GUI Component Detection Using YOLO and Faster-RCNN

Sena Nur Cavsak[1], Aysu Deliahmetoglu (Intern)[2], Berhan Turku Ay[3] and Senem Tanberk[4]

[1]Research and Innovation, Huawei Turkey Research and Development Center, Turkey
sena.nur.cavsak@huawei.com
[2]Research and Innovation, Huawei Turkey Research and Development Center, Turkey
aysudeliahmetoglu@gmail.com
[3]Research and Innovation, Huawei Turkey Research and Development Center, Turkey
berhan.turku.ay3@huawei.com
[4]Research and Innovation, Huawei Turkey Research and Development Center, Turkey
senem.a.tanberk@gmail.com

Abstract

The graphical user interface (GUI) is crucial for communicating with software users. The detection of GUI elements holds significant importance for various software test automation tasks. In this study, two different object detection models such as YOLOv8 and Faster R-CNN are used to address challenging GUI component detection problems in mobile applications. Two different datasets were used. The first dataset is the RICO data, which consists of 5 classes. The second dataset consists of 600 mobile application screenshots collected from the open-source. This dataset consists of 7 classes and has been labeled and trained on Roboflow. In addition, an automation system was developed to test the object detection models for errors after the version change in the mobile application. With this test automation, the locations of the errors in the mobile application in real-world scenarios were determined and reported. In this experiment, 2 different data were trained in 2 different models, and the data we labeled gave the best result with a mAP value of 0.81 in the YOLOv8 model.

1. Introduction

A GUI component is a fundamental building of a Graphical User Interface, allowing users to interact with software applications in a visual and intuitive manner. These components are crucial to create a user-friendly interface that enables users to input data, interact with the user interface, and get feedback from the system [1]. GUI element components can take various forms, such as buttons, text, header bar, edit text, text view and more. Each component has a specific function and has qualities that may be changed, including size and color. GUI components play an essential role in shaping an application's overall look and feel, enhancing usability, and guiding users to achieve their desired tasks efficiently [2].

GUI component detection refers to identifying and recognizing individual GUI elements within a software application. It involves using computer vision, image processing characteristic, and machine learning techniques to analyze the visual layout and structure of the interface, enabling the system to allocate components and other interactive elements [3]. This detection is very important for some purposes such as test automation because some common issues and challenges arise after a new version is released to mobile apps. These issues can often be caused by different reasons and vary depending on the type of app, platform, and update size. Some of these issues are security vulnerabilities, GUI component changes, performance, and storage issues [4]. These problems can be fixed with test automation. This method provides efficient testing methods, facilitates repeatable action, enables rapid detection of defects, and facilitates the engineer's job.

Object detection with the YOLO (You Look Only Once) model plays an important role in a computer vision task that involves the detection and localization of objects in an image [5]. In this study, Yun et al. [6] utilized the YOLO model, a deep neural network for object detection, to identify GUI elements by combining localization and classification techniques. Once the GUI components were detected, the authors represented them as hierarchical structures and converted them into suitable codes using machine learning algorithms. Bawankule et al. [7] analyzed the YOLOv8 model to classify household waste. This method classifies input images into several categories of household garbage while accurately and quickly predicting the input images. The mAP of the model is 0.97. Altinbas et al. [8] used the YOLOv5 model with the VINS dataset to detect GUI elements in the UI image. The mAP value of the YOLOv5 was found to be 15.69% ahead when the results of this study were compared with the SSD algorithm to train, validate, and test the model.

The Faster R-CNN model has played a significant role in the field of object detection and has become one of the fundamental building blocks in this area [9]. There are studies in the literature on the Faster R-CNN model [10, 11]. Singh et al. [10] used YOLO and fast R-CNN for face mask detection. A dataset of images of people in two categories-those wearing face masks and those not-was used to train both algorithms. They suggest a method for drawing bounding boxes around people's faces based on whether or not they are wearing masks and then compare the performance of both models. In this paper, Xie et al. [11] designed a toolkit called User Interface Element Detection. This toolkit includes computer vision and deep learning models. In addition, the Rico dataset used was trained with 5 models. The authors compared all model results and UIED gave the best result.

There are many studies on detection and test automation in the literature [8,11,18]. However, as in our project, there are not enough studies on the coexistence of these two issues in current studies on the GUI component. In particular, work on the GUI components of the YOLOv8 model is limited. Additionally, the previous works of literature were based on specific data such as Rico. Thus, the proposed system aims to fill the gap in the literature with both YOLOv8 and the data produced.

The contributions of this study to the literature can be listed as follows:

1. A labeled dataset with 7 classes collected from the internet is presented for the detection of GUI components in mobile applications [12].
2. We conduct extensive experiments and analysis to demonstrate the effectiveness of the proposed techniques by training our custom-labeled dataset on both YOLOv8 and Faster R-CNN models.
3. After the update, test automation was done via the mobile application, and the results of the errors were generated and printed as a detection report for diagnosis to be served on a test automation tool.

2. Method

2.1. Dataset

Custom Dataset

Our dataset consists of the mobile GUI component and is collected from the internet. Our dataset [12] consists of 600 images and the labeling was done in Roboflow. The dataset consists of 7 classes: Edit Text, Text View, Button, Image View, Image Button, Header Bar, and Text Button. From the dataset, 420 (70%) images were used for training, 120 (20%) images were used for validation and 60 (10%) images were used for testing.

Rico Dataset

The Rico dataset [13] was used in this study. The Rico dataset includes 72k Android apps spanning 27 categories. This dataset is labeled therefore no preprocessing is required. The Rico dataset consists of 11 classes: Text Button, Icon, Text, Image, Video, Checkbox, Slider, Input, Radio Button, List Item, and On/OFF Switch. In this study, a data cleaning method was used for data preprocessing. In this section, incorrect, corrupt, incorrectly formatted, and incorrectly labeled data in the dataset was deleted. Therefore, we reduced the Rico data to 5 classes (those with the most samples) and additionally removed duplicate data because, according to [14], Rico's annotations are noisy and sometimes even inaccurate [15]. These classes are as follows: Text Button, Icon, Text, Image, and Video.

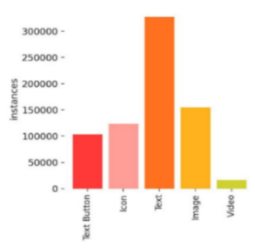


Fig. 1. Labels for Rico Data

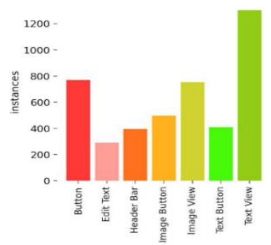


Fig. 2. Labels for Our Data

As can be seen in Figure 1 and Figure 2, there are more text, image, and button parts in both data, so there is more labeling for them.

2.2. Models

2.2.1. YOLOv8 Model

YOLOv8 [16] is a powerful object detection algorithm specifically designed for mobile graphic user interface (GUI) components. It is an enhanced version of the original YOLO (You Only Look Once) model, optimized to deliver high accuracy and fast processing times on mobile devices. YOLOv8 is a deep learning-based object detection model that uses Convolutional Neural Network (CNN) architecture to identify and classify objects in images. YOLOv8 works with a one-pass to detect and classify objects in images, providing fast and efficient object detection.

2.2.2. Faster R-CNN Model

Faster R-CNN [17] is a deep learning model that displays high performance in object detection and classification tasks. The model employs a two-stage approach, first detecting regions of interest (ROIs) and then classifying these ROIs. For the detection of mobile application components, the model takes screenshots or application interfaces as input and identifies GUI components by recognizing common elements such as buttons, text boxes, and images. Studies demonstrate that the Faster R-CNN model achieves high accuracy and precision in the task of mobile application GUI detection. This method holds potential for mobile application developers, automated testing tools, and other applications related to GUI analysis.

2.3. Test Automation

Test automation in mobile applications is a method used to automate the testing process of mobile applications. It automates tests that need to be repeated manually, saving time and resources, reducing the error rate, and improving the quality of the application [18]. In this study, the Rico dataset and the data we labeled are trained in YOLOV8 and Faster R-CNN models. After this step, test automation was done. This part was made in Python and OpenCV toolkits were used. The coordinates obtained in the training section are used here. Thus, errors made in the application after the update can be detected in this way.

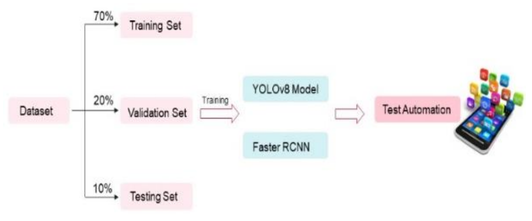


Fig. 3. Proposed Framework

3. Experiments

In this part, we trained our labeled dataset and the Rico dataset in both the Yolov8 model and the Faster R-CNN model. After the detection part, we did the test automation after the mobile application update. Finally, we compared each model and data by mAP value as done in the literature.

3.1. Performance YOLOv8 Model

Mean Average Precision or mAP, is a metric used to evaluate the performance of object detection models in computer vision. It takes into account both precision and recall for multiple object classes in an image. By calculating the average precision for each class and then taking the mean of these values, mAP provides a comprehensive evaluation of the model's ability to detect and localize objects accurately at various confidence thresholds [19]. The mAP score is calculated as in Equation 1 and the mAP scores of various models are used to compare their performance.

$$mAP = \frac{1}{N}\sum_{i=1}^{N} = APi \tag{1}$$

A detection model's performance at different levels of confidence can be usefully assessed using precision and recall. Finding the optimal level of confidence that balances the precision and recall values for a given model is particularly assisted by the F1 score. Using the following equation, the F1 score, precision, and recall may be assessed:

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \tag{2}$$

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \tag{3}$$

$$F\text{-}1\ Score = 2\ .\ \frac{Precision\ .\ Recall}{Precision + Recall} \tag{4}$$

The intersection over union (IoU) is the ratio that is determined by dividing the number of pixels in the union by the number of pixels in the intersection between a predicted object and a ground-truth object. In this study, we calculate the IoU between the predicted boxes and actual boxes in the labeled test dataset and we use an IoU threshold of 0.5 [20].

3.1.1. YOLOv8 Model With Labelled Data

During the training stage of the improved YOLOv8 GUI detection model, we set the batch size to 16, Epoch to 200, initial learning rate to 0.0009, optimizer to Adam, and the framework is Pytorch. Even though we set the epoch to 200, the 120th epoch also worked best with an early stop. YOLOv8l was preferred in YOLO models. Additionally, pre-trained was used.

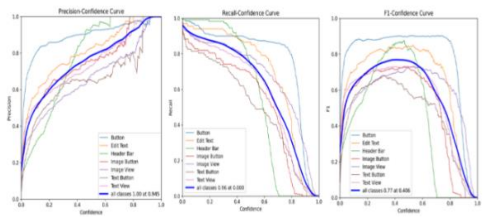


Fig.4. Precision, Recall and F-1 Confidence Curve for Our Data

According to the results of the YOLOv8 model shown in Fig. 4, as the confidence threshold increases, precision increases and recall decreases. The confidence level that optimizes precision and recall based on the F-1 curve is 0.406. Given that the F1 value for this model seems to be around 0.70 and is not too distant from the maximum value of 0.77, choosing a confidence level of 0.5 is the best option.

The precision, recall, and mAP scores obtained by the YOLOv8 model when used on the test dataset are shown in Fig.5.

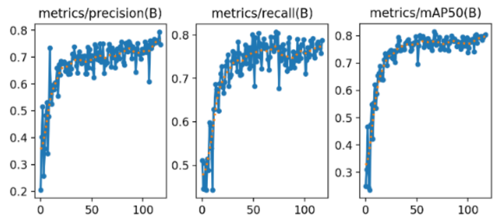


Fig. 5. mAP, Precision, Recall values of the YOLOv8 model for Labelled Data

A high mAP score shows that the model successfully strikes a balance between recall and precision. The mAP scores reached 0.81 when precision is 0.79 and recall is 0.75.

After the training part, the model was tested in applications using boxes, labels, scores and predictions were made in Fig.6.

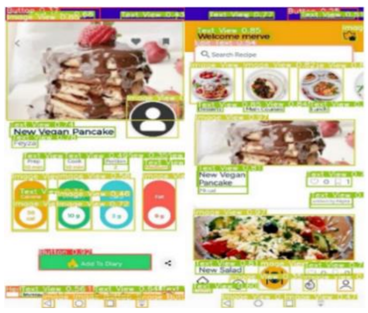


Fig. 6. Mobile Application Example

3.1.2. YOLOv8 Model With Rico Data

In GUI detection model with Rico dataset, we set the batch size to 16, initial learning rate to 0.001, optimizer to Adam and the total number of epochs is 47. Here, each epoch took a very long time and training was done with checkpoints to prevent interruptions.

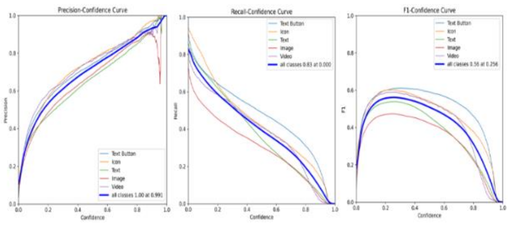


Fig. 7. Precision, Recall and F-1 Confidence Curve for Rico Data

The maximum value in the precision-confidence and recall-confidence graph is 1.00 and 0.83, respectively, as seen in Fig.7. Given that this model's F1 value appears to be about 0.60 and is close to the maximum value of 0.56.

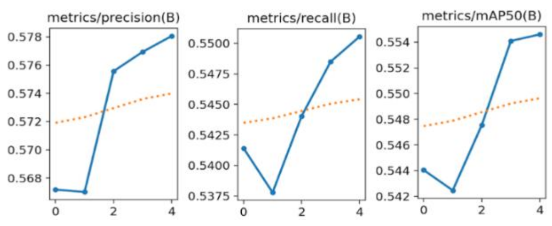


Fig. 8. mAP, Precision, Recall values of the YOLOv8 model for Rico Data

Fig. 8 displays the YOLOv8 model's precision, recall, and mAP50 scores after being applied to the Rico dataset. The mAP scores reached 0.56 when precision is 0.58 and recall is 0.55.

3.2. Performance Faster R-CNN Model

3.2.1. Faster R-CNN Model With Labelled Data

The effectiveness of the proposed Faster R-CNN model for the detection of GUI components was assessed using a number of measures, including mean average precision. Our labeled datasets were used to train and test the model, and Fig. 9 displays the results.

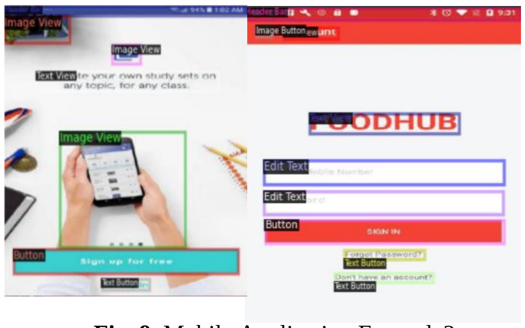


Fig. 9. Mobile Application Example2

After the training part, the score value was plotted. The mAP50 value is 0.77 and the accuracy value is 0.92.

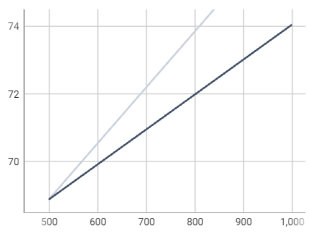


Fig. 10. MaP50 Graph

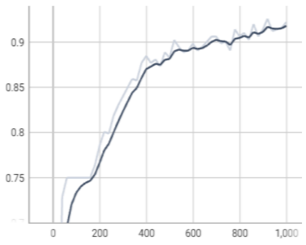


Fig. 11. Accuracy Graph

3.2.2. Faster R-CNN Model With Rico Data

In this section, data was edited before training. The format of the data has been changed. While duplicated data is deleted in the YOLO model, it is not deleted in Faster R-CNN. That's why this part is done manually. Our labeled data part of the Faster RCNN model was not done in this step because the data adjustment part was done automatically since the data was taken from Roboflow.

During the training stage of the improved Faster R-CNN detection model, we set the batch size to 4, initial learning rate to 0.0003, optimizer to Adam, threshold to 0.5, and IoU threshold to 0.8. After the training part, the score value was plotted. The mAP50 value is 0.47. We fine-tuned the Faster R-CNN ResNet50 FPN model in this study.

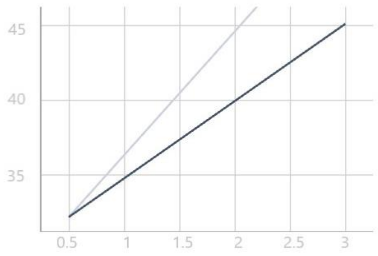


Fig. 12. mAP50 Graph

After all training was done, mAP score was compared. In both models, the YOLOv8 model gave the best results.

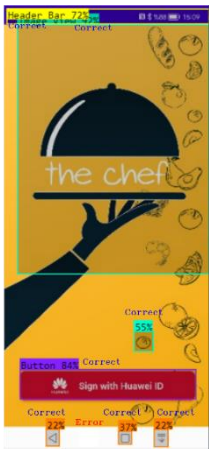
Table 1. Compare mAP50 Result (IoU > 0.5)

Dataset	YOLOv8	Faster R-CNN
Rico Dataset	0.56	0.47
Our Labeled Dataset	0.81	0.77

Overall, our labelled data contains 7 classes and we used the YOLOv8 model. Bunian et al. [21] used 12 classes in the VINS dataset but they used the YOLOv5 model. Accordingly, It is observed that our proposed YOLOv8 model has outperformed the Bunian et al. [21] mAP of the model by 4.61%.

3.3. Test Automation

In this study, YOLOV8 and Faster R-CNN models are trained using the Rico dataset and the data that we labeled. After the training step, test automation was completed. This part is made in Python and openCV toolkits are used. The 'x', 'y', 'width', and 'height' values in the detection section were tested according to these coordinates. After the application update, there is a button deficiency in Figure 13 and a button and image button shift in Figure 14. As seen in the photos, errors were found with test automation and an error was printed on the image with the "cv2.putText" command.



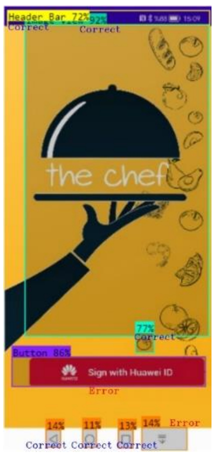
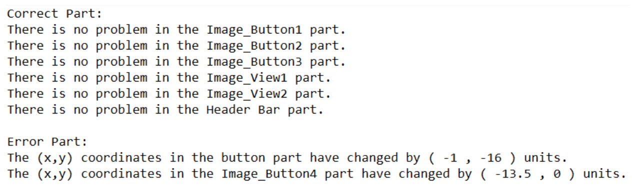


Fig.13. Application Example1 **Fig.14.**Application Example2

After the test, the result was printed with the error and the correct part. In the correct part, it gives the output "There is no problem". In the error part, it gives information about how much the coordinates have changed. It finds this by subtracting the first coordinate from the last coordinate (5).

$$(X_{new}, Y_{new}) = (X_2,Y_2) - (X_1,Y_1) \tag{5}$$



```
Correct Part:
There is no problem in the Image_Button1 part.
There is no problem in the Image_Button2 part.
There is no problem in the Image_Button3 part.
There is no problem in the Image_View1 part.
There is no problem in the Image_View2 part.
There is no problem in the Header Bar part.

Error Part:
The (x,y) coordinates in the button part have changed by ( -1 , -16 ) units.
The (x,y) coordinates in the Image_Button4 part have changed by ( -13.5 , 0 ) units.
```

Fig. 11. Application report with Error 2 Result

4. Conclusion

In this article, the GUI component problem encountered after the update in mobile applications was investigated. First, the problem started with GUI component detection. In this part, 2 models were used for detection, these are YOLOv8 and Faster R-CNN. Detection models were trained with 2 data separately. One of these is Rico data, which is open source and consists of 5 classes. The other data is an open source set consisting of 7 classes and we labeled it from Roboflow. After the detection part, the test automation part was made in Python with OpenCV toolkits. With test automation, the problems in the application are detected and the outputs of the problems are printed. In this study, as a result of model comparison, YOLOv8 gives the best result in mAP50 value with 0.81. In the future, the GUI detection problem can be reconsidered from different perspectives by examining new model structures. Additionally, the data we label can be improved and integrated into different models. Finally, we will investigate integrating our GUI component detection framework into a visual test automation tool for mobile applications to enhance testing efficiency.

5. References

[1] Xue, Feng, Junsheng Wu, and Tao Zhang. "Visual Identification of Mobile App GUI Elements for Automated Robotic Testing." Computational Intelligence and Neuroscience 2022 (2022).

[2] Chen, Jieshan, et al. "Object detection for graphical user interface: Old fashioned or deep learning or a combination?." proceedings of the 28th ACM joint meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. 2020.

[3] Jaganeshwari, K., and S. Djodilatchoumy. "an Automated Testing Tool Based on Graphical User Interface With Exploratory Behavioural Analysis." J. Theor. Appl. Inf. Technol. 100.22 (2022): 6657-6666.

[4] Coppola, Riccardo, et al. "Scripted GUI testing of Android open-source apps: evolution of test code and fragility causes." Empirical Software Engineering 24 (2019): 3205-3248.

[5] Diwan, Tausif, G. Anirudh, and Jitendra V. Tembhurne. "Object detection using YOLO: Challenges, architectural successors, datasets and applications." multimedia Tools and Applications 82.6 (2023): 9243-9275.

[6] Yun, Young-Sun, et al. "Detection of gui elements on sketch images using object detector based on deep neural networks." Proceedings of the Sixth International Conference on Green and Human Information Technology: ICGHIT 2018. Springer Singapore, 2019.

[7] Bawankule, Ram, et al. "Visual Detection of Waste using YOLOv8." 2023 International Conference on Sustainable Computing and Smart Systems (ICSCSS). IEEE, 2023.

[8] Altinbas, Mehmet Dogan, and Tacha Serif. "GUI Element Detection from Mobile UI Images Using YOLOv5." International Conference on Mobile Web and Intelligent Information Systems. Cham: Springer International Publishing, 2022.

[9] Chaudhuri, Arindam. "Hierarchical modified Fast R-CNN for object detection." Informatica 45.7 (2021).

[10] Singh, Sunil, et al. "Face mask detection using YOLOv3 and faster R-CNN models: COVID-19 environment." Multimedia Tools and Applications 80 (2021): 19753-19768.

[11] Xie, Mulong, et al. "UIED: a hybrid tool for GUI element detection." Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. 2020.

[12] https://app.roboflow.com/huawei-tz75a/gui-detection-uz7l4/2

[13] Deka, Biplab, et al. "Rico: A mobile app dataset for building data-driven design applications." Proceedings of the 30th annual ACM symposium on user interface software and technology. 2017.

[14] Bunian, Sara, et al. "Vins: Visual search for mobile user interface design." Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems. 2021.

[15] Gu, Zhangxuan, et al. "Mobile User Interface Element Detection Via Adaptively Prompt Tuning." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2023.

[16] G. Jocher, A. Chaurasia, and J. Qiu, “YOLO by Ultralytics.” https://github.com/ultralytics/ ultralytics, 2023. Accessed: February 30, 2023.

[17] Ju, Rui-Yang, and Weiming Cai. "Fracture Detection in Pediatric Wrist Trauma X-ray Images Using YOLOv8 Algorithm." arXiv preprint arXiv:2304.05071 (2023).

[18] Nam, Seong-Guk, and Yeong-Seok Seo. "GUI Component Detection-Based Automated Software Crash Diagnosis." Electronics 12.11 (2023): 2382.

[19] Bawankule, Ram, et al. "Visual Detection of Waste using YOLOv8." 2023 International Conference on Sustainable Computing and Smart Systems (ICSCSS). IEEE, 2023.

[20] White, Thomas D., Gordon Fraser, and Guy J. Brown. "Improving random GUI testing with image-based widget detection." Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis. 2019.

[21] Bunian, S., et al.: VINS: visual search for mobile user interface design. In: Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems, pp. 1–14. ACM (2021)