Beyond Voice Commands: A comprehensive Visual-Audio Model for Mobile device Operation

1nd Dr. Gaurav Singal
Computer Science
Netaji Subhas University of Technology
New Delhi, India
gaurav.singal@nsut.ac.in

2st Tanmay Nagori
Computer Science
Netaji Subhas University of Technology
New Delhi, India
tanmay.nagori.ug21@nsut.ac.in

3rd Aatish Malik
Computer Science
Netaji Subhas University of Technology
New Delhi, India
aatish.malik.ug21@nsut.ac.in

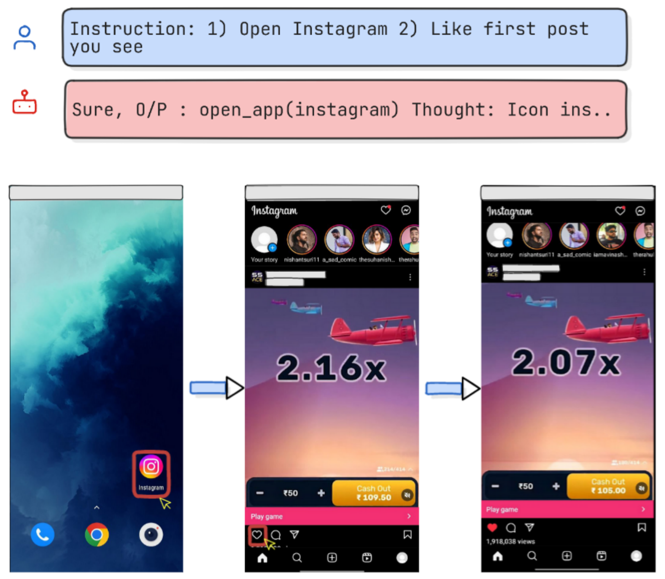
4rd Deepanshu
Computer Science
Netaji Subhas University of Technology
New Delhi, India
deepanshu-ug21@nsut.ac.in

***Abstract*—With the era of large language models on the horizon and the possibilities that come with it. We look at the future prospects of integration of voice assistants with large language models. This integration seems imminent due to the fact that most of the operations and tasks performed by human individuals are based on vision and the concept of voice assistants rely on the fact that we want to automate these tasks. Existing voice assistants rely on the technique of harnessing internal APIs such as XML files or using mobile metadata and scripts of the operating system. Our approach is an approach for integration of LLMs and voice commands. It first leverages the use of voice to text APIs and further uses a combination of LLMs inference capabilities to come up with a plan and then using single shot detection models to further execute the instructions by voice. Using the mixture of these state of the art, our model's approach can generalize a much wider range of instructions and autonomously execute them using the crafted execution engine. Hence, eliminating the need of system specific customizations. We further performed comprehensive evaluation to discuss the shortcomings as well as the accomplishment of our approach.**

***Index Terms*—voice assistant, automation, large language models**

I. Introduction

With LLM-based models already available Li et al. [7]; Yang et al. [16]; Shen et al. [20]; already demonstrates a strong reasoning capacity and long term task planning at disposal. And multimodal large language models liu, ye, dai, chen, bai; they all are excellent performers in their domain. However, there is still a major requirement for even better models which are much more economical, faster and cheaper. In this paper, we embarked on the path to better optimize this approach of task automation. The main focus of this paper is to produce a result which can be the pacemaker and path finder for the upcoming voice assistants. We firmly believe that with introduction for better and better models with smaller parameter size and better efficiency such as the Phi 1.5 [8] model which have a performance at par with much bigger models. The graph of improvement doesn't show any stagnation. The introduction of such models surely demonstrate that eventually we will see small form factors models running inference on mobile devices hardware to formulate a plan to execute user instruction and other vision based models trained to detect icons and OCR models would be a building blocks to the generalization of tasks completion through the use of voice assistants. During the writing of this paper, we observed the launch of products such as Humane AI Pin [13] and Rabbit [19] which popularized the term Large Language Action Models. We are on the same track,


Instruction: 1) Open Instagram 2) Like first post you see

Sure, O/P : open_app(instagram) Thought: Icon ins..

Fig. 1. A voice assistant that can execute tasks based on vision inference

however we firmly believe that all this advancements can be achieved through existing hardware and better models. A key focus in this paper was to demonstrate such working

possibility and further evaluate how far the existing state of the art models can characterize and plan the execution further into the future.
The Figure 1 shows an exemplar case where a normal voice assistants fails because of it's limitations to natural language understanding for the second command by the user as well as the lack of ability of current assistants to not be able to generalize the actions.
The use of the term General computer control (GCC) [18] is an accurate path the authors of this paper believe the coming technologies would head into. This GCC allows the use of existing technologies and without much modifications the natural language models with audio-visual capabilities can fit right into our world without much friction. GCC refers to mastering any task with just capturing the screen images and making inferences based on it to come up with a plan to execute instruction using human based inputs. Traditionally, voice assistants have operated within the confines of predefined scripts and limited APIs, tethered to specific systems and functionalities. However, with the advancement in LLMs a new horizon of possibilities emerged. This paper presents our approach to craft a unified framework capable of understanding a diverse array of instructions through voice commands and executing them autonomously. At the core of our method is the fusion of LLMs, voice to text state of the art models that can easily convert voice to text which is later on sent to LLMs to predict a plan and assist our execution engine in continuous stages of plan execution. The detectors such as icon detector and OCR (Object character recognition) classify the different texts and visual elements forming a coordinate store which assists our execution engine with knowledge of where to click or swipe as suggested by the LLM. This method is faster than other methods since it doesn't include the network costs and vision based inferences made by other multimodal large language models that require high bandwidth and compute power. Hence, making it extremely economical with most of the computation able to be handled by a small factor device such as a laptop.This introduction sets the stage for coming sections where we will first delve into related works to gain better understanding of existing works and ones we have drawn inspiration from, later to which we propose our architecture. A further model evaluation is done to determine its capabilities and shortcomings. This empirical validation will shed a light on the potential of our much cheaper and efficient method and approach.

Abstract
Introduction
Literature review
Methodology
Model Evaluation
Conclusions and Future work

Fig. 2. Table of content and paper organization

II. Literature Review

Agents built upon the large language model (LLM) have attended a remarkable performance. These large language models are based on advanced computer programs. LLM performs very efficiently in various tasks as referenced in [1]–[3], [6], [9], [10], [12], [14]. These programs support various operations. It easily understands user commands and has various types of tools to perform complex tasks. By the integration of various tools, LLMs have expanded their limits far beyond performing only processing tasks. Now the LLMs are used to perform various tasks like image diffusion, text recognition, generating visual and audio content, prediction, chatbot, answering questions about image etc. This makes us expand the limits of artificial intelligence, machine learning etc and use their intelligence to perform real time tasks and predictions more efficiently.

There is a Modelscope-agent [6] that can be used to build the customizable agent with open-source large language models. ModelScope-Agent brings us with Multimodal Large Language Models (MLLM) that can significantly contribute to the development of your Mobile-Agent. It helps us to build a general and customizable agent framework for real time event applications using open-source LLMs as controllers. So Mobilescope agent can enhance the vision centric approach in mobile devices.Hence ModelScope-Agent framework allows mobile apps to navigate through operations autonomously with enhanced accuracy and completion rates. The open-sourced code and models provided by ModelScope-Agent can serve as a foundation for your Mobile-Agent's development which paves the way towards a new era of AI Agent applications in mobile environments.

Another tool that is implemented and can help us with searching on the mobile screen includes the Control-LLM [12] tool that augments large language models with tools by searching on apps. This framework closely helps our Mobile agent in achieving its goals. It uses Multi model large language models

(MLLM) to autonomously navigate mobile applications. The controlLLM framework has the ability to accurately control tool usage like including text, image, audio, and video, etc,- which overall provides us with valuable insights to enhance the working of our Mobile-Agent.

Llava-plus [10] framework can be used for training to familiarize operational tools for creating agents with multiple modes of communication. The LLaVA-plus system, as disclosed in this paper, presents a coherent multimodal assistant based on large language and vision models (LMMs) that grows capabilities systematically through complete training. Among them include focus within your Mobile-Agent abstract on vision-centered autonomy and adaptability within different mobile operating systems. You can use the lessons from LLaVA-Plus in improving the Mobile-Agent by looking at its skill bank and activating relevant equipment based on multimodal input. If your Mobile-Agent is designed in this way it will adapt in real time and utilize numerous distinct tools for observing and executing duties. You can improve the visual comprehension, creation, and internal or ingested knowledge acquisition skills of your Mobile-Agent by using multimodal instruction-following data in its training, the same as in the case of LLaVA-Plus. the direct grounding of image queries in LLaVA-Plus throughout human-AI interaction sessions provides an effective model for actively engaging users and significantly improving tool use performance.To achieve new capabilities and meet challenges in the ever changing mobile device landscape, your Mobile-Agent could use a skill repository, multimodal instruction-following data and direct image queries grounding.

Visual instruction tuning [9] is another important aspect of our model. It demonstrated how visual instruction tuning functions. It introduced an automatic process which generates data for following instructions on images through speech; it was used as the basis for training LLaVA, an AI model that can execute visual commands issued by humans on the images . evolved as the best hyperplane available when ScienceQA is refined. It also performs excellently in visual chats based on multimodal chat datasets. To find more quantitative results of LLaVA on academic benchmarks, the improved baselines with visual instruction tuning were cited.

We also aimed for video object detection based on background subtraction. Video serial image processing [2], which focuses on moving segmentation and the extraction of representative bodies, holds significant potential for enhancing the capabilities of the Mobile-Agent. The Mobile-Agent can leverage visual perception tools more effectively, particularly in scenarios involving dynamic visual content within mobile applications. Its emphasis on segmenting correctly and extracting moving objects is in line with the Mobile-Agent aim to navigate mobile apps through operations on its own. We may extend the background extraction method to some of the tools used in the visual perception of the agent. The capability of the algorithm to make the extraction of a stable background using the gray characteristics of video images is in harmony with the vision-centric nature of the Mobile-Agent. Integration of this would highly augment the competence of the Mobile-Agent in the identification and localization of visual elements within the front-end interface of the app and further the agent's performance in the task of complex operation. This form of updating of the background by the algorithm according to the current frame further includes a mechanism of dynamic adaptation. The introduction of this dynamic background update process further enhances the adaptability of the Mobile-Agent to the changes that occur in the visual context of the mobile application and, hence, increases the reliability of its operations planning.

Consequently, embedding these video serial image processing methods in the tools of the Mobile-Agent's visual perception will increase the ability to effectively locate, identify, and navigate dynamically within mobile applications' visual elements. It justifies well within the vision-centric approach of the Mobile-Agent and meets the goals of your abstract more effectively, such as adaptability and autonomy in the complexity of various mobile operating environments.

To embed the speech recognition we pursue Learning the transferable visual models from natural language supervision. We can apply transferable visual models from natural language supervision to computer vision systems and their ability to learn image representations from raw text has a lot of potential to enhance the capabilities of the Mobile-Agent. The Mobile-Agent is designed as an autonomous multi-modal mobile device agent, with heavy reliance on visual perception tools to navigate mobile applications. Integrating the methods proposed in the computer vision paper, specifically the CLIP model, it can add to the Mobile-Agent capabilities in a more robust and versatile type of visual understanding.

The CLIP model [14] shows the effectiveness of pre-training on a large image-text pair dataset, making transfer to downstream tasks a zero-shot process. This is concomitant with the vision-centric approach of the Mobile-Agent, allowing it to correctly identify and locate both visual and textual elements within mobile app interfaces. The CLIP model's ability to draw on learned visual concepts using natural language can be easily integrated into the Mobile-Agent, making it more autonomous in planning and decomposing complex operation tasks. Aside from the zero-shot transfer capability of the CLIP model, it also means that the Mobile-Agent, with proper adaptation, can traverse through mobile applications even with no specific training on each app. This is, in particular, advantageous to achieve greater adaptability across diverse operating environments for mobile, without requiring heavy customizations for each system. By leveraging the open-source code and pre-trained model weights of the CLIP model, the Mobile-Agent will tap from an already large pre-learned pool of visual concepts, potentially resulting in higher accuracy and completion rates, especially in tackling complex instructions such as multi-app operations.

Now to parse the mobile screen text we used the Optical Character Recognition (OCR) technology [1], [5]. The OCR technology used in the paper is the fundamental tool for reading printed text and converting it into machine-readable

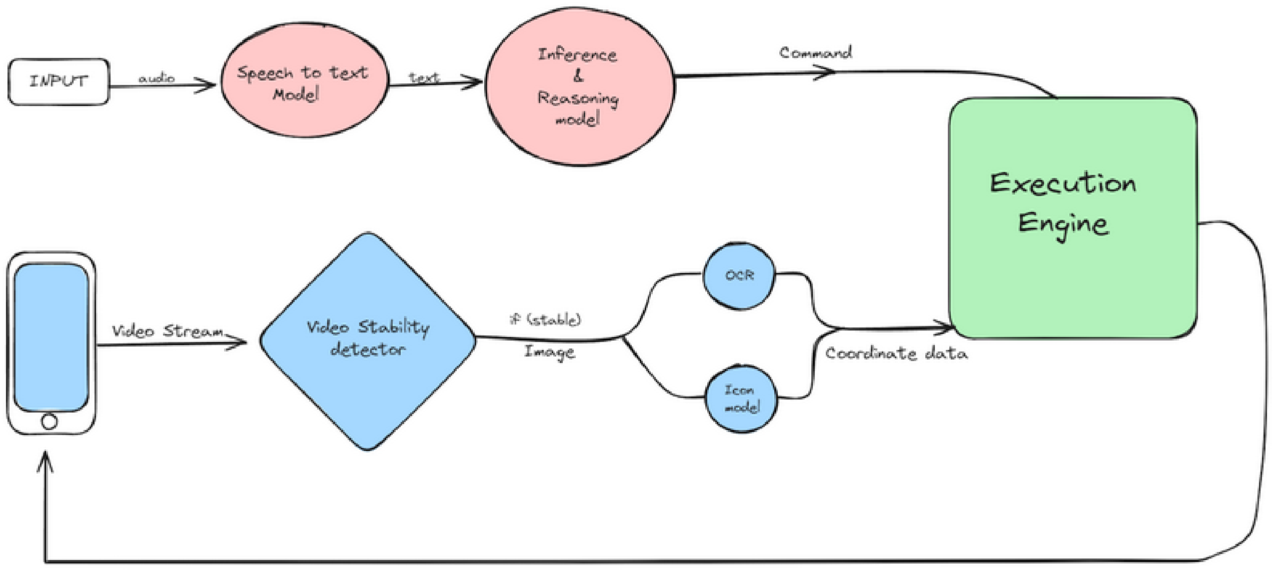


Fig. 3. The architecture of our model with it's components

code and storing the information in the system memory. With the capabilities provided by OCR, the Mobile-Agent can, on using OCR, further enhance its capability of understanding and interacting with textual information present in mobile applications, which gives it a more comprehensive and intelligent navigation process. The ability of OCR technology to convert printed text into editable text is in sync with the vision-centric nature of the Mobile-Agent, which allows it to plan and decompose complex operation tasks autonomously based on both visual and textual elements.

Furthermore, keeping in mind that OCR is in wide use to recognize text in scanned documents and images, the incorporation of this technology into your Mobile-Agent can facilitate the extraction of information from various sources, making it adaptive to the environment and different mobile operating systems. The ability of OCR to convert physical paper documents or images into accessible electronic versions supplements the Mobile-Agent's aim to achieve adaptability without being dependent on system-specific customizations.

Checking for the video stability is done so that there is not very fast transfer of the screenshots of the phone. The screenshot will only be taken when the screen is stabilized, that is the screen is not changing dynamically fast [3]. Video Stabilization enables us to enhance the capabilities of the Mobile-Agent specifically in the context of visual perception and task execution within mobile applications. Video stabilization is an important component of visual processing that can easily be integrated into the Mobile-Agent's framework to enhance its performance in identification and location of visual elements within mobile apps' front-end interfaces.

More robust visual perception can assist the Mobile-Agent in identifying and locating visual and textual elements in dynamic or shady environments. The Video Stabilization Quality Assessment (VSQA) methods can be adapted to the assessment of quality in visual inputs and would give a useful evaluation metric to the decision-making process of the Mobile-Agent. The challenges, practical aspects, and mathematical core concepts of video stabilization techniques, insights on this can be used to further enhance the vision-centric nature of the Mobile-Agent, so it can better autonomously plan and decompose difficult operation tasks. The IQA-inspired (Image Quality Assessment) methodology can be used to ensure that the visual perception tools used by the Mobile-Agent produce high-quality outputs.

In summary, this union of video stabilization techniques and quality assessment methodologies allows for greater accuracy and efficiency in navigation through mobile applications by the Mobile-Agent. The robust capabilities of visual processing derived from this paper can significantly enhance the Mobile-Agent's overall performance, making it more adaptive to different mobile operating environments and ensuring a smooth user experience.

III. Architecture

The architecture of our framework is one of the most important parts. It is designed to be a modular approach where we can change different parts of the framework with ease. The proposed architecture is comprehensive to execute voice commands on mobile devices through a multi-component pipeline. The architecture comprises several interconnected modules, each responsible for a very special task in the workflow. The primary components are as follows:

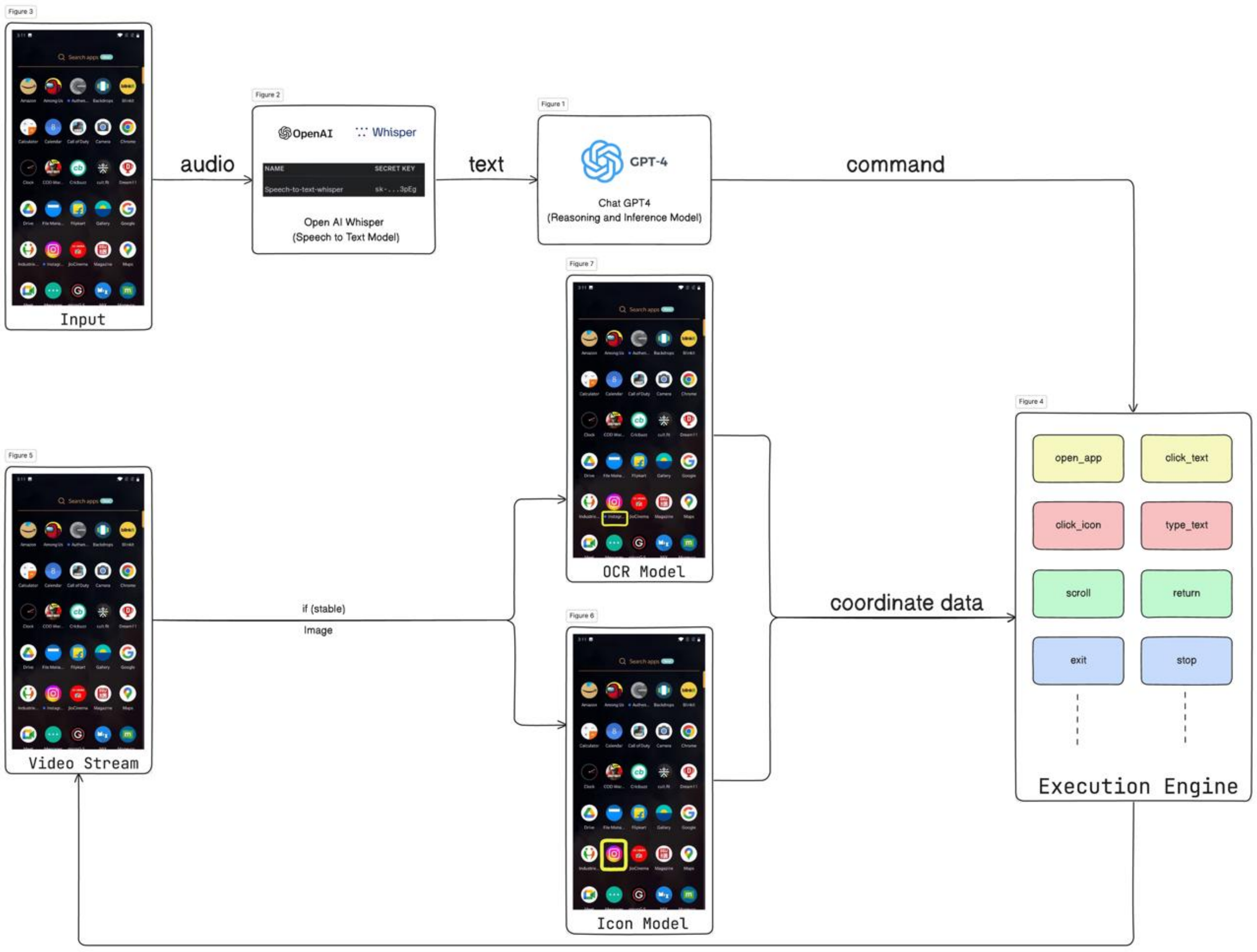


Fig. 4. The framework and it's working as proposed by the model of beyond voice commands for better efficiency and speed

A. Speech-to-text module

The audio component of the input MP4 file is processed by the speech-to-text module, which plays an important part such that it converts the audio data recorded by the command giver to a textual representation. This textual transcription is attained by state of the art models such as Whisper by OpenAI, the transcribing accuracy of whisper is 95% to 98% []. This transcribed data later is ingested by our inference and reasoning model and also a crucial step for the whole architecture.

B. Inference and Reasoning Module

This section is essentially a Large language models and is the core component which is responsible to formulate a plan for our appropriate course of action. At the heart of this module state of art models such as GPT4 or Claude AI Opus. In this study we have went with GPT4 for their easy integration and developer friendly SDK. These LLMs are trained on vast amounts of data to comprehend natural language adn in turn LLMs also show an exceptional thinking capability, this capability is enhanced even more with the prompt engineering concept of which we have used a specific prompt method. The prompt explains the model to only give a specific answer where it starts with a Thought process of why it thinks the task will be executed by the following command, we have also mentioned the model about how it can reply in only few commands as such actions and gestures are already predefined as capabilities of our execution engine. The nuances that the model have to go through had influenced our choice for a larger model at this instance of time but as we discussed that the motive of our paper is to run all these inferences locally and such capabilities are already demonstrated by smaller models which are hard to get hands but there parameter size and capabilities are noteworthy to be swapped in future works of this study.This module is designed to output one or more of such commands that can be executed by the subsequent components of the architecture.

C. Video Stability Detector

A parallel pipeline is present where the component video stability detector concurrently analyzes the video and picks up the frame only when there are enough changes to update our coordinate store. We employ the background subtraction technique with frame difference to figure out how much change is noticed on the screen. If the frame difference does not pass certain threshold then we do not run the subsequent pipeline which in turn saves our computation cost. The further inference models in this pipeline are although very fast and efficient but it was not helpful to run those models on each frame 60 times a second to update the location of the icons and text data on the screen.

D. OCR and Icon detection model

The object character recognition model is Tesseract and the one used for the icon detection model is based on the paper of Grounding DINO: Marrying DINO with Grounded Pre-Training for Open-Set Object Detection. [] The grounding dion model is a very efficient open-set object detector. It can detect arbitrary objects with human inputs such as category names or referring expressions. It was perfect for our use case without much training requirements.

E. Execution engine

The Execution Engine is the final component responsible for executing the commands provided by the Inference and Reasoning module. It acts as the bridge between the system's computational components and the physical mobile device, translating the interpreted commands into actual actions. It uses the coordinate data from the coordinate model to perform a wide range of actions such as opening applications, swiping, clicking text and clicking icons. Using the functions of open_app(app_name), swipe(bool left), click_text(string text), click_icon(string icon_name). This module is implemented using python and Android debug bridge (ADB) enabling us to remotely execute commands on our mobile devices while running the inferences on the laptop. It leverages the coordinate data to accurately target the desired icon or text. The architecture is designed with modularity and extensibility in mind, allowing for the integration of new components or the replacement of existing ones as technology advances. The flow of data and instructions between the various modules is very easily chageable,This all sets the groundwork for a model which is efficient and fast.

IV. Methodology

The methodology used in this research aimed to automate the user interactions taking place on the mobile application. Consider the above example in figure, the instruction is to open the instagram app and comment "Hello" on the first post. The instructions involved will be like opening of the app, accessing a particular post, and posting a comment. To attain all these, a step-by-step procedure was used in which different techniques and tools emulate the users' actions automatically.

The process forwards with the use of an icon detection model referred to as "grounding dino" [11], [17]. This model identifies the Instagram icon within the mobile device's screen. If there was no direct recognition of the icon, another procedure applied. The OCR technique was applied to parse the text on the screen and locate the name of the Instagram app. This model took an adaptive approach to ensure that it remained robust in cases of device configuration variations and variations in the appearance of the icon.

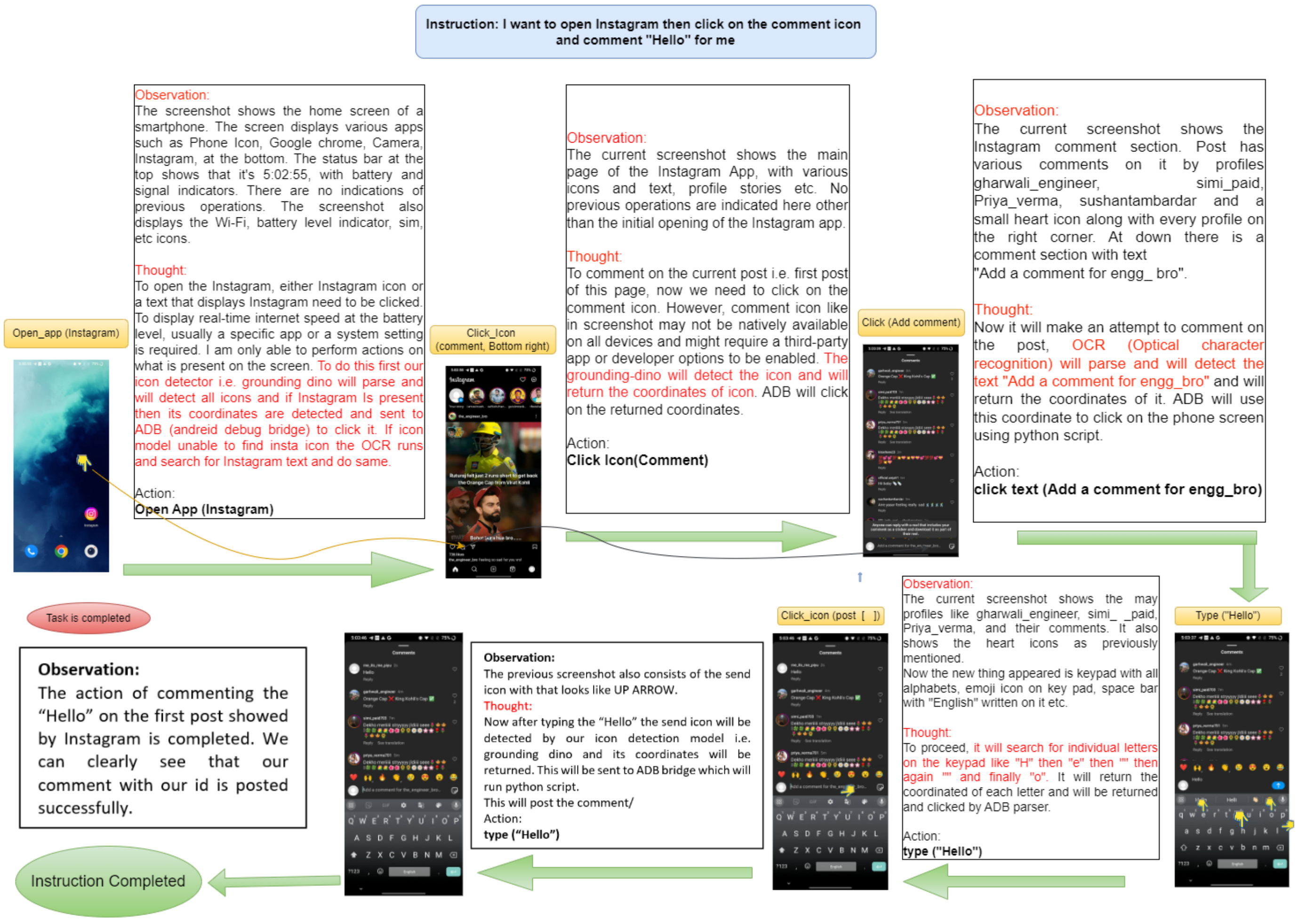
Once the application identification was successful, the next procedure from the methodology needed the application to navigate to the target post. The grounding dino was programmed to interact with the comment icon that usually appears on the main page as a speech bubble or as a similar graphical element. Special care was deployed with regard to the possible variations in the appearance and placement of the comment icon on various models of devices so that this methodology could be applicable universally.

In the comment section of the target post, the methodology applied the OCR to detect the existence of the comment text field. This comment text field is usually labeled "Add a comment," and it is the entry point of the user's input of the comment. Based on the obtained coordinates from the OCR result, the methodology applied the Android Debug Bridge, which simulates user interaction by clicking on the detected text field.

The next step was the typing of the desired text that the comment would contain. This was done by typing one character at a time, by locating the letter of the comment ("Hello") on the virtual keypad using OCR. OCR output was, in turn, coordinated with ADB commands to ensure that the typing was simulated appropriately, thus making input efficient.

After the typing of the text, the next procedure followed was that of posting the comment. The send icon—with its upwards arrow or other graphic representation—was identified using techniques for icon detection. Once this has been identified, the ADB [4], [15] was used to interact with the send icon, thus posting the comment to the target post effectively. The visual confirmation of the posted comment was a validation of the successful execution of the methodology.

Experiments were conducted to ensure the robustness and reliability of the automated process. This came in the form of testing the methodology on various device configurations and on different versions of the Instagram interface. Also we tested our model on different apps and for different features. The results were carefully documented, encompassing success rates, execution times, and variations observed. In addition to this, error handling mechanisms were integrated to address the failure to detect icons or text fields. The methodology in this study was able to provide a thorough and efficient way of automating user interactions on Instagram. By incorporating icon detection, OCR, and ADB, the process is able to replicate user actions with precision and reliability. Experiments and validation efforts conducted helped enforce effectiveness while also pointing out areas of further refinement and improvement


Instruction: I want to open Instagram then click on the comment icon and comment "Hello" for me

Observation:
The screenshot shows the home screen of a smartphone. The screen displays various apps such as Phone Icon, Google chrome, Camera, Instagram, at the bottom. The status bar at the top shows that it's 5:02:55, with battery and signal indicators. There are no indications of previous operations. The screenshot also displays the Wi-Fi, battery level indicator, sim, etc icons.

Thought:
To open the Instagram, either Instagram icon or a text that displays Instagram need to be clicked. To display real-time internet speed at the battery level, usually a specific app or a system setting is required. I am only able to perform actions on what is present on the screen. To do this first our icon detector i.e. grounding dino will parse and will detect all icons and if Instagram Is present then its coordinates are detected and sent to ADB (andreid debug bridge) to click it. If icon model unable to find insta icon the OCR runs and search for Instagram text and do same.

Action:
Open App (Instagram)

Observation:
The current screenshot shows the main page of the Instagram App, with various icons and text, profile stories etc. No previous operations are indicated here other than the initial opening of the Instagram app.

Thought:
To comment on the current post i.e. first post of this page, now we need to click on the comment icon. However, comment icon like in screenshot may not be natively available on all devices and might require a third-party app or developer options to be enabled. The grounding-dino will detect the icon and will return the coordinates of icon. ADB will click on the returned coordinates.

Action:
Click Icon(Comment)

Observation:
The current screenshot shows the Instagram comment section. Post has various comments on it by profiles gharwali_engineer, simi_paid, Priya_verma, sushantambardar and a small heart icon along with every profile on the right corner. At down there is a comment section with text "Add a comment for engg_ bro".

Thought:
Now it will make an attempt to comment on the post, OCR (Optical character recognition) will parse and will detect the text "Add a comment for engg_bro" and will return the coordinates of it. ADB will use this coordinate to click on the phone screen using python script.

Action:
click text (Add a comment for engg_bro)

Observation:
The current screenshot shows the may profiles like gharwali_engineer, simi_ _paid, Priya_verma, and their comments. It also shows the heart icons as previously mentioned.
Now the new thing appeared is keypad with all alphabets, emoji icon on key pad, space bar with "English" written on it etc.

Thought:
To proceed, it will search for individual letters on the keypad like "H" then "e" then "l" then again "l" and finally "o". It will return the coordinated of each letter and will be returned and clicked by ADB parser.

Action:
type ("Hello")

Observation:
The previous screenshot also consists of the send icon with that looks like UP ARROW.
Thought:
Now after typing the "Hello" the send icon will be detected by our icon detection model i.e. grounding dino and its coordinates will be returned. This will be sent to ADB bridge which will run python script.
This will post the comment/
Action:
type ("Hello")

Observation:
The action of commenting the "Hello" on the first post showed by Instagram is completed. We can clearly see that our comment with our id is posted successfully.

Fig. 5. The framework and it's working as proposed by the model of beyond voice commands for better efficiency and speed

in future research efforts.

V. Model Evaluation

In this segment, we aim to perform a thorough assessment of the visual-audio model. We've opted for the Android operating system for its user-friendly operation invocation interfaces. Although we currently focus on Android, we plan to extend our investigations to include other operating systems in subsequent research. Our study comprises two main components: quantitative experiments and qualitative analyses. Through quantitative experiments, we will scrutinize the visual-audio model using our devised benchmark. Meanwhile, the qualitative analyses will involve an in-depth examination of particular scenarios.

A. *Quantitative Analysis*

With the aim of extensively evaluating the capabilities of our approach, we introduce this visual-audio model, which has a benchmark based on current mainstream apps in table I. This evaluation methodology consists of a total of 6 commonly used apps on mobile devices. This benchmark is super important because it helps us measure how effective and efficient our approach is across different tasks that people commonly do on their phones.

Our evaluation method is pretty detailed. We're not just looking at one app at a time; we're also testing how our approach handles using two apps simultaneously. This is crucial because in real life, people often have to juggle multiple things on their phones at once.

To really understand how our approach is doing, we've set up three different tasks for each app. The first task is like a basic warm-up, focusing on simple things you'd normally do in the app. Then, we ramp up the difficulty in the second task by adding more stuff to do. And finally, in the third task, we're throwing our approach a challenge by giving it vague instructions and seeing how well it can figure out what to do on its own and make its independent judgement.

This way of testing not only tells us how good our approach is at following specific instructions but also how well it can adapt and think for itself in tricky situations. It's like putting our approach through its paces to see how it performs in the real world.

TABLE I
APPS WITH INSTRUCTIONS

S.No.	Application	Instructions
1	Chrome	1) Search result for today's IPL match. 2) Search the information about Virat Kohli. 3) I wish to know the result for today's IPL match. Find an app to help me.
2	Gmail	1) Send a subscription email to address. 2) Send an email invitation to address for my birthday party. 3) I wish to know let my sister know about the party and her address is address. Find an app to help me.
3	Spotify	1) Search album New Life in Spotify. 2) Search an album about ”spring” in Spotify and play it. 3) I wish to listen music to relax. Find an app to help me.
4	Google Play Store	1) Download Facebook in Play Store. 2) Download SonyLiv in Play Store. 3) I wish to use Facebook on my phone. Find an app to help me.
5	Settings	1) Turn on the Mobile HotSpot. 2) Turn on the airplane mode. 3) I wish to see the notifications for Facebook, please turn on this setting for me.
6	Instagram	1) Like a post on Instagram. 2) Comment on a post on Instagram. 3) Search for Virat Kohli on Instagram.

TABLE II
PERFORMANCE METRICS

Applications	Instruction 1			Instruction 2			Instruction 3		
	S	CO	CR	S	CO	CR	S	CO	CR
Chrome	✓	4/10	40.0%	✓	8/10	80.0%	✓	8/5	100%
Gmail	✓	4/10	40.0%	✗	3/10	30.0%	✗	7/10	70.0%
Spotify	✗	5/10	50.0%	✓	8/10	80.0%	✗	6/10	60.0%
Google Play Store	✓	3/10	30.0%	✓	4/10	40.0%	✓	3/10	30.0%
Notes	✓	7/10	70.0%	✓	6/10	60.0%	✓	6/10	60.0%
Settings	✓	4/10	40.0%	✓	4/10	40.0%	✓	5/10	50.0%
Instagram	✓	9/10	90.0%	✓	9/10	90.0%	✓	7/10	70.0%
Avg.	0.85	6/10	60.0%	0.85	7/10	70.0%	0.71	7/10	70.0%

B. Quantitative Metrics

We have come up with three metrics in Table II to assess the performance of the visual-audio model from different perspectives-

• Success (S): If the visual-audio model successfully executes an instruction, it is deemed successful.

• Accuracy Score (AS): This metric evaluates the correctness of the visual-audio model's execution across all instructions. It is calculated as the total number of correct steps divided by the total number of steps attempted.

• Consistency (CO): We manually execute each instruction for 10 iterations, initialising each iteration with a different starting point. Next step is to record the number of iterations required by our visual-audio model to successfully execute the instruction under trial. By comparing the visual-audio model success count with that of total number of iterations, we assess whether the visual-audio model utilizes the mobile device more efficiently.

• Completion Rate (CR): This metric quantifies the extent to which the visual-audio model can succeed in different iterations starting with different initial points. It is expressed as

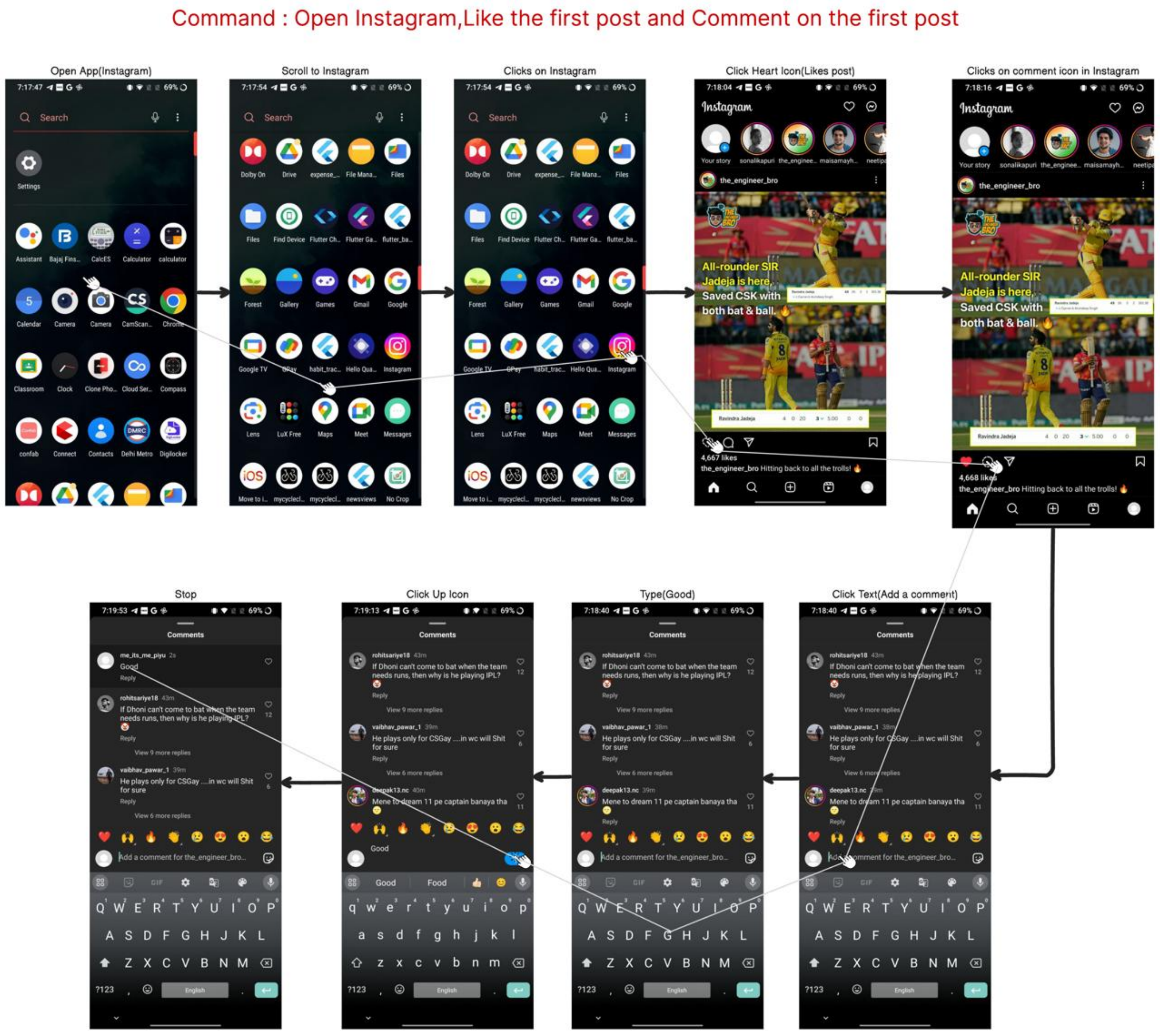

Command : Open Instagram,Like the first post and Comment on the first post
Open App(Instagram)
Scroll to Instagram
Clicks on Instagram
Click Heart Icon(Likes post)
Clicks on comment icon in Instagram
Click Text(Add a comment)
Type(Good)
Click Up Icon
Stop

Fig. 6.

the percentage of successful visual-audio model iterations to total given iterations. A completion rate of 100% indicates that the instruction was fully executed by the visual-audio model in every single iteration.

The overall evaluation results of this model are presented ahead in the Average row in Table II.

C. Quantitative Results

We present the experimental findings detailed in Table II. Initially, across the three sets of instructions, our visual-audio model demonstrated completion rates of 60%, 70%, and 70% correspondingly. Although not all instructions were executed successfully, the completion rates for all three instruction types exceeded 60%. Additionally, based on the S metric, our model exhibited a high likelihood of executing operations correctly across the board, achieving an average of approximately 80%. Furthermore, the CO metric indicates that visual-audio model can achieve a consistency level of 0.7 over multiple distinct iterations. These findings collectively underscore the effectiveness of our model as a mobile device assistant.

However, it's imperative to acknowledge that certain instructions did not achieve a perfect S value, denoted by the absence of a correctly ticked mark. This observation suggests that our model may occasionally falter, manifesting in the performance of invalid or erroneous operations. Nevertheless, it's noteworthy that even in such instances, the majority of instructions were ultimately completed successfully. This phenomenon underscores the robust self-reflective capabilities

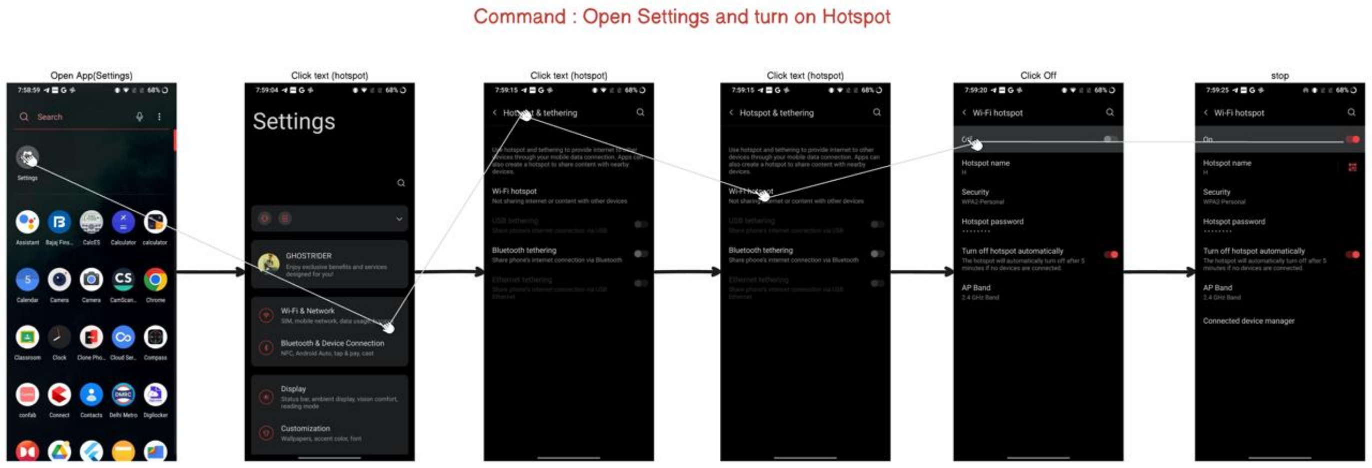


Fig. 7.

inherent within our model. Equipped with the ability to analyze screenshots and rectify errors accordingly, our model demonstrates resilience in the face of challenges, mirroring the adaptive nature of human problem-solving. This inherent capacity for error correction is crucial for mobile device agents, as it acknowledges the inherent uncertainty and complexity inherent in operational environments, necessitating adaptive strategies for better and effective performance.

D. *Tasks details*

- **Figure 6 Open Instagram, Like the first post and comment "Good" on the first post** The model searches for Instagram using OCR as well as icon model and scrolls through the apps, on not finding it.When it finds the Instagram app, it opens the app and likes the first picture by finding the heart icon as it is specified as like by the LLM model. Then it finds the comment icon to comment on the post and clicks on add a comment to type "'good" on it. After typing the comment,it clicks on the send button and stops.
- **Figure 7 Open Settings and turn on Mobile Hotspot** The model searches for the settings icon , opens settings and then searches for hotspot and on finding it clicks on it. Then on the next screen it again searches for hotspot and find it in the title page but as it is not a clickable button it again searches for hotspot and this time clicks on it. Then it clicks on off to turn hotspot on and stops.
- **Figure 8 Open Gmail and compose a mail to another mail(mention the mail here aatish…xxxx)** The model searches for Gmail and scrolls through the apps, not finding it. On not finding it scrolls and finds the app. Then it opens the app. In the app it searches for compose button to compose a mail and clicks on the To , to write the mail whom to write to then writes the message and presses on the send button and stops.
- **Figure 9 Search for todays ipl match on Chrome** The model searches for Chrome and on finding it , opens the app. In the app it searches for Search text and clicks on it to type "todays ipl match". It displays the matches and stops.
- **Figure 10 Open PlayStore and install Facebook** The model searches for PlayStore and scrolls through the apps, not finding it. On not finding it scrolls and finds the app. After opening it finds the search icon and clicks on it. Then it types "Facebook" and clicks on the search icon in the keyboard to start the search. On finding the search results , it clicks on install to finally install the app and stops.
- **Figure 10 Open Spotify and search for album new life** The model searches for Spotify and on finding it, opens the app. After opening it finds the search icon and clicks on it. Then again on the search page it finds the search icon and clicks on it to type "new life" album to search it and clicks on search button on keyboard to search. In the search results it searches for the album name and clicks on it to open the album.

VI. Conclusions and Future work

We introduce to you an autonomous multi-modal agent, adept in operating a large number of mobile applications proficiently through a unique approach of recognition on the go within the device and using inference from LLM models for higher accuracy, called Beyond Voice Commands through this work . It deploys unique text and icon detection models on-device to make the task of locating and identifying visual elements fast and more accurate. It plans and simplifies complex tasks using LLM models and acts on them using the on-device Text and ICon models to navigate through them one task at a time. Distinct from other models using device specific techniques and models, it uses a vision based model to perform tasks on any environment. Through various experimentation, we showcase its effectiveness and versatility across various platforms and environments. This demonstrates its potential as an efficient and adaptable solution for interaction with mobile applications in a language independent manner.

Command: Open the gmail app and compose a mail to mail id "↗ aatish.ugxxx.xxx.com" posting "How are you my boy?".

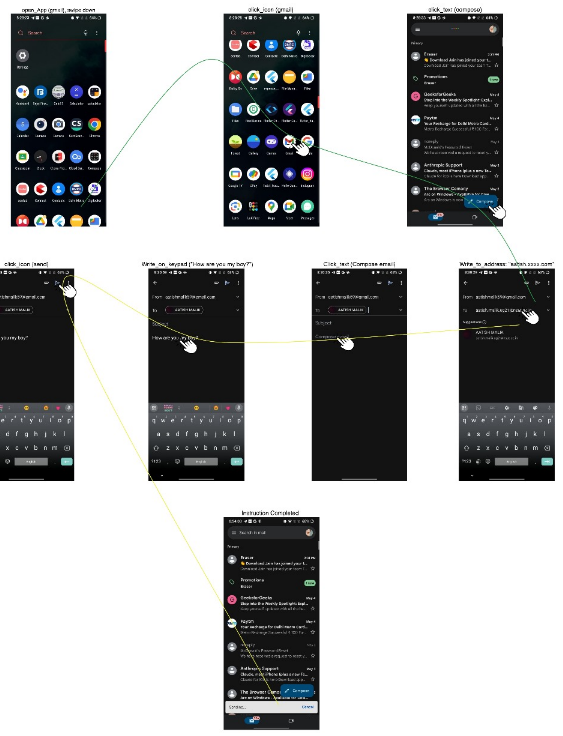

open_App (gmail), swipe down
click_icon (gmail)
click_text (compose)
click_icon (send)
Write_on_keypad ("How are you my boy?")
Click_text (Compose email)
Write_to_address: "aatish.xxxx.com"
Instruction Completed

Fig. 8.

Command : Search for todays ipl match on Chrome

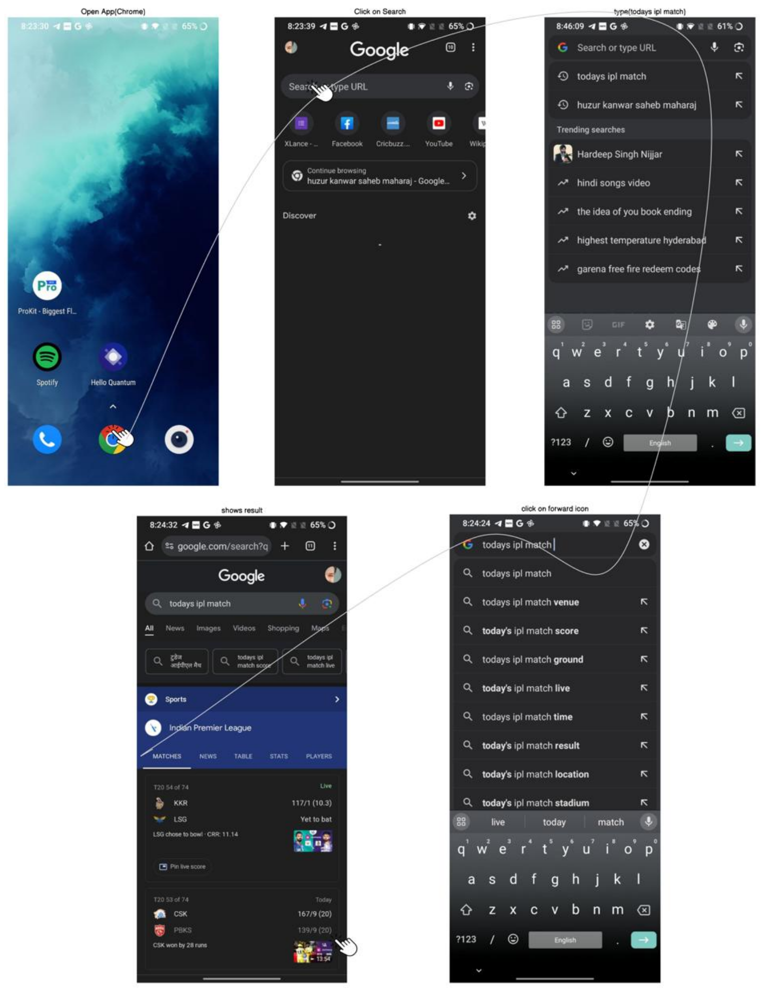


Fig. 9.

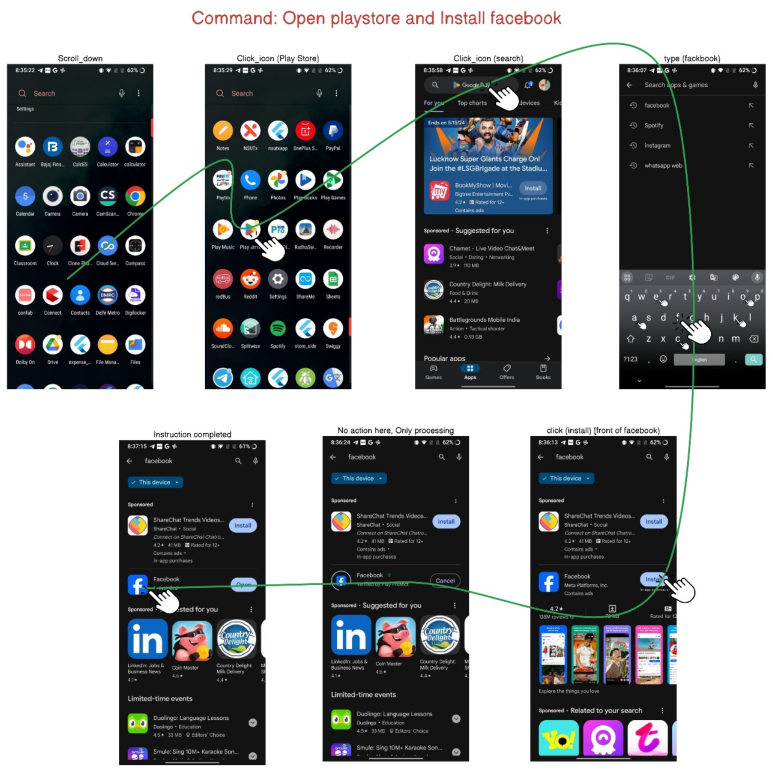

Command: Open playstore and Install facebook
Scroll_down
Click_icon (Play Store)
Click_icon (search)
type (fackbook)
Instruction completed
No action here, Only processing
click (install) [front of facebook)

Fig. 10.

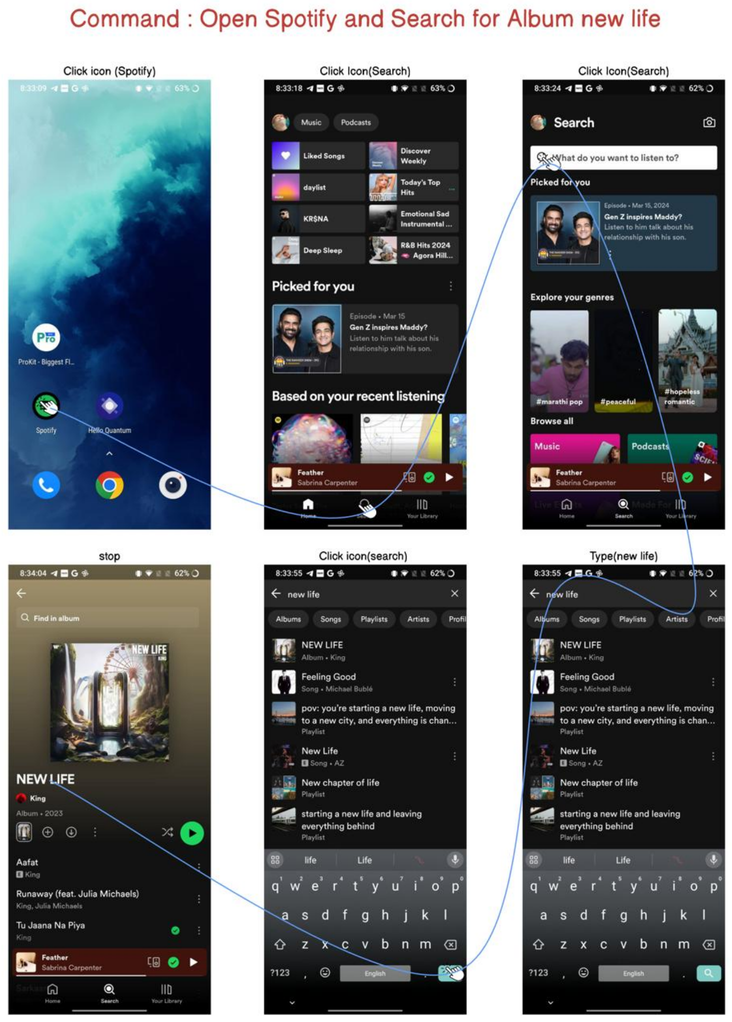

Command : Open Spotify and Search for Album new life
Click icon (Spotify)
Click Icon(Search)
Click Icon(Search)
stop
Click icon(search)
Type(new life)

Fig. 11.

We can continue to push the boundaries of autonomous multi modal agents and let them bloom to their fullest potential and revolutionize the way mobile devices are interacted with. This could be achieved by integrating text as well as icon models onto the device itself. But as the task of making large machine learning and deep learning models available on a device seems too far fetched, we could use distributed and cloud based computing by spreading different models onto different devices on the cloud and using each model from each device to perform different tasks and passing the result from each model to the next model.This makes the processing faster as well as light on the user device. In the future when we are able to deal with more complex inputs such as gestures,movements, which require multiple cameras as well as different input devices to capture exact motion then they could also be further integrated into the model. Future technology like AR, VR which are emerging as we talk and use more of these inputs can use this model to further increase their usefulness and use cases, unlocking new possibilities for user interaction and user experience improvements. Implementing LLM models on-device in the future could also help in improving the performance of the model over time. Expanding the horizon of the current systems we can also expand language support for voice as well as text commands by using improved Natural Language Processing(NLP) Models. It will make it accessible and improve its utility on a global scale. Ensuring robust privacy and security measures is crucial as with any AI-driven system. Future work should focus on implementing mechanisms protecting user data and ensuring secure interactions with mobile applications. Exploring methods for scaling the agent to support a larger number of applications and devices, as well as optimizing deployment strategies for efficiency and accessibility, will be important for widespread adoption.

These all use cases when implemented would revolutionize the world as we see it and make interacting with machines more human-like and help us in catering the needs of all kinds of people with the utmost care.

REFERENCES

[1] Opical character recognition. https://www.researchgate.net/publication/360620085_OPTICAL_CHARACTER_RECOGNITION.

[2] Video object detection based on background subtraction. https://www.researchgate.net/publication/289805044_Video_object_detection_based_on_background_subtraction.

[3] Video stabilization: overview, challenges and perspectives. *ScienceDirect*, 2023.

[4] Chuck Easttom and Willie Sanders. On the efficacy of using android debugging bridge for android device forensics. In *2019 IEEE 10th Annual Ubiquitous Computing, Electronics Mobile Communication Conference (UEMCON)*, pages 0730–0735, 2019.

[5] Noman Islam, Zeeshan Islam, and Nazia Noor. A survey on optical character recognition system. *arXiv preprint arXiv:1710.05703*, October 2017.

[6] Chenliang Li, Hehong Chen, Ming Yan, Weizhou Shen, Haiyang Xu, Zhikai Wu, Zhicheng Zhang, Wenmeng Zhou, Yingda Chen, and Chen Cheng. Modelscope-agent: Building your customizable agent system with open-source large language models. *arXiv preprint arXiv:2309.00986*, 2023.

[7] Chenliang Li, Hehong Chen, Ming Yan, Weizhou Shen, Haiyang Xu, Zhikai Wu, Zhicheng Zhang, Wenmeng Zhou, Yingda Chen, Chen Cheng, et al. Modelscope-agent: Building your customizable agent system with open-source large language models. *arXiv preprint arXiv:2309.00986*, 2023.

[8] Yuanzhi Li, Sébastien Bubeck, Ronen Eldan, Allie Del Giorno, Suriya Gunasekar, and Yin Tat Lee. Textbooks are all you need ii: phi-1.5 technical report, 2023.

[9] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *arXiv preprint arXiv:2304.08485*, 2023d.

[10] Shilong Liu, Hao Cheng, Haotian Liu, Hao Zhang, Feng Li, Tianhe Ren, Xueyan Zou, Jianwei Yang, Hang Su, and Jun Zhu. Llava-plus: Learning to use tools for creating multimodal agents. *arXiv preprint arXiv:2311.05437*, 2023c.

[11] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, and Lei Zhang. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. *arXiv preprint arXiv:2303.05499*, March 2023. Code will be available at https://example.com.

[12] Zhaoyang Liu, Zeqiang Lai, Zhangwei Gao, Erfei Cui, Zhiheng Li, Xizhou Zhu, Lewei Lu, Qifeng Chen, Yu Qiao, and Jifeng Dai. Controlllm: Augment language models with tools by searching on graphs. *arXiv preprint arXiv:2310.17796*, 2023a.

[13] Alex Mathew. Humane ai pin - innoglove ai embrace. *International Journal of Frontiers in Medical Research (IJFMR)*, 5(6), November–December 2023.

[14] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, and Jack Clark. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[15] Rajaram Regupathy. Android debug bridge (adb). In *Unboxing Android USB*, pages 125–138. Apress, 2014.

[16] Weizhou Shen, Chenliang Li, Hongzhan Chen, Ming Yan, Xiaojun Quan, Hehong Chen, Ji Zhang, and Fei Huang. Small llms are weak tool learners: A multi-llm agent. *arXiv preprint arXiv:2401.07324*, 2024.

[17] Jinhwan Son and Heechul Jung. Teacher–student model using grounding dino and you only look once for multi-sensor-based object detection. *Appl. Sci.*, 14(6):2232, 2024. Department of Artificial Intelligence, Kyungpook National University, Daegu 41566, Republic of Korea. Author to whom correspondence should be addressed.

[18] Weihao Tan, Ziluo Ding, Wentao Zhang, Boyu Li, Bohan Zhou, Junpeng Yue, Haochong Xia, Jiechuan Jiang, Longtao Zheng, Xinrun Xu, Yifei Bi, Pengjie Gu, Xinrun Wang, Börje F. Karlsson, Bo An, and Zongqing Lu. Towards general computer control: A multimodal agent for red dead redemption ii as a case study, 2024.

[19] Wired. Review: Rabbit r1, Year the article was published, if available.

[20] Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Ehsan Azarnasab, Faisal Ahmed, Zicheng Liu, Ce Liu, Michael Zeng, and Lijuan Wang. Mm-react: Prompting chatgpt for multimodal reasoning and action. *arXiv preprint arXiv:2303.11381*, 2023.