Jilin Daxue Xuebao (Gongxueban)/Journal of Jilin University (Engineering and Technology Edition)
ISSN：1671-5497
E-Publication: Online Open Access
Vol: 42 Issue: 04-2023
DOI 10.17605/OSF.IO/2XBPZ

GUIDE- A TOOL TO DETECT THE GUI ELEMENTS TO ENHANCE THE EFFICIENCY OF TESTING

MUHAMMAD AWAIS
Department of Software Engineering, Superior University, Lahore, Pakistan.
Email: SU92-MSSEW-F22-002@superior.edu.pk

MUHAMMAD TAHIR
Department of Software Engineering, Superior University, Lahore, Pakistan.
Email: msse-f21-016@superior.edu.pk

RABIA FARID
Department of Software Engineering, Superior University, Lahore, Pakistan.
Email: rabiafarid111@gmail.com

SALEEM ZUBAIR AHMED
Department of Software Engineering, Superior University, Lahore, Pakistan.
Email: saleem.zubair@supeior.edu.pk

MUHAMMAD WASEEM IQBAL
Department of Software Engineering, Superior University, Lahore, Pakistan.
Email: waseem.iqbal@superior.edu.pk

KHALID HAMID
Department of Computer Science, Superior University, Lahore, Pakistan. Email: Khalid6140@gmail.com

Abstract

The most difficult and critical part of graphical user interface testing is to detect the graphical component of a graphical user interface. To conduct GUI testing or reverse engineering for the GUI interface, the first step is to detect the classes of GUI elements and their exact positions. In this article, we implement a web-based tool to implement graphical user interface element detection named GUIDE. This platform enables its users to detect the classes of graphical user interface testing along with their exact positions by following a very simple series of steps. To deal with complex and diverse images, this toolkit uses a combination of both traditional computer vision methods and models of deep learning. Moreover, to produce an accurate result, this toolkit also has a unique method. It also provides the facility of the dashboard where users can edit or change the results of testing. It also facilitates its user to export GUI classes and images in the form of design files. These files can be further modified in graphical design tools like Photoshop. Overall, this toolkit provides accurate detection and is best to utilize in stream works.

Keywords: GUI, Deep Learning, UI Elements, Testing, Element Detection, GUIDE

1. INTRODUCTION

You can see data and interact with apps through the use of widgets, graphics, and text when you have a graphical user interface (GUI). Creating a good user interface may be difficult at times, but it is really necessary. Work that is labor-intensive and repetitive includes tasks such as testing user interfaces and implementing programs that use a graphical user interface (GUI). Research on GUI automation and testing is now being conducted, and continual efforts are being made to expedite the development process

Jilin Daxue Xuebao (Gongxueban)/Journal of Jilin University (Engineering and Technology Edition)
ISSN：1671-5497
E-Publication: Online Open Access
Vol: 42 Issue: 04-2023
DOI 10.17605/OSF.IO/2XBPZ

and lighten the load placed on engineers [1]. For this job, you must be familiar with the various GUI components. It is possible to locate GUI components using either the instrument-based method or the photo-based method, both of which are distinct from one another. Access to the backend system is necessary for the execution of intrusive scripts that are employed by instrumentation-based approaches. However, if you do not have access to the source code, it is time-consuming to write scripts for them, and it is tough to work with them [2].

Image-based techniques, on the other hand, need only a picture of a graphical user interface (GUI) to determine which elements of the GUI it is. This makes image-based techniques more comprehensive and less invasive. We are not aware of any straightforward and dependable way that users can use to gain knowledge regarding GUI components. We developed web-based, interactive computer vision tools known as Graphical User Interface Element Detection to easily detect and manage GUI elements in GUI images [3]. These tools fall under the category of computer vision (GUIDE). Users can recognize graphical user interface features by uploading their images into the straightforward web application known as GUIDE and receiving accurate results. Finding your way through graphical user interface pictures is a lot like looking for things in the real world. The process involves assigning a category to an item that may be observed in a digital photograph or video that was captured against a natural background [4]. Similarly, our GUI element identification engine can examine a GUI picture such as a screenshot or a design sketch to recognize and retrieve GUI widgets, images, and text. GUIDE utilizes a total of five cutting-edge algorithms, three of which are deep learning methods, and two computer vision approaches that have since been rendered obsolete (Faster-RCNN, Yolo v, CenterNet). On the other hand, graphical user interfaces frequently contain a large number of scenes, objects, and other components [4]. Nevertheless, the graphical user interface elements themselves come in a wide variety of forms. Due to the aforementioned factors, the aforementioned procedures are insufficient for the reasons for which they were developed. As a result, we offer an innovative method for recognizing GUI elements by making use of the techniques that are often associated with computer vision. Those approaches that make use of text, on the one hand, and those that don't, on the other, are the two primary classifications that can be applied to the many methodologies [5].

Image processing techniques such as flood-fill and linked component labeling are utilized first when attempting to recognize non-textual things for classification. The information is subsequently categorized with the help of a ResNet50 classifier. In contrast to previous methods, which used a bottom-up edge/contour aggregation method, our approach takes into account the multiple edges, forms, textures, and layouts of graphical user interfaces (GUIs). Second, the most cutting-edge deep learning EAST scene text model was utilized for the construction of the text components [6]. Our method has shown to be the most successful one for recognizing GUI elements since it combines cutting-edge and tried-and-true methodologies with a classifier based on deep learning. Users can change the output by rearranging the position, size, shape, and visibility of the GUIDE parts by using the interactive dashboard that is provided by GUIDE. The programmer creates a repository of reusable user interface kits by

Jilin Daxue Xuebao (Gongxueban)/Journal of Jilin University (Engineering and Technology Edition)
ISSN：1671-5497
E-Publication: Online Open Access
Vol: 42 Issue: 04-2023
DOI 10.17605/OSF.IO/2XBPZ

collecting all of the components that have been discovered. When the user is pleased with their edits, they have the option to export both the updated GUI as well as the element data related to that GUI from GUIDE (e.g., position, size, class, etc.) [7].

Two methods that can be used to improve the output are testing using graphical user interfaces (GUIs), as well as using software known as UI2CODE, which tries to automate GUI design by writing code directly from GUI graphics. This paper makes the following contributions, to detect GUI elements, we use our method for finding GUI elements and five other methods for finding things, we offer GUIDE, an interactive web application that lets users control GUI elements and create outputs for further development that can be used again and again, and expert research shows how important a reliable GUI element recognition method is.

2. LITERATURE REVIEW

It is necessary to go through the steps of designing and testing brand-new software applications. This, in turn, lowers the possibility that inappropriate behavior would occur, which, in turn, raises the bar for productive behavior. It is standard procedure to run tests by hand, with a human operator making use of the graphical user interface (GUI) (GUI). In this step, testers make use of their experience as well as their instincts to hunt for problems. This is a process that is both expensive and time-consuming. It is possible to quickly check the application under test (AUT) by executing a large number of test cases with the use of test scripts; nonetheless, it is common for hidden flaws to go unnoticed. To build useful tests, you need a deep understanding of the underlying system as well as strong programming skills. The implementation of these tactics is not only difficult and time-demanding, but with each new iteration, they fall farther and further behind [1].

Academics are interested in model-based testing (MBT) for software-intensive systems, but commercial adoption has lagged. Our business partners have noticed MBTs using more graphical models. They blame the lack of evidence-based MBT recommendations that consider technical and non-technical factors. Since there is no scientific direction, this idea is more likely to be accurate. This study consults MBT industry experts to learn how to apply graphical modeling to MBT procedures. The findings should inspire scholars and practitioners. The research is based on semi-structured in-depth interviews with 17 MBT professionals from around the world in various software industry professions. Semantic equivalence analysis compiles interview data, which MBT specialists with business expertise double-check. Thirteen sets of findings generate twenty-three recommendations for accepting, using, and quitting a strategy. We evaluate why the industry hasn't widely adopted MBT with graphical models using factual evidence and industry experts' viewpoints. After reviewing the results, you will receive structure and approach remarks. A business needs structure, mission, knowledge, and resources to implement a technique. This study advances the state of the art and helps experts employ graphical models to improve MBT effectiveness [2]. It is the responsibility of the users to notify the developers of any issues with the product as soon as they become aware of them. In contrast, non-technical users frequently lack

Jilin Daxue Xuebao (Gongxueban)/Journal of Jilin University (Engineering and Technology Edition)
ISSN：1671-5497
E-Publication: Online Open Access
Vol: 42 Issue: 04-2023
DOI 10.17605/OSF.IO/2XBPZ

the terminology required to provide good defect descriptions. [Citation needed] [Citation needed] Screen recordings are gaining popularity as a method of bug reporting due to their ease of use and the fact that they provide step-by-step instructions on how to duplicate the issue being reported. Despite the length of the tape and the haziness of the movements, the developers are still having a difficult time and spending a lot of time trying to find out how to make the problem happen again. We advise using GIFdroid as a quick remedy rather than manually generating the execution trace from visual bug reports. This will save a significant amount of time. GIFdroid uses image processing to extract key frames from the recording and then maps those key frames to states in the GUI Transitions Graph to establish which state was the cause of the problem. This allows GIFdroid to determine which state was the cause of the problem. Both automated and human testing have demonstrated that the procedure is accurate, useful, and effective in achieving its intended results [3].

Automated software testing makes meeting delivery deadlines easier. CI/CD pipelines have made automation solutions less useful. The testing business is considering artificial intelligence to stay pace with technological advancement. We investigate how transformers can generate test cases from UI element descriptions. This replaces manually extracting test cases from the test specification paper. The proposed strategy can also enhance failed testing. We use EfficientDet and Detection Transformer to find things in an application's user interface. Thus, a tester need not actively search for perfectly created user interface pieces. Tesseract automatically identifies UI component text. GPT-2 and T5, among others, translate UI element descriptions into test cases. Test scripts are created from test cases using a simple parser. This case study uses thirty online buying sites. Detection Transformer and EfficientDet create 98.08 and 93.82 percent valid, executable test cases, respectively. The framework reduced app instability by 96.05% across all test apps. The suggested strategy will improve industry-standard automation tools' test case generation and test improvement. [4].

Quality-of-Experience (QoE) is a metric that every mobile network provider is required to track, and it is measured concerning established Quality-of-Service criteria. User feedback on numerous stimuli or variable-processed sources serves as the foundation for subjective quality-of-experience assessment methodologies such as the Multi-Stimulus test with Hidden Reference and Anchor, or MUSHRA, and the Subjective Assessment Methodology for Video Quality (SAMVIQ). The process of producing stimuli that are following a set quality of service standard is labor- and time-intensive, but it is vital for determining the level of quality of the user experience provided by an application. In light of this, we propose a fresh method, one that is driven by the participation of the community, for the creation of stimuli for measuring the level of experience provided by mobile apps. It does this by autonomously manipulating the user interface of a real-world mobile device and monitoring the activities of the device through a network. This is done to produce the proper stimulus. The article begins with an explanation of the framework, continues with a discussion of its open-source implementation, and comes to a close with a concise summary of the framework concerning several different kinds of mobile applications [5].

Jilin Daxue Xuebao (Gongxueban)/Journal of Jilin University (Engineering and Technology Edition)
ISSN：1671-5497
E-Publication: Online Open Access
Vol: 42 Issue: 04-2023
DOI 10.17605/OSF.IO/2XBPZ

The production of iron in a blast furnace is strongly reliant on continuous quality control as well as inspections of the smelting condition. A critical phase involves utilizing data-driven models to make predictions about molten iron quality (MIQ) indices. Nevertheless, the MIQ forecast must still overcome a few obstacles: It is challenging to get a hold on data-driven models since there is a high demand for labeled samples, there hasn't been enough research done on the links between MIQ indices, and there is a dearth of individuals who are skilled in the nonlinear and dynamic description. Our original, data-driven, and deep model for online prediction of MIQ indicators is shown here. To get started, we create a focus-splitting module that will enable individual inquiry into the dynamic and nonlinear connection that exists between process variables and prediction targets. To achieve superior weights and rely less on labeled samples, the attention-wise deep network is pertained using the granular molten iron temperature (MIT) data obtained by our custom-built apparatus. This is done to produce accurate results. The capabilities of the model are then increased by adding a framework to it. This framework includes a weighted attention-wise module as well as task-separate prediction networks. This is done to examine the dynamic interplay that exists between the many different sorts of prediction. We gave the suggested attention-wise deep network a workout at a commercial iron foundry to test its capabilities. The results are now substantially more accurate and easy to understand than they were previously [13].

The fact that mobile computing is undergoing development and the market requires new products to satisfy the requirements of an increasing number of customers highlights the significance of assuring the quality of mobile applications. Testing of graphical user interfaces (GUIs) is now the subject of research that has just begun. According to the findings of recent studies, it is still difficult to acquire an exhaustive list of operations, transitions, functionality coverage, and reproduction failures. Deep-GUIT is a tool that is built on a Deep Q-Network, and the authors of this study recommend utilizing it to build test cases for Android mobile apps in a way that encourages discovery through the use of code coverage and inventive exercises. To evaluate the technology, fifteen mobile applications that are freely available to the public were used. Code coverage and failure rates were found to be higher and more frequent, respectively, for the new tools Monkey (average increase of 61%) and Q-testing (average increase of 47%) [7]. It is critical to conduct GUI testing on every piece of GUI software. The quality of software degrades as a result of automated GUI testing that is difficult to reuse and cannot be deployed portably because it maintains track of coordinates and handles. The lightning-fast rate at which graphical user interface (GUI) software develops highlights the pressing need for improved testing procedures for GUIs. The testing procedure has the potential to be vastly improved and completed much more quickly if automated intelligence is utilized. But to test graphical user interfaces, you need to understand how their components work. Deep learning is being investigated in this study as a potential method for intelligently recognizing graphical user interface (GUI) elements. It not only generates datasets that may be used to train a model that is capable of recognizing GUI elements, but it also provides instructions for how those elements should be marked up. We train a model to recognize GUI elements using the network that we used for object detection, and then we discuss the results of this training. The results of the project offer research

Jilin Daxue Xuebao (Gongxueban)/Journal of Jilin University (Engineering and Technology Edition)
ISSN：1671-5497
E-Publication: Online Open Access
Vol: 42 Issue: 04-2023
DOI 10.17605/OSF.IO/2XBPZ

items and data about the reliability of intelligent systems, in addition to providing technical support for testing intelligent graphical user interfaces (GUIs) [8].

The software used for aircraft control is, by a significant margin, the most important aspect of aviation software. The quality standards are fairly rigorous because any faults could put the lives and safety of other people in jeopardy. The process for testing software needs to be dependable and effective. Because the conventional techniques of testing were insufficient for the flight control software, automation testing has gradually replaced the traditional ways as the major means of checking the program. This is because traditional testing methods were inadequate. Locating and comprehending the operation of GUI widgets on software screenshots presents the greatest challenge when it comes to automating the testing of aircraft control software. This has an immediate and significant bearing on the validity of the examination. To determine the likeness of the widget contained within the screenshots that were taken, we make use of picture recognition and matching techniques. When we take a picture of the widget, we then feed that picture into a convolutional neural network so that we can extract relevant visual information from it. After that, we make use of the encoder component so that the features can be transformed into a tensor. The encoded input, the tensor, and the result of the module that came before the decoder is the three components that are utilized in the construction of a word sequence by the decoder module. In addition to this, we carry out an experiment to determine how effective the process of identifying widgets and creating goals is. In most cases, the IOU for the identification of widgets was 0.81. To generate widget intent, our BLEU-1 model has a score of 0.567, BLEU-2 has a score of 0.356, BLEU-3 has a score of 0.261, and BLEU-4 has a score of 0.131. These data provide evidence that our existing strategy is both effective and efficient [16].

3. DETECTION METHODS

Alongside the deep learning models that were created on images of mobile GUI, existing, conventional computer vision-based methodology was utilized in GUIDE. GUIDE stands for graphical user interface design element toolkit is proposed to detect GUI elements or components on a user interface more efficiently and in less time than any other tool. Traditional computer vision systems do not make use of machine learning to improve their performance. Instead, analysis is done on a pixel-by-pixel basis. Because they call for relatively little in the way of training, they may be easily modified and put into place with minimal effort [17]. We make advantage of two of the most popular GUI detection methods available: Xianyu and REMAUI. The most recent methods for doing this task are the following: Faster-RCNN (two stages), Yolo v3 (one stage), and Center-Net. Made an effort to get retrained and enroll in the GUIDE program (anchor-free). Deep learning, on the other hand, is particularly effective in research involving object identification and may reliably anticipate future outcomes while operating at a high rate. Please keep in mind that for our deep learning models to recognize graphical user interface (GUI) elements, we retrained them using the Rico dataset of mobile app screenshots. This is very important [18].

Jilin Daxue Xuebao (Gongxueban)/Journal of Jilin University (Engineering and Technology Edition)
ISSN：1671-5497
E-Publication: Online Open Access
Vol: 42 Issue: 04-2023
DOI 10.17605/OSF.IO/2XBPZ

a) **REMAUI:** It is a piece of software that can "reverse engineer" user interfaces for various applications. The methods for image processing that are already incorporated into Open=CV are used to process photographs of mobile user interfaces. It uses a bottom-up technique to determine which GUI components are not textual. It starts by putting together individual items from the most fundamental visual regions by employing a technique called canny edge detection (such as edges and contours). Tesseract is a straightforward tool that locates text in GUIs by utilizing- optical character recognition (OCR) [19].

b) **Xianyu:** Alibaba has developed one more GUI reverse engineering project, which translates GUI picture files into code. This project focuses on the user interface. It adheres to the fundamental principles of REMAUI. It uses a technique known as "flood fill" to discover regions that are contextually significant while simultaneously reducing noise that is not related to the context. This helps it detect non-textual elements more accurately. After that, the user interface components are made by repeatedly cutting in both directions. Tesseract's ability to search for text in graphical user interfaces is another one of its many applications [20].

c) **Faster-RCNN:** It is a standard "two-stage" approach that consists of two phases: "stage one," in which the things to be located are categorized, and "stage two," in which the items themselves are located. The first step in the procedure is to create an area proposal network, which is responsible for compiling a list of regions that have a high probability of being suitable locations for item placement (RPN). RPN will compare a given box to a set of anchor boxes whose aspect ratios are already known to identify whether or not a particular box includes an object. To determine the bounding box of an object, the anchor boxes must first be inverted. After that, a convolutional neural network (CNN) image classifier is used to the data to sort the items into the appropriate categories [21].

d) **Yolo v3:** In contrast to Faster-RCNN, YOLO is capable of simultaneously doing object categorization and region regression. It does this by first clustering the ground truth in the training dataset, and then automatically calculate the aspect ratio of the anchor boxes. CNN is used to produce a gridding feature map as well as bounding boxes for each grid. After that, YOLO does a regression based on the box's coordinates to locate the object inside the box while simultaneously calculating the scores for the box without the object [22].

e) **Center-Net:** Yolo and Faster-RCNN both make use of anchor boxes to locate targets; however, the efficiency of the former is contingent on the latter's ratio of anchor boxes to targets. In addition, everything that cannot be contained within these cages will be of little service in the battle against them. By utilizing a method that does not count on anchors, Center-Net can get around these constraints that otherwise would be in place. It is a straightforward approach to locating the geometric center of an item, in addition to its upper- and lower-left, and lower-right extremities [23].

Jilin Daxue Xuebao (Gongxueban)/Journal of Jilin University (Engineering and Technology Edition)
ISSN：1671-5497
E-Publication: Online Open Access
Vol: 42 Issue: 04-2023
DOI 10.17605/OSF.IO/2XBPZ

4. HYBRID APPROACH

In this paper, we describe a novel technique for the recognition of GUI objects that takes into account the graphical representation of the GUI. We divide detection into two stages: the first stage involves locating text, while the second stage looks for non-text elements. Traditional computer vision methods are used to delete portions of a picture that do not include text, while deep learning models are utilized to classify and locate text within the image [24]. Because of this combination, it is much simpler to recognize non-text things without interrupting text, and state-of-the-art performance may be reached while identifying GUI elements. This is a significant improvement over the previous method. The steps involved in our methodology are outlined in Figure 1. The Detection and Identification of User Interface Elements Other than text, because it analyses images on a pixel-by-pixel basis, the traditional computer vision method is superior to deep learning models when it comes to determining the location and shape of objects. Deep learning models rely on statistical regression to create approximations of bounding boxes [25].

The traditional computer vision method examines images at the level of individual pixels. The bottom-up method, on the other hand, is frequently utilized by antiquated methods to give things finer characteristics (such as edges or contours). This method has the propensity to separate GUI elements an excessive amount, which might result in visual noise when used on GUIs with elaborate backgrounds. Because of this, we make use of traditional methods of computer vision to provide a top-down, coarse-to-fine approach to recognizing non-textual graphical user interface (GUI) elements [26]. There are three separate steps involved in the technique. In this approach, the method for locating layout blocks and generating their perimeters uses Sklansky's algorithm as well as the flood-filling methodology. Figure 1 illustrates one example of this kind of block map. On this particular map, the different colored areas reflect different possible configurations of the various blocks [27].

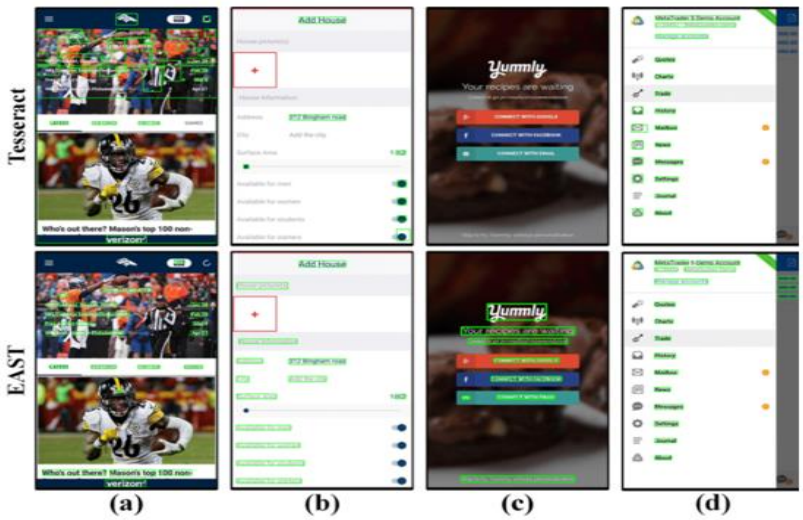

Figure 1: UI Element Detection

Jilin Daxue Xuebao (Gongxueban)/Journal of Jilin University (Engineering and Technology Edition)
ISSN：1671-5497
E-Publication: Online Open Access
Vol: 42 Issue: 04-2023
DOI 10.17605/OSF.IO/2XBPZ

After that, a shape recognition technique is used to locate and count rectangles so that they can be used as GUI layout blocks. Second, the procedure generates a binary map by combining the gradient map obtained from the user interface with a straightforward banalization technique [28]. This combination produces an accurate result. If it doesn't stand out too much from its neighbors, a pixel is considered to be part of the backdrop and is represented by the color black. If there is a significant amount of contrast, the pixel in question will be considered to be in the foreground and will be shown as white. As can be seen in Figure 1, the newly discovered blocks are put to use in the process of breaking the binary map up into more manageable portions [14].

We make use of the approach known as linked component labeling to locate GUI elements within each block binary map. After that, we use Sklansky's method to identify the limits of each component by applying them to the data. To classify the data, we used a ResNet 50 classifier and fed it 90,000 examples of GUI elements drawn from 15 distinct categories during the training phase. A graphical user interface is used to search for elements that are contained inside the text [29]. Because the text that appears in a GUI is regarded to be scene text, we apply a cutting-edge deep learning scene test detector called EAST to locate the text that appears in a GUI image. The first thing that happens is that an input image is given to a feature pyramid network to process. Based on the final feature map, six values are computed for each point. These values include the objectless score, as well as the top, left, bottom, and right offsets and the rotation angle [30].

5. IMPLEMENTATION

The GUIDE toolkit is a web-based program that makes it easier to locate and update graphical user interface (GUI) components in photographs. The findings of the detection can be exported to other programs so that they can be used in subsequent applications, such as GUI testing or GUI automation. GUIDE employs a variety of computer vision methods, including deep learning as well as more standard approaches. They are provided with an intuitive dashboard that enables them to monitor and adjust the detection results as required [31]. Tensor-flow and Pytorch are utilized for the development of specialized deep learning models, whereas Open-CV serves as the basis for both Xianyu and our technique. The landing page and the dashboard are the two parts of the GUIDE that are considered to be the most important. Homepage for the Release of Figure 2(a) presents a portion of our website's homepage that may be accessed by clicking on the link. This portion of the homepage contains brief instructions on how to use the website as well as links to relevant resources. End users can edit user-generated material that is displayed in the form of graphical user interface graphics [32].

Users have the option of using one of the example GUIs that we give or sending us their customized interfaces to obtain a better understanding of how detection is performed and how the dashboard is utilized in its most basic form. In addition, we present the user with a set of sliders that may be modified to modify a few important elements that influence the ultimate result of the detection. These factors are what we refer to as "key

Jilin Daxue Xuebao (Gongxueban)/Journal of Jilin University (Engineering and Technology Edition)
ISSN：1671-5497
E-Publication: Online Open Access
Vol: 42 Issue: 04-2023
DOI 10.17605/OSF.IO/2XBPZ

factors." We use an approach called "Google's Protocol Buffers serializing structured data" to encode the image so that it can be sent across the server. This allows us to send the image without any problems. The detection result is used by Dashboard GUIDE to partition the input GUI into repositionable GUI components, as seen in Figure 2(b). We included functions like drag-and-drop on the dashboard so that it would be easier for users to communicate with one another. The user can change the detection result by simply relocating the graphical user interface (GUI) widget [33].

Indicators of Management's Highest Priorities When the user selects a particular object and clicks on it, the features of that item will immediately become apparent (e.g., type, width, height, left, top). You can quickly modify the size of a piece, move it, as well as delete and undo any prior adjustments. UI Element Toolkit: Utilizing a UI kit to construct a user interface is both the most time-effective and cost-effective approach to doing so.

The user's discoveries are saved in the dataset for future use and are credited to the individual who made them. If the user wants to enhance the functionality of the input GUI, more GUI components are accessible in the UI kits that come with the software. The results obtained from Export-Smuggling Investigations When the picture editing process is finished, the user has the option of exporting the updated position, dimensions, and type of each GUI element. The information will be kept in a JSON file that is both readable by machines and accessible to a large number of people.

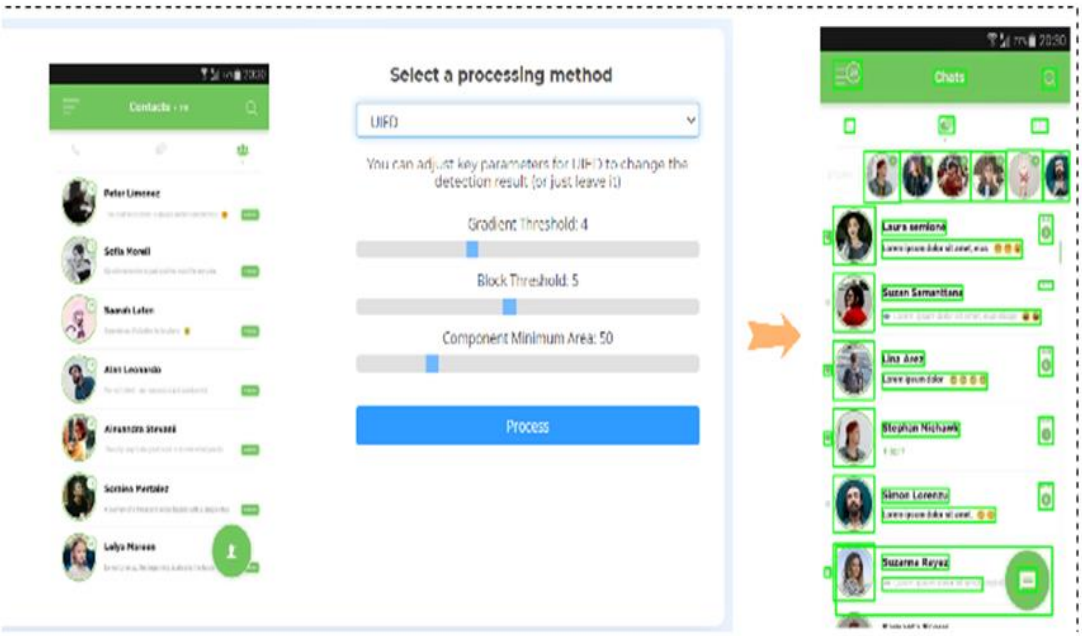

Figure 2: (a) Landing Page

Jilin Daxue Xuebao (Gongxueban)/Journal of Jilin University (Engineering and Technology Edition)
ISSN：1671-5497
E-Publication: Online Open Access
Vol: 42 Issue: 04-2023
DOI 10.17605/OSF.IO/2XBPZ

Figure 2: (b) Dashboard

6. EVALUATION

The goal of our study is to evaluate graphical user interface element detection according to (i) how well it can identify objects and (ii) how well it can be used in subsequent tasks.

a) Usability

We asked ten people who have significant experience in the domains of development and academics and have worked on GUI-related projects for their thoughts on the UIED so that we may have a better understanding of it. They are tasked with utilizing UIED and are afterward polled regarding its value to them and their jobs as well as its potential for further advancement. To "reverse engineer" the graphical user interface (GUI), three of the specialists are currently working on the project (UI2Code). They were all of the same minds when it came to the necessity of accurate GUI element recognition for efficient programming, but they were powerless to do anything about it because they did not have access to commercially available detection technologies. At the moment, there are four persons working in the same area on research involving the use of robots to do automated testing of graphical user interfaces. To avoid needing to build a testing script, they aim to test mobile applications with a robot arm that imitates a human tester. This will allow them to save time. This is not going to function because there is insufficient visual information. According to these researchers, there is not yet a mature and accurate domain-specific solution, even though accurate element information from GUI element identification is essential to teach the robot tester [34-42]. In addition, they believed that a straightforward application such as GUIDE was "very advantageous."

Jilin Daxue Xuebao (Gongxueban)/Journal of Jilin University (Engineering and Technology Edition)
ISSN：1671-5497
E-Publication: Online Open Access
Vol: 42 Issue: 04-2023
DOI 10.17605/OSF.IO/2XBPZ

b) Effectiveness

We run tests on a dataset of 5000 Android mobile GUIs that we received from Rico as part of our investigation into the efficacy of various element recognition algorithms. Previous work that we've done on this topic digs into it more thoroughly. The F1 score served as our key indicator of performance; it is a metric that is comprised of a weighted average of accuracy and recall. The accuracy of the measurements can be determined by determining whether or not the area under the curve (AoC) between the detected bounding box and the ground-truth box is greater than 0.90. To demonstrate the value of each strategy, we determine how long it will take to put it into action. Table 1 contains a summary of all the information gathered through the various methods. When it comes to detecting objects with deep learning models, the performance of Faster RCNN F1=0.271, YOLOv3 F1=0.249, and Center-Net F1=0.282 is superior to that of REMAUI F1=0.183 and Xianyu F1=0.106. The most recent findings (F1=0.524) demonstrate that our methodology is effective. The incompleteness of the dataset is one factor that can help to explain the unsatisfactory result. Because it uses a variety of models and gives users the ability to customize them to their preferences, GUIDE is capable of producing more accurate detection results.

7. CONCLUSION AND FUTURE WORK

This article is all about GUIDE, a toolkit for recognizing GUI elements that work with three popular deep-learning methods and two conventional computer vision methods. We also utilize a novel method that combines state-of-the-art GUI text recognition and non-text GUI element identification using distinct GUI properties. We can recognize text and non-text GUI elements. Our strategy and all user interface customization parameters are in GUIDE. GUIDE's user-tailored interface includes drag-and-drop, size, and class editors to improve detection. GUIDE accesses this interface. After that, the new GUI picture and data can be extracted and reused. GUIDE's detection accuracy and efficacy were assessed. GUIDE is a solid starting point for developing a GUI-based toolkit, according to the research. GUIDE's adaptability makes it fascinating. The UI2CODE project generates code from GUI images. This technique will help graphical user interface (GUI) designers enter their GUI designs and return the working code. In the future, GUIDE will be used for the detecting phase of automatic GUI testing.

References

[1] Y. Wang, M. V. Mäntylä, Z. Liu, J. Markkula, and P. Raulamo-jurvanen, ‘Improving test automation maturity: A multivocal literature review’, Softw. Test. Verification Reliab., vol. 32, no. 3, p. e1804, 2022.

[2] J. C. Paiva, J. P. Leal, and Á. Figueira, ‘Automated assessment in computer science education: A state-of-the-art review’, ACM Trans. Comput. Educ. TOCE, vol. 22, no. 3, pp. 1–40, 2022.

[3] D. Ramesh and S. K. Sanampudi, ‘an automated essay scoring systems: a systematic literature review’, Artif. Intell. Rev., vol. 55, no. 3, pp. 2495–2527, 2022.

ISSN：1671-5497
E-Publication: Online Open Access
Vol: 42 Issue: 04-2023
DOI 10.17605/OSF.IO/2XBPZ

[4] L. Westhofen et al., 'Criticality metrics for automated driving: A review and suitability analysis of the state of the art', Arch. Comput. Methods Eng., vol. 30, no. 1, pp. 1–35, 2023.

[5] I. S. Elgrably and S. R. Bezerra Oliveira, 'A Quasi-Experimental Evaluation of Teaching Software Testing in Software Quality Assurance Subject during a Post-Graduate Computer Science Course.' Int. J. Emerg. Technol. Learn., vol. 17, no. 5, 2022.

[6] S. Semerikov, A. Striuk, L. Striuk, M. Striuk, and H. Shalatska, 'Sustainability in Software Engineering Education: a case of general professional competencies', presented at the E3S Web of Conferences, 2020, vol. 166, p. 10036.

[7] A. Zonnenshain and R. S. Kenett, 'Quality 4.0—the challenging future of quality engineering', Qual. Eng., vol. 32, no. 4, pp. 614–626, 2020.

[8] D. Zimmermann, 'Automated GUI-based Software-Testing Using Deep Neuroevolution', presented at the 2022 IEEE Conference on Software Testing, Verification and Validation (ICST), 2022, pp. 477–479.

[9] E. Alégroth, K. Karl, H. Rosshagen, T. Helmfridsson, and N. Olsson, 'Practitioners' best practices to Adopt, Use or Abandon Model-based Testing with Graphical models for Software-intensive Systems', Empir. Softw. Eng., vol. 27, no. 5, pp. 1–42, 2022.

[10] S. Feng and C. Chen, 'GIFdroid: automated replay of visual bug reports for Android apps', presented at the Proceedings of the 44th International Conference on Software Engineering, 2022, pp. 1045–1057.

[11] Z. Khaliq, S. U. Farooq, and D. A. Khan, 'A Deep Learning-based automated framework for functional User Interface testing', Inf. Softw. Technol., p. 106969, 2022.

[12] L. Wischhof, J. A. Krahl, and R. Müllner, 'Stimuli Generation for Quality-of-Experience Evaluation of Mobile Applications.' Int. J. Interact. Mob. Technol., vol. 16, no. 6, 2022.

[13] K. Jiang, Z. Jiang, Y. Xie, D. Pan, and W. Gui, 'Prediction of Multiple Molten Iron Quality Indices in the Blast Furnace Ironmaking Process Based on Attention-Wise Deep Transfer Network', IEEE Trans. Instrum. Meas., vol. 71, pp. 1–14, 2022.

[14] E. Collins, A. Neto, A. Vincenzi, and J. Maldonado, 'Deep Reinforcement Learning based Android Application GUI Testing', presented at the Brazilian Symposium on Software Engineering, 2021, pp. 186–194.

[15] C. Zhang, T. Shi, J. Ai, and W. Tian, 'Construction of GUI Elements Recognition Model for AI Testing based on Deep Learning', presented at the 2021 8th International Conference on Dependable Systems and Their Applications (DSA), 2021, pp. 508–515.

[16] P. Zhu, Y. Li, T. Li, W. Yang, and Y. Xu, 'GUI Widget Detection and Intent Generation via Image Understanding', IEEE Access, vol. 9, pp. 160697–160707, 2021.

[17] Z. Liu, C. Chen, J. Wang, Y. Huang, J. Hu, and Q. Wang, 'Guided bug crush: Assist manual gui testing of android apps via hint moves', in CHI Conference on Human Factors in Computing Systems, 2022, pp. 1–14.

[18] T. Fulcini and L. Ardito, 'Gamified Exploratory GUI Testing of Web Applications: a Preliminary Evaluation', presented at the 2022 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW), 2022, pp. 215–222.

[19] M. K. Khan and R. Bryce, 'Android GUI Test Generation with SARSA', presented at the 2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC), 2022, pp. 0487–0493.

[20] Z. Khaliq, S. U. Farooq, and D. A. Khan, 'Transformers for GUI Testing: A Plausible Solution to Automated Test Case Generation and Flaky Tests', Computer, vol. 55, no. 3, pp. 64–73, 2022.

ISSN：1671-5497
E-Publication: Online Open Access
Vol: 42 Issue: 04-2023
DOI 10.17605/OSF.IO/2XBPZ

[21] M. Savic, M. Mäntylä, and M. Claes, 'Win GUI Crawler: A tool prototype for desktop GUI image and metadata collection', presented at the 2022 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW), 2022, pp. 223–228.

[22] A. Mulders, O. R. Valdes, F. P. Ricós, P. Aho, B. Marín, and T. E. Vos, 'State Model Inference through the GUI Using Run-Time Test Generation', presented at the International Conference on Research Challenges in Information Science, 2022, pp. 546–563.

[23] Z. Liu, C. Chen, J. Wang, Y. Su, and Q. Wang, 'NaviDroid: A Tool for Guiding Manual Android Testing via Hint Moves', ArXiv Prepr. ArXiv220513992, 2022.

[24] M. Larrea, M. Luj, and D. K. Urribarri, 'A Testing Tool for Information Visualizations based on User Interactions', J. Comput. Sci. Technol., vol. 22, no. 1, pp. e06–e06, 2022.

[25] A. Fergusson and M. Pfannkuch, 'Introducing teachers who use GUI-driven tools for the randomization test to code-driven tools', Math. Think. Learn. vol. 24, no. 4, pp. 336–356, 2022.

[26] T. Su, J. Wang, and Z. Su, 'Benchmarking automated GUI testing for Android against real-world bugs', presented at the Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 2021, pp. 119–130.

[27] L. Ardito et al., 'Feature Matching-based Approaches to Improve the Robustness of Android Visual GUI Testing', ACM Trans. Softw. Eng. Methodol. TOSEM, vol. 31, no. 2, pp. 1–32, 2021.

[28] M. Nass, E. Alégroth, and R. Feldt, 'Why many challenges with GUI test automation (will) remain', Inf. Softw. Technol., vol. 138, p. 106625, 2021.

[29] S. Di Martino, A. R. Fasolino, L. L. L. Starace, and P. Tramontana, 'Comparing the effectiveness of capture and replay against automatic input generation for Android graphical user interface testing', Softw. Test. Verification Reliab., vol. 31, no. 3, p. e1754, 2021.

[30] T. Xu et al., 'Guider: Gui structure and vision co-guided test script repair for android apps', presented at the Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis, 2021, pp. 191–203.

[31] F. Cacciotto, T. Fulcini, R. Coppola, and L. Ardito, 'A metric framework for the gamification of web and mobile gui testing', presented at the 2021 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW), 2021, pp. 126–129.

[32] M. F. Granda, O. Parra, and B. Alba-Sarango, 'Towards a Model-Driven Testing Framework for GUI Test Cases Generation from User Stories.', presented at the ENASE, 2021, pp. 453–460.

[33] S. Saber et al., 'Autonomous GUI Testing using Deep Reinforcement Learning', presented at the 2021 17th International Computer Engineering Conference (ICENCO), 2021, pp. 94–100.

[34] N. Rasool, S. Khan, U. Haseeb, S. Zubair, M. waseem Iqbal, and K. Hamid, "SCRUM AND THE AGILE PROCEDURE'S IMPACT ON SOFTWARE PROJECT MANAGEMENT," Jilin Daxue Xuebao Gongxueban Journal Jilin Univ. Eng. Technol. Ed., vol. 42, pp. 380–392, Feb. 2023, doi: 10.17605/OSF.IO/MQW9P.

[35] K. Hamid, M. waseem Iqbal, H. Muhammad, Z. Fuzail, and Z. Nazir, "ANOVA BASED USABILITY EVALUATION OF KID'S MOBILE APPS EMPOWERED LEARNING PROCESS," Qingdao Daxue Xuebao Gongcheng Jishuban Journal Qingdao Univ. Eng. Technol. Ed., vol. 41, pp. 142–169, Jun. 2022, doi: 10.17605/OSF.IO/7FNZG.

[36] H. Riasat, S. Akram, M. Aqeel, M. waseem Iqbal, K. Hamid, and S. Rafiq, "ENHANCING SOFTWARE QUALITY THROUGH USABILITY EXPERIENCE AND HCI DESIGN PRINCIPLES," vol. 42, pp. 46–75, Feb. 2023, doi: 10.17605/OSF.IO/MFE45.

ISSN：1671-5497
E-Publication: Online Open Access
Vol: 42 Issue: 04-2023
DOI 10.17605/OSF.IO/2XBPZ

[37] D. Hussain, S. Rafiq, U. Haseeb, K. Hamid, M. waseem Iqbal, and M. Aqeel, “HCI EMPOWERED AUTOMOBILES PERFORMANCE BY REDUCING CARBON-MONOXIDE,” vol. 41, pp. 526–539, Dec. 2022, doi: 10.17605/OSF.IO/S5X2D.

[38] K. Hamid, H. Muhammad, M. waseem Iqbal, A. Nazir, shazab, and H. Moneeza, “ML-BASED META MODEL EVALUATION OF MOBILE APPS EMPOWERED USABILITY OF DISABLES,” Tianjin Daxue Xuebao Ziran Kexue Yu Gongcheng Jishu Ban Journal Tianjin Univ. Sci. Technol., vol. 56, pp. 50–68, Jan. 2023.

[39] K. Hamid, H. Muhammad, M. waseem Iqbal, S. Bukhari, A. Nazir, and S. Bhatti, “ML-BASED USABILITY EVALUATION OF EDUCATIONAL MOBILE APPS FOR GROWN-UPS AND ADULTS,” Jilin Daxue Xuebao GongxuebanJournal Jilin Univ. Eng. Technol. Ed., vol. 41, pp. 352–370, Dec. 2022, doi: 10.17605/OSF.IO/YJ2E5.

[40] K. Hamid, M. waseem Iqbal, Z. Nazir, H. Muhammad, and Z. Fuzail, “USABILITY EMPOWERED BY USER’S ADAPTIVE FEATURES IN SMART PHONES: THE RSM APPROACH,” Tianjin Daxue Xuebao Ziran Kexue Yu Gongcheng Jishu BanJournal Tianjin Univ. Sci. Technol., vol. 55, pp. 285–304, Jul. 2022, doi: 10.17605/OSF.IO/6RUZ5.

[41] K. Hamid et al., “Usability Evaluation of Mobile Banking Applications in Digital Business as Emerging Economy,” p. 250, Feb. 2022, doi: 10.22937/IJCSNS.2022.22.2.32.

[42] H. Muhammad et al., Usability Impact of Adaptive Culture in Smart Phones. 2022.