI-Vision training pool	5,189	Supervised data used for classifier, reranker, and hybrid-gate training

gold-label fields are available to the gate at evaluation time. The final five-predictor recall-push hybrid gate is the best evaluated artifact, not a full seed-stability claim. Earlier v3 reranker seed sweeps provide limited robustness context: five seeds ranged from 0.3167 to 0.3333 top-1, with mean 0.3273 and population standard deviation about 0.0057.

C. Heuristic Proposals and Evidence JSON

The baseline grounding path begins with OCR spans, contours, click-local components, rectangular visual regions, and window/focus context [28]. Candidate elements can originate from text boxes, button-like regions, document panes, rows, or fallback regions around the click. Each candidate carries geometric features and provenance. The goal is not to produce perfect UI parsing; it is to create a pool of plausible elements that can be ranked, inspected, and traced back to pixels.

The per-frame JSON layer preserves capture metadata, screenshot paths and hashes, window context, clicks, OCR tokens, GUI element proposals, action hypotheses, attention regions, change metrics, role-family predictions, grounding source fields, and privacy/provenance metadata. Session-level outputs include reconstructed commands, security findings, facts, episodes, joined steps, and quality reports. This design makes evidence review possible in the Frame Inspector and keeps natural-language narrative subordinate to saved observations.

D. Role-Family Classifier and Detector Path

The crop classifier is a MobileNetV3-Large model over five role families: input, button, list_item, document_content, and other [6]. During training, UI-Vision-derived crops provide supervised role labels. During inference, the classifier adds semantic role confidence to candidate boxes, but it does not itself decide which box is the target. This distinction matters because classifier accuracy asks whether a crop is typed correctly, while top-1 grounding accuracy asks whether the selected bounding box matches the gold target at the required IoU threshold [29].

The detector path is YOLOv8-based and is cited as software rather than a formal paper [7]. It was intended to supply additional GUI element proposals. Detector-fusion combines detector confidence, classifier confidence, click geometry, OCR overlap, and native focus/window context into a target score. In evaluation, detector-only grounding did not beat the heuristic baseline on UI-Vision. That result keeps the detector path experimental and prevents it from becoming the default live target source.

E. Grounding Progression

Each grounding approach fits at a different point in the training and inference pipeline. The heuristic path has no learned training step and supplies the always-available fallback candidates. The classifier trains on UI-Vision-derived crops and contributes semantic role features at inference. The detector trains as a proposal model, then contributes candidate boxes and confidence scores. The v3, union, geometry, calibrated, and transformer rerankers train on candidate pools exported from the repo's benchmark pipeline and choose a single box from a pool at inference. The hybrid gates train on predictor-output caches and select among already-computed predictor decisions rather than generating new boxes.

The first learned-ranking step was the v3 proposal-pool reranker. It is an MLP ranker over proposal features that sits after heuristic proposal generation and before target selection, following the general learning-to-rank framing that candidate order can be learned from supervision rather than fixed by hand-built scores [10]. It produced the first major jump over heuristic, classifier-only, and detector-fusion baselines. Its strength was ranking within an existing proposal pool; its weakness was that candidate recall remained 0.4912, so no ranker could recover examples where the correct target was absent.

The next stage attacked candidate recall with a click-conditioned visual proposer. This model is DETR-style in spirit: the proposal stage sees the image together with click-conditioned information and tries to emit boxes near plausible targets [8]. It is weak as a standalone target source, but useful as an additive proposal generator. When its boxes were unioned with the heuristic pool and the MLP ranker was retrained, combined top-1 rose to 0.3620 and recall rose to 0.5284. The key lesson is that an individually weak proposer can still improve the pipeline if it contributes complementary candidates.

Geometry and document-context candidates then expanded the pool further. These candidates add click-local component geometry, document-region candidates, line/block context, and larger target hypotheses. They raised the recall ceiling to 0.5720, but the raw geometry reranker dropped to 0.3472 because the richer pool introduced more confusing candidates. A calibrated geometry MLP reduced some tight-box and oversized-panel errors, improving to 0.3509, but it still trailed the click-visual union MLP. This is the central recall/top-1 tradeoff: more correct candidates can lower top-1 if ranking does not improve at the same time.

A listwise transformer selector was added to test whether a set-aware model could reason over the expanded candidate

set better than the MLP, combining transformer-style self-attention with a listwise ranking objective [9], [10]. It preserved the 0.5720 recall ceiling but reached only 0.3424. The result suggests that adding model capacity alone did not solve the ranker problem; the hard examples were not merely a lack of set context.

The hybrid gate reframed the task as predictor selection, similar in spirit to stacked generalization and mixture-of-experts approaches that learn when to trust different component predictors [11], [12]. Instead of forcing one ranker to solve every example, a small MLP gate chooses among the click-visual union MLP, geometry-calibrated MLP, and geometry listwise transformer using inference-safe features such as scores, margins, ranks, box shape, cross-predictor agreement, candidate variant, and pool mode. This three-predictor gate reached 0.3817, showing that individually imperfect predictors became stronger together because they failed on different examples.

The next experimental step adds a visual-candidate reranker as a fourth predictor. This model scores candidates with image-plus-geometry features: candidate crops, context crops, role/geometry features, and a lightweight visual tower. Alone, it reached 0.3817; inside the four-way hybrid gate, it helped reach a combined top-1 result of 0.3939. At inference time, this gate sits after all candidate generators and predictor exports, selecting the predictor whose target should own the final click attribution.

The final workstation experiment pushes recall more aggressively. A disabled recall-push profile increases candidate retention, adds more click-visual proposals, includes more document variants, and adds profile-only click-scale ladder proposals. This richer candidate cache reached high coverage but also increased selector difficulty. A recall-push visual-candidate reranker v2 trained with partial visual fine-tuning reached 0.3966 combined top-1 and 0.7150 candidate recall as a standalone predictor. The final five-predictor hybrid gate then selected among the union MLP, geometry-calibrated MLP, geometry listwise transformer, visual-candidate v1, and recall-push visual-candidate v2. This gate reached 0.4450 combined top-1 and 0.6906 selected-predictor recall. It remains an offline experimental result because the live runtime still requires further visual-candidate integration, harmful-switch analysis, and calibration of high-confidence errors.

F. Narration and Privacy

The deterministic narrator consumes saved evidence, not raw screenshots alone. It groups primitive observations into steps such as application switch, text entry, click, scroll, command execution, reading/review, and idle review. It preserves uncertainty markers and avoids over-specific target names when evidence is weak. Optional local model support can refine prose, but deterministic artifacts remain the authoritative substrate.

Privacy-sensitive text is handled conservatively. Trusted raw text fragments, such as terminal input or code-editor text, are stored only in restricted local artifacts. Security enrichment is deterministic and labels reconstructed commands with categories such as filesystem change, package install, network fetch, remote execution, permission change, credential touch, and delete/overwrite risk. This fits the evidence-first goal: the system can support troubleshooting or security review without sending the activity stream to a server.

The quantitative evaluation in this report is limited to crop classification, click grounding, and predictor-selection diagnostics. Narration is demonstrated qualitatively through saved evidence artifacts and deterministic narrative outputs, not through a large human-rated narrative study.

V. Results

The final results are best read as a progression. Table IV preserves the historical baseline, detector, classifier, reranker, and ensemble numbers rather than replacing the original table with only the newest values. Rows are ordered from weakest to strongest combined top-1 grounding accuracy.

A. Classifier Results

The MobileNetV3-Large role-family classifier achieved 0.7767 raw accuracy, 0.7533 thresholded accuracy, and 0.6221 thresholded macro-F1 [6]. These values support the claim that the crop-level learned component is useful. The gap between accuracy and macro-F1 also shows why role-family reporting should not rely only on aggregate accuracy: some role families are easier than others, and thresholding changes both coverage and class balance.

The per-role breakdown also reinforces the main grounding lesson. document_content was comparatively easy to label at the crop level, but it remained difficult to ground as an exact clicked target. This is why the report treats crop semantics and click-target selection as separate ML measurements.

B. Grounding Progression Results

The progression shows four patterns. First, classifier-only improved role semantics and detector-fusion added visual/object-detection signals, but both improved over heuristic only modestly. Second, the v3 reranker-shadow run was the first major improvement because it directly attacked target ranking rather than only adding role labels. Third, recall-expanding candidates raised the candidate ceiling but also introduced distractors. Fourth, hybrid predictor selection produced the best result by choosing among complementary failure modes. In narration terms, these gains increase the share of clicks for which the evidence layer can attach a plausible visible target, but they do not by themselves establish narrative correctness. The best 0.4450 value is grounding accuracy, not classifier accuracy or a direct narrative-quality score.

C. Diagnostic Variants

The diagnostic terms in Table V refer to concrete selector behavior. A *runtime gate* is the criterion for adopting a learned path into the live/default runtime. A *useful switch* occurs when the hybrid gate chooses a predictor that fixes a union-model

TABLE IV

HISTORICAL TOP-1 GROUNDING PROGRESSION ORDERED FROM WEAKEST TO STRONGEST COMBINED RESULT. VALUES ARE TOP-1 GROUNDING ACCURACY EXCEPT FOR THE RECALL COLUMN. STANDALONE PREDICTOR ROWS REPORT CANDIDATE RECALL, WHILE HYBRID-GATE ROWS REPORT SELECTED-PREDICTOR RECALL, ALL AT IOU ≥ 0.5.

Approach	Pipeline role	UI-Vision	ScreenSpot-Pro	Combined	Recall	Interpretation
Heuristic baseline; detector-only matched	Heuristic boxes or detector replacement baseline	0.1000	0.1765	0.1643	0.4912	Weakest step; detector-only matched the heuristic and failed the runtime gate.
Classifier-only	Crop role-family semantics	0.1600	0.2024	0.1956	0.4912	Improved role labels, but could not fix missing or wrongly ranked candidate geometry.
Detector-fusion	Detector, classifier, OCR, click, and native-context fusion	0.1700	0.2030	0.1978	0.4912	Added signals but remained far below learned proposal-pool ranking.
v3 reranker-shadow	MLP ranking over proposal pool	0.3333	0.3435	0.3418	0.4912	First major jump over heuristic and fusion baselines.
Transformer listwise	Set-aware selector over expanded geometry candidates	0.3467	0.3416	0.3424	0.5720	More capacity preserved recall but did not solve the ranking gap.
Geometry calibrated MLP	Geometry/document-context pool with score calibration	0.3500	0.3510	0.3509	0.5720	Reduced some shape errors, but the richer pool remained hard to rank.
Click-visual union MLP	Heuristic plus click-conditioned visual proposal union reranker	0.3500	0.3643	0.3620	0.5284	Strongest single reranker baseline; weak visual proposals helped when unioned with heuristics.
Raw hybrid gate	Three-predictor ensemble selector	0.3800	0.3820	0.3817	0.5603	Combined union, geometry, and transformer predictors into a stronger selector.
Hybrid + visual-candidate gate	Four-predictor ensemble with image-plus-geometry candidate scoring	0.3933	0.3941	0.3939	0.5646	Previous best; showed image-plus-geometry scoring helped predictor selection.
Recall-push visual-candidate v2	Larger recall-push candidate pool plus partially tuned visual reranker	0.4267	0.3909	0.3966	0.7150	Strong recall expansion; top-1 improved modestly because the harder pool still challenged ranking.
Five-predictor recall-push hybrid gate	Five-way selector adding recall-push visual-candidate v2	0.4533	0.4434	0.4450	0.6906	Best experimental result; offline until visual-candidate runtime adoption and calibration improve.

TABLE V

NEGATIVE AND NEAR-MISS HYBRID VARIANTS. THESE DIAGNOSTICS WERE USEFUL, BUT NONE BEAT THE FIVE-PREDICTOR RECALL-PUSH HYBRID GATE.

Variant	Top-1	Recall	Interpretation
Guarded hybrid	0.3812	0.5556	Validation-tuned guard did not beat raw held-out hybrid accuracy.
Seed ensemble	0.3806	0.5625	Median-logit seed ensemble improved stability but not top-1.
Veto hybrid	0.3785	0.5577	Reduced some harmful switches but lost too many useful recoveries.
Union-priority hybrid	0.3700	0.5311	Over-corrected toward union and missed geometry rescue signal.
Rescue switcher	0.3620	0.5460	Recovered many union misses but introduced too many harmful switches.

miss. A *harmful switch* occurs when the gate switches away from a correct union prediction and fails. A *high-confidence wrong pocket* is a high- or very-high-margin gate selection that still misses the IoU threshold [29]. Synthetic click supervision means benchmark-derived click/box supervision rather than real local user click labels.

The final five-predictor gate selected the union MLP 130 times, the geometry-calibrated MLP 287 times, the geometry listwise transformer 536 times, visual-candidate v1 439 times, and recall-push visual-candidate v2 489 times. It produced 310 useful switches away from the union MLP and 154 harmful switches. It recovered 310 of 393 recoverable union errors, a recovery rate of 0.7888, but still left 359 high-confidence wrong pockets. The five-predictor oracle is 0.5710, so the new predictor pool raises the earlier oracle ceiling above 0.50; the remaining gap is now selection, curriculum, and calibration rather than recall alone.

VI. DISCUSSION

A. What the Results Support

The strongest supported claim is that Vision UI successfully converts the earlier survey thesis into a working local-first evidence pipeline. The repo contains live and batch analysis paths, saved screenshots, JSON evidence, GUI inspection artifacts, deterministic narration, optional local model support, and a benchmarked grounding extension. That is a substantive

TABLE VI

REPRESENTATIVE DECISION-PATTERN EXAMPLES FROM THE FIVE-PREDICTOR HYBRID GATE. THESE ARE QUALITATIVE DIAGNOSTICS FROM EXISTING HELD-OUT DECISION ARTIFACTS, NOT ADDITIONAL BENCHMARK METRICS.

Case	Source	Union behavior	Gate behavior	Interpretation
Useful switch	UI-Vision / VLC / document content	Selected a tiny partial box with IoU `0.1645`.	Selected a geometry/listwise row-union candidate with IoU `0.6113`.	Predictor diversity can recover errors when a richer candidate family contains the right target.
Harmful switch	UI-Vision / Flameshot / document content	Correct at IoU `0.5195`.	Switched to a visual-candidate component block and failed at IoU `0.2847`.	The gate sometimes over-trusts plausible document candidates and breaks a correct baseline.
High-confidence wrong pocket	UI-Vision / Scribus / document content	Failed at IoU `0.0203`.	Very-high-margin recall-push region clip also failed at IoU `0.0096`.	Some high-margin decisions are confidently wrong, so runtime use needs calibration and review.

Fig. 1. Thresholded role-family confusion matrix for the MobileNetV3-Large classifier.

systems result independent of whether every learned component is production-ready.

At the ML level, the results support a more specific claim: grounding improved when the project moved from role labeling and detector-fusion toward proposal-pool ranking, predictor selection, and then recall-push candidate expansion [10], [11], [12]. The v3 reranker-shadow value of `0.3418` should be read as the first major jump, not the endpoint. The click-visual union MLP showed that a weak visual proposer can still be useful when it adds complementary candidates to the pool. The four-predictor hybrid gate showed that multiple imperfect predictors can become stronger together when their errors are not identical. The five-predictor recall-push gate then showed that expanding the candidate pool can pay off when a selector has enough predictor diversity to avoid many new distractors.

The grounding gains should therefore be interpreted as improved evidence quality for downstream narration, not as a direct measurement of narrative fidelity. They improve grounding readiness for an evidence-first narrator by making correct visible targets available and selectable more often.

B. Why Individually Weak Pieces Helped

The visual proposer alone was weak, and the geometry-expanded candidate pool lowered top-1 before calibration because it added confusing alternatives. Those results might look negative if each component were judged in isolation. In combination, however, they changed the candidate and predictor landscape. The union MLP benefited from additional proposal recall without taking the full geometry ranking penalty. The geometry and transformer predictors recovered some targets missed by the union model [9], [10]. The visual-candidate rerankers added image-plus-geometry evidence that was different from tabular score features. The recall-push v2 predictor was especially important because it raised candidate recall to `0.7150` as a standalone predictor, giving the five-way gate more recoverable cases.

This is the central lesson of the ensemble result. The best system was not created by simply adding more boxes or a larger model. It was created by combining predictors that made different mistakes, then learning a selector from inference-safe signals [11], [12]. The tradeoff remains visible: useful switches improved top-1, but harmful switches and high-confidence wrong pockets are still too common for automatic runtime adoption. The five-predictor result is therefore a stronger experimental endpoint, not a claim that the learned stack is production-ready.

C. Why the Detector Failure Still Matters

The detector path matters because it tests a tempting but risky assumption: that adding a modern object detector will automatically improve GUI grounding [7]. The current results reject that assumption for this repo state. Detector-only performance remained equal to the heuristic baseline on UI-Vision, and detector-fusion remained far below reranking and ensemble conditions. This suggests that detector proposals did

not reliably match the target geometry needed for desktop GUI actions, especially under dense layouts, small controls, and ambiguous panels.

This failure also validates the project's evidence-first discipline. Because the pipeline records candidate provenance, top-1 accuracy, candidate recall, ranker gaps, and gate diagnostics, the detector issue appears as a measurable result rather than as vague demo failure. The system can keep the heuristic path active while treating detector fusion and ensemble gating as experimental.

D. Connection Back to the Survey

The final results align closely with the five-paper survey and the broader grounding literature. SeeClick predicted that grounding would dominate downstream reliability, while IVG made the contrast between OCR-grounded and direct visual grounding strategies explicit [1], [19]. UI-Vision showed that desktop grounding is difficult enough that modest gains are meaningful. ScreenSpot-Pro showed that professional high-resolution screens are a serious transfer test, which is why the `0.4434` ScreenSpot-Pro result is valuable even though it is not a state-of-the-art claim. DeskVision remains the most plausible future route for improving desktop proposal priors at scale. Sharingan remains the strongest motivation for improving temporal action extraction once frame-level grounding becomes more reliable.

The key research gap also remains intact. Existing papers provide benchmarks, datasets, visual agents, and recording-level action extraction, but they do not deliver the same combination of local capture, native context, inspectable evidence JSON, deterministic fallback, privacy-aware raw-text handling, GUI review, and session narration. Vision UI's contribution is therefore a systems integration and evaluation contribution, not a claim that a small course model has solved GUI grounding.

E. Academic Fit and Evidence-First Interpretation

This framing places Vision UI between GUI-agent grounding research and desktop activity reconstruction. It is not presented as a state-of-the-art autonomous computer-use agent. Its academic fit is narrower and more practical: it studies how learned GUI grounding can be embedded inside an evidence-preserving desktop understanding system where every downstream narrative claim remains tied to saved artifacts.

The best `0.4450` combined top-1 result is therefore useful experimental evidence, but it is not reliable enough to authoritatively fill every click target in a live narrative. In the implemented system, learned visual predictions should be treated as uncertain evidence inside a deterministic pipeline. A model-selected box can improve the event record when it is correct, but it should not overwrite stronger signals or remove uncertainty when the confidence and supporting context are weak.

This distinction also clarifies why the system can still produce useful narratives even while the learned grounding stack remains experimental. OS primitives, OCR, active-window context, click coordinates, keyboard evidence, command reconstruction, and deterministic step joining often carry more immediate narration value than visual classification alone. The stronger academic contribution is the combination: local capture, structured JSON evidence, grounding evaluation, deterministic fallback, and bounded narration from saved evidence.

F. Implementation Difficulties, Limitations, and Future Work

Several implementation difficulties shaped the final result. The first was that repeated GPU experiments often produced small or inconsistent top-1 gains even when the architectural change appeared promising. That experience was informative rather than wasted: it made clear that practical desktop grounding depends on the interaction among proposal recall, role semantics, ranking, calibration, and evaluation design rather than on any single model improvement.

The second difficulty was that strong crop classification did not automatically produce strong clicked-target grounding. A candidate crop can be correctly classified as a button or document region while still being the wrong box for the user's actual click. This forced the project to separate role-family classification, proposal recall, and top-1 target ranking in the evaluation.

The third difficulty was candidate-pool expansion. Adding click-conditioned visual proposals and geometry/document-context candidates improved recall, but it also made ranking harder. Some added boxes were plausible enough to confuse the MLP or transformer selector [8], [9], [10]. This is why the geometry pool reached `0.5720` recall while its top-1 result trailed the click-visual union MLP until later hybrid selection. The recall-push profile raised the standalone visual-candidate predictor's candidate recall to `0.7150`, but the five-predictor oracle of `0.5710` and the final gate's `0.4450` top-1 show that selection quality still lags the available candidate ceiling. The fourth difficulty was avoiding leaky evaluation for ensemble gates. The valid hybrid gate had to train on generated train/validation predictor-output caches and evaluate once on held-out UI-Vision plus ScreenSpot-Pro exports; otherwise, the gate could appear stronger than it really was [11].

The first limitation is scale. The final held-out UI-Vision split contains `300` examples, and ScreenSpot-Pro contributes `1,581` examples. That is enough for a defensible course report but not enough to claim broad desktop generalization. The second limitation is narrative evaluation. The repo demonstrates qualitative narration through saved local artifacts, but it does not include a large human evaluation of narrative correctness, usefulness, or safety.

The third limitation is provider variability. OCR availability, active-window metadata, focus hints, click capture, and accessibility APIs differ by operating system and permissions. Vision UI handles missing signals through graceful degradation, but lower-quality signals will reduce grounding quality, narrative detail, and reliability. The fourth limitation is privacy risk. Screenshot streams can include credentials, documents, messages, and private work. The system's local-first design

and restricted raw-text handling reduce that risk, but they do not eliminate the need for careful deployment boundaries.

The learned grounding stack is also not the default truth path. Detector-only failed the runtime gate, detector-fusion remains below the live-readiness bar, and the best five-predictor hybrid gate is an offline benchmark result. Runtime adoption should wait for visual-candidate runtime integration, further harmful-switch review, calibration of high-confidence wrong pockets, and validation on real click-labeled local sessions rather than only synthetic click supervision and public benchmark labels.

A natural next iteration is to create the originally proposed project-specific Vision UI adaptation dataset from real local sessions. That dataset should include screenshot paths, click coordinates, target boxes, role-family labels, and optional application-specific semantic roles. DeskVision should also be used to scale region-level supervision beyond UI-Vision. Finally, temporal evaluation should test whether higher top-1 grounding accuracy actually improves target mentions, operation recovery, temporal grouping, and human-rated narrative faithfulness on real local sessions, following the broader action-sequence and screencast-evaluation motivation [20], [21], [22], [23], [5], [24], [25], [26], [27].

VII. Conclusion

Vision UI began as a survey-driven hypothesis: the most valuable learning target for evidence-first desktop activity narration is not a full end-to-end agent, but reliable GUI grounding. The final repo implements that hypothesis as a local-first system that captures screenshots, records native context, extracts OCR and heuristic proposals, serializes inspectable JSON, supports deterministic narration, and evaluates a learned grounding progression on UI-Vision and ScreenSpot-Pro.

The final results are mixed but useful. The role-family classifier is strong at the crop level, with `0.7767` raw accuracy, `0.7533` thresholded accuracy, and `0.6221` thresholded macro-F1. Grounding progressed from the v3 reranker-shadow result of `0.3418` combined top-1 through click-visual union, raw hybrid gating, hybrid plus visual-candidate gating, and finally the five-predictor recall-push hybrid gate at `0.4450` combined top-1, `0.4533` on UI-Vision, `0.4434` on ScreenSpot-Pro, and `0.6906` selected-predictor recall. Detector-only grounding still failed the UI-Vision runtime gate, matching the `0.1000` heuristic baseline with `0.0000` improvement. The final claim is therefore bounded: Vision UI is a substantive desktop understanding system, and its learned grounding stack is a promising but still experimental progression toward stronger grounding readiness for evidence-first narration.

Vision UI did not solve GUI grounding, but it suggests a path toward making desktop activity review more useful for troubleshooting, auditing, forensic summaries, and human understanding in a bounded, evidence-first setting.

References

[1] K. Cheng, Q. Sun, Y. Chu, F. Xu, Y. Li, J. Zhang, and Z. Wu, "Seeclick: Harnessing gui grounding for advanced visual gui agents," *arXiv preprint arXiv:2401.10935*, 2024.

[2] S. Nayak, X. Jian, K. Q. Lin, J. A. Rodriguez, M. Kalsi, R. Awal, N. Chapados, M. T. Ozsu, A. Agrawal, D. Vazquez, C. Pal, P. Taslakian, S. Gella, and S. Rajeswar, "Ui-vision: A desktop-centric gui benchmark for visual perception and interaction," *arXiv preprint arXiv:2503.15661*, 2025.

[3] Y. Xu, L. Yang, H. Chen, H. Wang, Z. Chen, and Y. Tang, "Deskvision: Large scale desktop region captioning for advanced gui agents," *arXiv preprint arXiv:2503.11170*, 2025.

[4] K. Li, Z. Meng, H. Lin, Z. Luo, Y. Tian, J. Ma, Z. Huang, and T.-S. Chua, "Screenspot-pro: Gui grounding for professional high-resolution computer use," *arXiv preprint arXiv:2504.07981*, 2025.

[5] Y. Chen, Y. Ren, X. Qin, J. Zhang, K. Yuan, L. Han, Q. Lin, D. Zhang, S. Rajmohan, and Q. Zhang, "Sharingan: Extract user action sequence from desktop recordings," *arXiv preprint arXiv:2411.08768*, 2024.

[6] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam, "Searching for mobilenetv3," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1314–1324.

[7] Ultralytics, "Ultralytics yolov8," https://github.com/ultralytics/ultralytics, 2023, software and documentation.

[8] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *European Conference on Computer Vision*, 2020, pp. 213–229.

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017.

[10] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li, "Learning to rank: From pairwise approach to listwise approach," in *Proceedings of the 24th International Conference on Machine Learning*, 2007, pp. 129–136.

[11] D. H. Wolpert, "Stacked generalization," *Neural Networks*, vol. 5, no. 2, pp. 241–259, 1992.

[12] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural Computation*, vol. 3, no. 1, pp. 79–87, 1991.

[13] Y. Li *et al.*, "Widget captioning: Generating natural language description for mobile user interface elements," *arXiv preprint arXiv:2010.04295*, 2020.

[14] B. Wang, G. Li, Y. Li, X. Chen, Y. Shi, and Y. Li, "Screen2words: Automatic mobile ui summarization with multimodal learning," *UIST / arXiv:2108.03353*, 2021.

[15] Y.-C. Hsiao *et al.*, "Screenqa: Large-scale question-answer pairs over mobile app screenshots," *arXiv preprint arXiv:2209.08199*, 2022.

[16] C. Bai *et al.*, "Understanding mobile gui: From pixel-words to screen-sentences," *arXiv preprint arXiv:2105.11941*, 2021.

[17] K. Lee, M. Joshi, I. Turc, H. Hu, F. Liu, J. Eisenschlos, U. Khandelwal, P. Shaw, M.-W. Chang, and K. Toutanova, "Pix2struct: Screenshot parsing as pretraining for visual language understanding," *arXiv preprint arXiv:2210.03347*, 2022.

[18] G. Baechler, S. Sunkara, M. Wang, F. Zubach, H. Mansoor *et al.*, "Screenai: A vision-language model for ui and infographics understanding," *arXiv preprint arXiv:2402.04615*, 2024.

[19] E. M. Ettifouri *et al.*, "Visual grounding methods for efficient interaction with desktop graphical user interfaces," *arXiv preprint arXiv:2407.01558*, 2024.

[20] Y. Li *et al.*, "Mapping natural language instructions to mobile ui action sequences," *arXiv preprint arXiv:2005.03776*, 2020.

[21] C. Rawles *et al.*, "Android in the wild: A large-scale dataset for android device control," *arXiv preprint arXiv:2307.10088*, 2023.

[22] K. Q. Lin *et al.*, "Videogui: A benchmark for gui automation from instructional videos," *arXiv preprint arXiv:2406.10227*, 2024.

[23] J. Zhang *et al.*, "Gui-world: A video benchmark and dataset for multimodal gui-oriented understanding," *arXiv preprint arXiv:2406.10819*, 2024.

[24] S. Feng *et al.*, "Read it, don't watch it: Captioning bug recordings automatically," *arXiv preprint arXiv:2302.00886*, 2023.

[25] J. Li *et al.*, "Screencast tutorial video understanding," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[26] A. Yang *et al.*, "Vid2seq: Large-scale pretraining of a visual language model for dense video captioning," *arXiv preprint arXiv:2302.14115*, 2023.
[27] M. Qasim *et al.*, "Dense video captioning: A survey of techniques, datasets and evaluation protocols," *arXiv preprint arXiv:2311.02538*, 2023.
[28] R. Smith, "An overview of the tesseract ocr engine," in *Ninth International Conference on Document Analysis and Recognition*, vol. 2, 2007, pp. 629–633.
[29] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, 2010.