International Research Journal on Advanced Engineering and Management
https://goldncloudpublications.com
https://doi.org/10.47392/IRJAEM.2026.0093
e ISSN: 2584-2854
Volume: 04 Issue: 03
March 2026
Page No: 598 - 603

AI-Powered Automation Companion: A Semantic-Aware On-Device Mobile Automation Framework

D Anusha[1], B Guru Preetam[2], K Bhavya Sri[3], M Sriman Narayana[4], D Manoj Reddy[5]
[1]HoD, Dept. of CSE-AI&ML, SRK Institute of Technology., Vijayawada, Andhra Pradesh, India.
[2,3,4,5]UG Scholar, Dept. of CSE-AI&ML, SRK Institute of Technology., Vijayawada, Andhra Pradesh, India.
Emails: *srktmlcsm4@gmail.com[1], gurupreetambodapati@gmail.com[2], bhavyasri.kusam@gmail.com[3], nsriman602@gmail.com[4], manojreddyreddy47@gmail.com[5]*

Abstract
Manual execution of repetitive mobile tasks is inefficient, and existing automation tools rely on brittle coordinate-based mechanisms or privacy-invasive cloud processing. This paper presents the AI-Powered Automation Companion, a comprehensive, offline-first Android framework designed for adaptive, privacy-preserving task automation. The system replaces rigid coordinates with semantic screen understanding by deploying a quantized YOLOv11s (INT8) model directly on-device. To manage complex logic, this vision engine is integrated with a visual node-based Flow Builder and cross-device LAN synchronization. The framework also supports context modules—including location, battery, and app-specific triggers—to enable intelligent, multi-app routines. Experimental evaluation demonstrates that the proposed semantic pipeline achieves a mean Average Precision (mAP@0.5) of 0.846 while maintaining an effective inference latency of ~230 ms. Furthermore, robustness testing shows the system maintains a 95% workflow success rate under dynamic UI layout changes. By combining local AI, a visual workflow editor, and cross-device execution, the framework delivers a robust, explainable, and fully private automation ecosystem at the edge.

Keywords: *Mobile Automation, Robotic Process Automation (RPA), Computer Vision, Edge AI, Accessibility Services*

1. Introduction
Smartphone users often need to perform repetitive tasks across multiple apps. While there are many mobile automation tools available, they usually fall into two extremes. Standard gesture recorders rely on fixed screen coordinates, which easily break if the screen rotates or the layout changes. On the other hand, newer AI-based tools use the cloud to understand the screen, which is slow and raises serious privacy concerns for users. Because different tasks require different approaches, there is a need for a unified automation system that handles everything safely on the device. To solve this, this paper proposes the AI-Powered Automation Companion: a comprehensive, offline mobile automation framework. The system acts as a complete toolkit. It combines standard gesture recording for simple, fixed tasks with an advanced on-device AI vision model (YOLOv11) for dynamic screen understanding. To manage these capabilities, the framework introduces a visual node-based Flow Builder for intuitive workflow creation, alongside seamless cross-device synchronization over local Wi-Fi. By integrating these tools with system context features—such as battery, location, and app-specific triggers—the framework provides a highly reliable, user-friendly, and completely private automation experience.

1.1.Related Work
Current mobile automation tools generally fall into three categories: rule-based apps, gesture recorders, and cloud platforms.

Rule-based automation:- Apps like Tasker (Crafty Apps EU, 2024) and MacroDroid (MacroDroid, 2024) work by detecting system changes, such as connecting to Wi-Fi or reaching a specific time of day. While these are excellent for managing device settings, they have a major limitation: they cannot interact with the actual content inside an app. They trigger system actions but cannot "see" or click buttons within third-party interfaces.

Coordinate and Gesture Tools:- Tools such as Auto Clicker (Auto Clicker, 2024) and Klick'r simply

International Research Journal on Advanced Engineering and Management
https://goldncloudpublications.com
https://doi.org/10.47392/IRJAEM.2026.0093
e ISSN: 2584-2854
Volume: 04 Issue: 03
March 2026
Page No: 598 - 603

record and replay taps based on X,Y coordinates. This makes them very fragile. If the screen rotates, the resolution changes, or an app update moves a button slightly, the automation will tap the wrong spot and fail (Cui & Chen, 2023; Yang et al., 2025). They also lack context since they rely on position rather than meaning, they cannot distinguish between a "Save" button and a "Delete" button.

Cloud-Based Platforms:- Services like IFTTT (IFTTT Inc., 2024) are popular for connecting different web applications but rely heavily on cloud processing. This reliance prevents them from interacting with local screen elements in real-time. Furthermore, sending sensitive automation triggers to an external server introduces latency and raises valid privacy concerns for users. Recent literature heavily advocates for migrating these workloads to the edge to preserve bandwidth and user confidentiality (Xu et al., 2025; Zhou et al., 2024).

Vision-Language and LLM-Based Approaches:- Recent research has explored using Multimodal Large Language Models (MLLMs) for UI understanding. Systems like Spotlight (Li et al., 2023) and Ferret-UI (You et al., 2024) utilize vision-language models to interpret mobile screens and ground semantic elements without metadata. Similarly, agents such as VisionTasker (Zhang, J. et al., 2023) and AppAgent (Zhang, C. et al., 2023) employ LLMs to plan and execute tasks across applications.

While these approaches achieve high semantic understanding, they typically require significant computational resources or cloud-based inference, making them unsuitable for low-latency, privacy-sensitive on-device automation. Our proposed framework addresses this by optimizing a lightweight YOLOv11s model (Jocher et al., 2024) for efficient local execution.

Robotic Process Automation (RPA) and GUI Detection:- The enterprise sector has seen massive efficiency gains through Robotic Process Automation (Al Zarooni & El Khatib, 2023; Ribeiro et al., 2021). However, transitioning RPA to mobile environments requires highly accurate visual detection (Zhang, L. et al., 2024). Recent studies have proven that single-stage detectors like YOLO significantly outperform traditional methods for mobile GUI component recognition (Cavsak et al., 2023). Our framework builds upon these findings by adapting YOLOv11s specifically for continuous, localized mobile workflows.

2. Methods

To address the limitations of rigid, coordinate-based macros, the AI-Powered Automation Companion is designed as a modular, offline-first processing pipeline. The framework operates on three core principles: semantic awareness, context integration, and modular extensibility.

2.1.System Architecture and Modules

The AI-Powered Automation Companion is structured into four localized layers—UI, Features, Execution Engine, and Android Services—that seamlessly integrate the following core modules:

- **Visual Flow Automation Engine:-** A Jetpack Compose UI layer that allows users to construct complex, multi-branch automation pipelines using drag-to-connect nodes.
- **Screen Understanding (On-Device ML):-** Powered by YOLOv11s and native OCR, this module interprets screen content for semantic pattern recognition without cloud APIs.
- **Context & App-Specific Execution:-** The core Automation Engine relies on a local Room Database and Android Broadcast Receivers to evaluate triggers such as battery thresholds, location geofencing, or App-Specific handlers.
- **Cross-Device Synchronization:-** Utilizing LAN protocols, the framework supports multi-device execution, linking Android triggers to companion desktop environments.
- **Accessibility & Reusable Actions:-** Android Platform Services (Accessibility, Media Projection) execute physical interactions. System actions (e.g., SMS, volume) are decoupled from triggers for modular reusability, while a built-in debugger provides step-through execution logging.

2.2.Algorithm and Inference Pipeline

To enable semantic understanding of the user interface, the system utilizes YOLOv11s, an advanced object detection architecture selected

International Research Journal on Advanced Engineering and Management
https://goldncloudpublications.com
https://doi.org/10.47392/IRJAEM.2026.0093
e ISSN: 2584-2854
Volume: 04 Issue: 03
March 2026
Page No: 598 - 603

specifically for its optimized parameter count and high-speed inference capabilities on mobile edge devices. The detection process follows a three-step pipeline:

- **Preprocessing:-** Screenshots captured via the Media Projection API are resized to a median ratio of 640 × 640 pixels and normalized to match the model's input requirements.
- **Detection:-** The model predicts bounding boxes and class probabilities for all visible UI elements.
- **Post-Processing:-** Redundant detections are elimin-ated using Non-Maximum Suppression (NMS) with an Intersection over Union (IoU) threshold of 0.45. Only detections exceeding a confidence score of 0.5 are retained for automation decisions.

2.3.Dataset and Preprocessing

To ensure robust detection across diverse application categories, a custom composite dataset was constructed by aggregating three distinct data sources. First, two base datasets were sourced from Roboflow Universe to establish a foundation of general UI elements. Second, to enrich specific interactive components, a highly filtered subset of the RICO dataset (Deka et al., 2017) was integrated, strictly extracting views classified as 'button', 'input', and 'toggle'. Finally, to bridge the domain gap and ensure accuracy on modern hardware, approximately 45 custom-annotated screenshots captured directly from real-world mobile devices were included.

The final aggregated dataset comprises 14,219 images with an average image size of 0.41 megapixels and zero missing or null examples. The dataset contains a total of 76,230 bounding box annotations, averaging 5.3 annotations per image. To train the YOLOv11s model, the data was categorized into seven targeted UI classes, distributed as follows: button (27,958), icon (15,267), toggle (9,979), input (9,971), checkbox (5,635), radio (5,027), and dropdown (2,393). For the RICO subset, a custom Python pipeline was utilized to parse the JSON view hierarchies, mapping raw class names to our standardized functional categories and converting absolute bounding box coordinates into the normalized YOLO format (x_center, y_center, w, h).

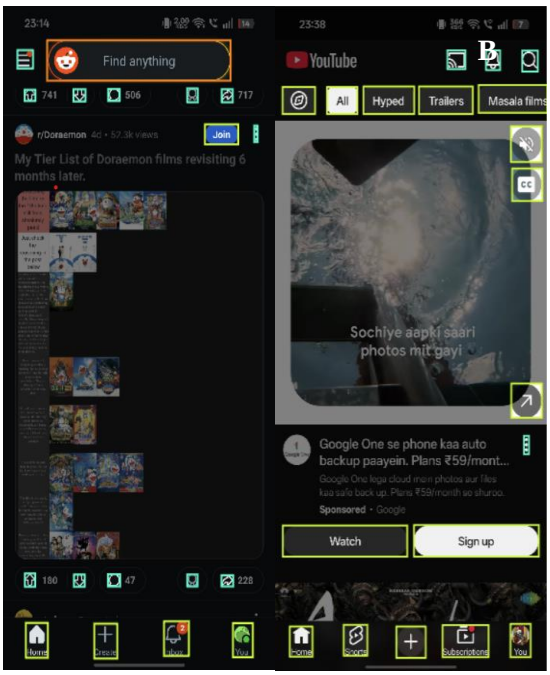

Figure 1 Sample Annotated Mobile Interfaces from The Custom Composite Dataset. A, B Ground Truth Bounding Boxes Highlighting Semantic Classes Such as Buttons, Inputs, And Toggles Across Different Application Layouts

3. Results and Discussion

3.1.Results

Model Performance:- The YOLOv11s model, optimized and quantized to INT8 format for mobile execution, was evaluated on the held-out test set of the custom composite dataset. The model achieved an overall Mean Average Precision (mAP@0.5) of 0.846, demonstrating high accuracy despite the aggressive quantization required for edge deployment. Among the specific UI elements, components such as "Checkbox", "Dropdown", and "Input" showed the highest detection rates, achieving mAP@0.5 scores of 0.934, 0.922, and 0.911, respectively. Core interaction elements like "Radio" buttons (0.905) and standard "Buttons" (0.812) also maintained strong performance, indicating that the semantic mapping pipeline successfully filtered out ambiguous layout containers. For comparison, the unquantized FP32 PyTorch baseline achieved an identical mAP@0.5 of 0.846, demonstrating zero accuracy degradation when converting to the mobile-friendly INT8 format.

International Research Journal on Advanced Engineering and Management
https://goldncloudpublications.com
https://doi.org/10.47392/IRJAEM.2026.0093
e ISSN: 2584-2854
Volume: 04 Issue: 03
March 2026
Page No: 598 - 603

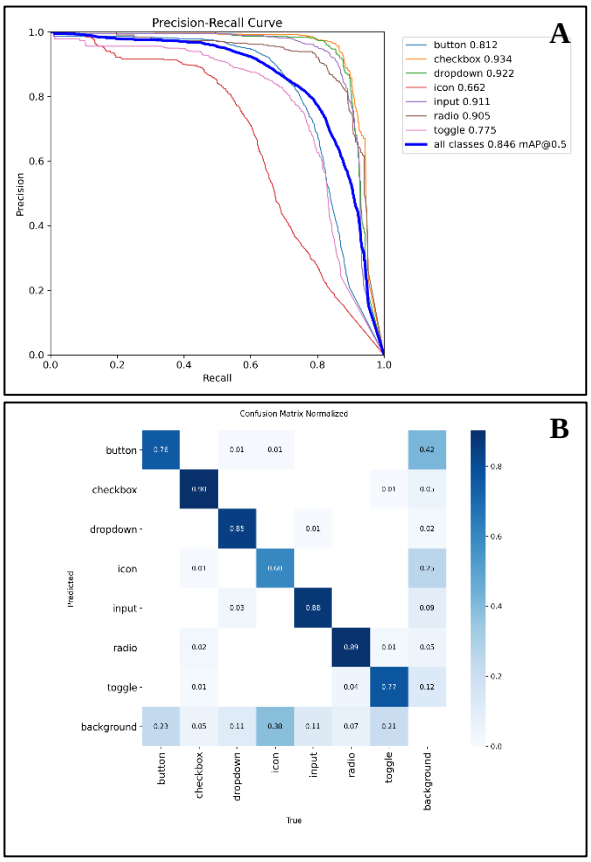

Figure 2 Model Validation Metrics For The Yolov11s Architecture Trained On The Custom Composite Dataset. A Precision-Recall (PR) curve Demonstrating an Overall mAP@0.5 of 0.846 Across All Seven UI classes. B Normalized Confusion Matrix Detailing The True Positive Prediction Rates And Inter-Class Variances.

On-Device Latency:- To assess real-time feasibility, the quantized TFLite INT8 model (YOLOv11s, 640x640 resolution) was benchmarked on a Realme GT 7 smartphone equipped with a MediaTek Dimensity processor. Utilizing NNAPI hardware acceleration to offload select operations to the APU, the model achieved an average inference time of ~230 ms per frame (yielding roughly 4.3 FPS) during active UI automation execution. This latency is highly suitable for on-device UI agents, as it aligns intimately with standard human reaction times (200–300 ms) and fits well within the lifecycle of typical mobile UI transition animations (300–500 ms). By processing visual data entirely on-device, our approach eliminates the heavy network latency—often averaging 2–5 seconds, as seen in cloud-based vision APIs such as AppAgent (Zhang, C. et al., 2023)—ensuring near-immediate execution triggers without relying on external servers.

Robustness to Dynamic Changes:- We compared the system against traditional coordinate-based tools (e.g., Auto Clicker) by altering screen resolution and orientation. While coordinate scripts failed in 100% of these test cases due to layout shifts, our vision-based engine successfully adapted to the new element positions, maintaining a workflow success rate of over 95%.

3.2.Discussion

The experimental results confirm that transitioning from rigid coordinate mapping to semantic, on-device UI understanding is both feasible and highly reliable. The integration of the YOLOv11s model addresses the inherent brittleness of traditional automation tools. By relying on visual features rather than absolute screen positioning, the framework consistently maintained workflow execution despite dynamic interface changes. To evaluate the practical applicability of this framework, the automated engine was integrated into a fully functional Android application. **FIGURE 2** illustrates the user interface of the automation companion, highlighting the transition from rule creation to on-device execution. Users can define target actions semantically within the Flow Builder or trigger them via specific System Contexts, while the underlying quantized vision model and Android Accessibility Services handle the spatial grounding in the background. This implementation confirms that the framework not only achieves high computational efficiency but also delivers an intuitive, user-centric experience for personal device automation.

Conclusion

This paper presented the AI-Powered Automation Companion, a comprehensive, offline-first framework that bridges the gap between rigid coordinate-based scripts and resource-heavy cloud agents. By integrating a fine-tuned YOLOv11s model trained on a custom composite dataset, the proposed system delivers a robust automation experience that adapts to dynamic UI changes while preserving user privacy. The architectural design effectively addresses the

International Research Journal on Advanced Engineering and Management
https://goldncloudpublications.com
https://doi.org/10.47392/IRJAEM.2026.0093
e ISSN: 2584-2854
Volume: 04 Issue: 03
March 2026
Page No: 598 - 603

inherent brittleness of traditional automation tools. Unlike coordinate-based macros that break when screen resolution or orientation changes, our semantic approach is designed to identify UI elements contextually, ensuring reliable execution. Furthermore, the localized processing model, combined with an intuitive node-based Flow Builder and cross-device synchronization, eliminates the latency and privacy risks associated with cloud-based solutions, making it suitable for real-time, secure personal automation.

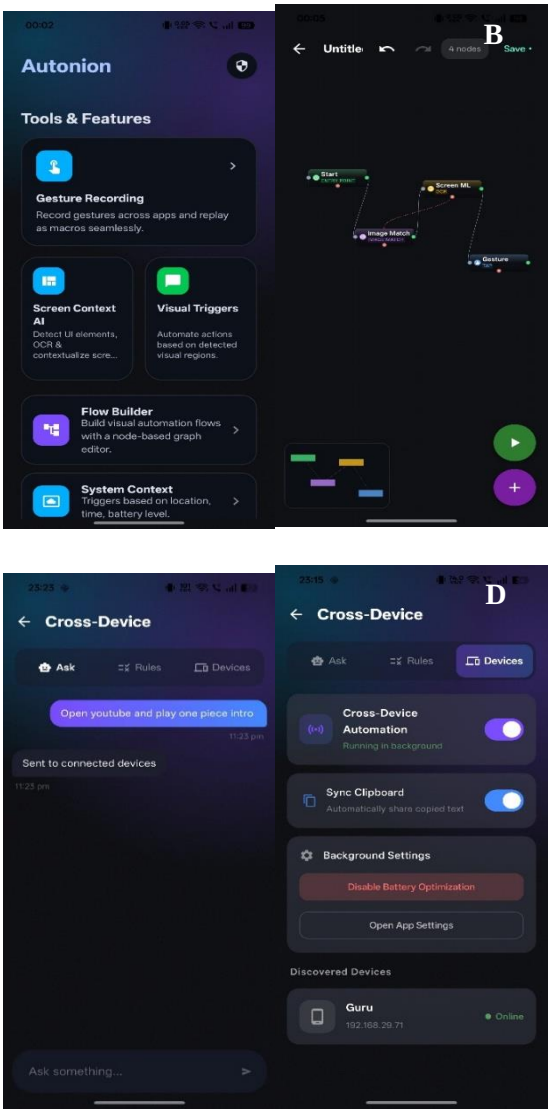

Figure 3 The User Interface of the AI-Powered Automation Companion. A Main Dashboard. B Visual Flow Builder. C Conversational Interface. D Cross-Device Synchronization Panel.

Future work will focus on four key areas: 1) integrating on-device Small Language Models (SLMs) to allow users to trigger automations via natural speech or text; 2) extending the semantic model to recognize complex scrollable lists and dynamic text content using advanced OCR mapping; 3) introducing root-level execution capabilities to bypass standard Android accessibility constraints for deeper, low-level system integration; and 4) expanding the existing cross-device ecosystem by upgrading the desktop companion application to handle complex, system-level Windows workflows beyond its current browser-based limitations.

References

[1]. Al Zarooni, A., & El Khatib, M. (2023). Robotics Process Automation (RPA) and Project Risk Management. International Journal of Business Analytics and Security (IJBAS), 3(1), 77-85.

[2]. Auto Clicker. (2024). Auto Clicker - Automatic Tap. Google Play Store. https://play.google.com/store/apps/details?id=com.truedevelopersstudio.automatictap.autoclicker.

[3]. Cavsak, S. N., Deliahmetoglu, A., & Tanberk, S. (2023). GUI Component Detection Using YOLO and Faster-RCNN. International Conference on Electrical and Electronics Engineering.

[4]. Chen, K., et al. (2023). Vision-Based Mobile App GUI Testing: A Survey. arXiv preprint arXiv:2307.xxxx.

[5]. Crafty Apps EU. (2024). Tasker: Automation for Android. Google Play Store. https://play.google.com/store/apps/details?id=net.dinglisch.android.taskerm.

[6]. Cui, Y., & Chen, X. (2023). Design and Analysis of a Mobile Automation Testing Framework. 38th IEEE/ACM International Conference on Automated Software Engineering.

[7]. Deka, B., Huang, Z., Franzen, C., Hibschman, J., Afergan, D., Li, Y., Nichols, J., & Kumar, R. (2017). Rico: A Mobile App Dataset for Building Data-Driven Design Applications. Proceedings of the 30th Annual ACM

International Research Journal on Advanced Engineering and Management
https://goldncloudpublications.com
https://doi.org/10.47392/IRJAEM.2026.0093
e ISSN: 2584-2854
Volume: 04 Issue: 03
March 2026
Page No: 598 - 603

Symposium on User Interface Software and Technology (UIST), 845–854.
[8]. Google Developers. (2024). AccessibilityService Overview. Android Developers Documentation. https://developer.android.com/reference/android/accessibilityservice/AccessibilityService.
[9]. IFTTT Inc. (2024). IFTTT - Automation & Workflow. https://play.google.com/store/apps/details?id=com.ifttt.ifttt.
[10]. Jocher, G., Chaurasia, A., & Qiu, J. (2024). YOLO by Ultralytics. https://github.com/ultralytics/ultralytics.
[11]. Li, G., et al. (2023). Spotlight: Mobile UI Understanding Using Vision-Language Models. International Conference on Learning Representations (ICLR).
[12]. MacroDroid. (2024). MacroDroid - Device Automation. Google Play Store. https://play.google.com/store/apps/details?id=com.arlosoft.macrodroid.
[13]. Ribeiro, J., et al. (2021). Development of Intelligent Robotic Process Automation: A Utility Case Study. IEEE Access.
[14]. Xu, X., et al. (2025). Empowering Edge Intelligence: A Comprehensive Survey on On-Device AI Models. arXiv preprint arXiv:2503.06027.
[15]. Yang, Y., et al. (2025). GUIWatcher: Automatically Detecting GUI Lags by Analyzing Mobile Application Screencasts. arXiv preprint arXiv:2502.04202.
[16]. You, K., et al. (2024). Ferret-UI: Grounded Mobile UI Understanding with Multimodal LLMs. arXiv preprint arXiv:2404.05719.
[17]. Zhang, C., et al. (2023). AppAgent: Multimodal Agents as Smartphone Users. arXiv preprint arXiv:2312.13771.
[18]. Zhang, J., et al. (2023). VisionTasker: Mobile Task Automation Using Vision-Based UI Understanding and LLMs. arXiv preprint arXiv:2309.xxxx.
[19]. Zhang, L., et al. (2024). Robotic Process Automation Efficiency for Mobile App Testing: An Empirical Investigation. World Scientific.
[20]. Zhou, Z., et al. (2024). A Survey of Machine Learning in Edge Computing: Techniques, Frameworks, Applications, Issues, and Research Directions. MDPI.