Hindawi
Mobile Information Systems
Volume 2022, Article ID 8900734, 9 pages
https://doi.org/10.1155/2022/8900734

Research Article

YOLOv5-MGC: GUI Element Identification for Mobile Applications Based on Improved YOLOv5

Jing Cheng,[1] Dingmei Tan,[1] Tao Zhang,[2] Aodi Wei,[2] and Jingyi Chen[2]

[1] *School of Computer Science and Engineering, Xi'an Technological University, Xi'an 710021, China*
[2] *School of Software, Northwestern Polytechnical University, Xi'an 710072, China*

Correspondence should be addressed to Tao Zhang; tao_zhang@nwpu.edu.cn

Received 12 April 2022; Accepted 4 July 2022; Published 8 August 2022

Academic Editor: Ali Khattak Hasan

The identification of interface elements is the first step in mobile application automated testing and the key to smooth testing. However, existing object detection algorithms have a low accuracy rate, and some tiny elements are missed in the recognition of graphical user interface (GUI) elements. To address this limitation, this paper proposes the YOLOv5-MGC algorithm, a robot vision-based interface element recognition algorithm for mobile applications. The algorithm improves the network by using K-means++ algorithm for target anchor box generation, applying the attention mechanism, adding a microscale detection layer, and introducing the Ghost bottleneck module. The proposed approach enhances the recognition accuracy of the elements through the target anchor box and attention mechanism. Moreover, it enhances the network's ability to detect tiny elements, which improves the shortcomings of the current target detection algorithm and is conducive to further promoting mobile application robot testing and enhancing robot testing automation. Experimental results show that the YOLOv5-MGC algorithm is superior to the YOLOv5 for object detection in the recognition of GUI elements, with the mean average precision (mAP_0.5) reaching 89.8% and the recognition precision reaching 80.8%.

1. Introduction

In recent years, the demand for mobile applications has been surging due to their portability and convenience, and medical applications are also on the rise. To ensure the quality of mobile applications and improve system reliability, mobile application testing technology has become a hot topic in the current research [1]. GUI provides the medium for interaction between users and computers, offering great convenience to users by transferring information through graphics such as buttons, text boxes, and windows [2]. Therefore, GUI element identification is a critical issue in mobile application testing. With the rapid development of Internet applications, especially in mobile devices, the design and testing of GUI around various businesses is becoming increasingly complex. Traditional manual testing can no longer meet the needs of GUI testing [3, 4]; hence, automated testing technology is necessary to solve these problems.

Although many automatic testing frameworks and tools are available at this stage [5–7], these testing tools require investment in learning costs, high maintenance costs, and harsh conditions of use, with strong restrictions on the platform, device, and application under test. Robotic testing is a new approach to mobile application automated testing. It overcomes the inefficiency and high cost of traditional manual testing, as well as the complexity and maintenance difficulties of general automated testing. Robot vision is an important part of robotic automated testing. By using robot vision, a GUI framework can be obtained, test scripts are automatically generated, and the robotic arm is controlled to perform the corresponding test actions, thereby laying the foundation for subsequent robot testing. In comparison with the testing method of acquiring a GUI framework for mobile applications through scripts [8], the whole testing process does not require manual writing of test scripts, and human involvement is greatly reduced, which can effectively improve the automation degree of robotic testing.

At present, in the process of using robot vision for GUI element identification for mobile application, the common identification techniques can be divided into two categories: traditional computer vision-based methods and deep learning-based methods. Traditional computer vision methods [9–11] locate interface elements through image matching or Canny algorithm, which are simple and fast, but difficult to extract complex combined elements accurately and have a low detection rate. Deep learning methods [12–14] mainly identify GUI elements by training the object detection algorithm to improve the detection accuracy. As a representative one-stage object detection algorithm, YOLOv5 has been widely adopted due to its speed and accuracy [15]. However, when the current YOLOv5 algorithm identifies GUI elements with complex backgrounds, it can lead to poor accuracy of GUI element recognition, false detection of some tiny elements, missed detection, and repeated detection [16].

To solve the above problems, this paper proposes a detection algorithm, YOLOv5-MGC, based on the improved YOLOv5 to recognize GUI elements from the perspective of robot vision. First, the K-means++ algorithm is used to generate target anchor frames instead of the K-means algorithm, making the recognition algorithm more applicable to the mobile application interface dataset. Second, the Convolutional Block Attention Module (CBAM) is applied to the YOLOv5 backbone network to improve the accuracy of the recognition algorithm. In the detection stage, to enhance the recognition success rate of tiny elements, a microscale detection layer is added to the network structure. Finally, the Ghost bottleneck (G-Bneck) module is introduced to build a lightweight network to reduce the degradation of the algorithm's detection speed because of the increase in network structure. The experimental results show that our algorithm is feasible. The main contributions of our work are as follows:

(a) We propose a YOLOv5-MGC algorithm for GUI element identification, which improves the accuracy of element recognition and reduces the rate of missing tiny elements in GUI.

(b) We apply robot vision to object detection in an attempt to combine object detection technology with automated robotic testing. The proposed approach serves the automated testing of mobile applications and improves the automation degree of mobile application tests.

2. Related Works

Our work in this paper and the proposed approach therein are related to object detection-based recognition of interface elements. In recent years, the object detection algorithm for an optimal trade-off between precision and speed has been a popular research topic. The two-stage and one-stage detectors have made great contributions to the improvement of efficient network and better methodology [17].

Zhang [12] proposed a new mobile application element recognition algorithm based on computer vision and object detection techniques. He used mainstream object detection algorithms, Faster R–CNN and RetinaNet, for training and validation. The experimental results show that the object detection algorithm can be applied to element recognition with good robustness and accuracy. Gallery [18] used Faster R–CNN to determine the type, size, and location of GUI elements by automatically collecting a library of 11 types of element using screenshots of the application interface. Faster R–CNN is a target detection technology based on a two-stage anchor box. In this method, a generator region proposal network (RPN) was used to generate the proposals, and the anchor was introduced to cope with the different sizes of objects, such that detection accuracy and speed were significantly improved. In addition, improved Faster R–CNN networks, Cascade R–CNN [19], and Mask R–CNN [20] were proposed to improve the accuracy of tiny target detection. These two-stage object detection algorithms have high accuracy, but high time complexity. Thus, applying them to real-time detection systems is difficult.

To improve object detection accuracy, Redmon et al. [21] developed the real-time detector YOLO in 2016, laying the foundation of one-stage object detection. The one-stage object detection model generates high-quality target proposal regions through a dedicated RPN. In comparison with the two-stage object detection, it provides faster detection. At the same time, the one-stage object detection algorithm has obvious advantages for dense and overlapping GUI elements.

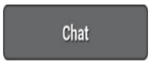
White et al. [14] used YOLOv2 to achieve fast identification and localization of elements when performing GUI automated testing tasks, which improved the detection accuracy of GUI elements. Subsequently, Redmon and Farhadi [22] proposed the YOLOv3 network to improve the weak generalization ability and low detection accuracy. In 2020, Wu et al. [23] proposed the YOLOv5 model based on YOLOv4, which has become one of the best algorithms in terms of detection accuracy and speed performance, and has been widely used in various industrial scenarios [24, 25]. However when faced with GUI elements, YOLOv5 recognition algorithm still has some problems. Given the inter-class similarity between GUI elements, element recognition errors, and low recognition accuracy will exist. Conversely, as GUI elements are generally small and dense, YOLOv5 may miss detection when recognizing these tiny elements. However, the recognition of GUI elements is the first step of mobile application testing, and it is the key to smooth testing. The shortcomings of existing algorithms will affect the quality of later mobile application testing. Therefore, we need to improve the existing network to raise the recognition accuracy of GUI elements.

Generally, the recognition of GUI elements plays a pivotal role in automated robotic test modeling. The result of the recognition directly affects the generation of test scripts and the subsequent test execution of the robotic arm. Hence, this paper investigates how to use robot vision to identify mobile application interface elements accurately and efficiently in the context of the above research.

3. Revisiting the Problem of GUI Element Recognition

3.1. Classification of Interface Elements. GUI element classification is a prerequisite for element identification. Currently, GUI interface design has changed from the text-based

Table 1: Categories and features of GUI elements.

Categories	Features	Example
Text	Some descriptive text present in the interface	
Image	Rich and colourful, mostly used for people and landscape	
Icon	Single colour, simple, and clear patterns	
Button	With distinct bumps and a sharp border	
Radio button	Round icons with a clear blank circular hole in the center	
Check box	Blank rectangles or blank rectangles with checkmarks	
Switch	Rounded rectangles or rectangles with right angles, with shades of colour at the front and back	
Others	Other types of interface elements	

interface to the interface consisting of windows, icons, menus, and toolbars [26]. These basic elements can constitute different styles of GUI interfaces. To collect and label GUI elements, we classify different GUI elements according to the design semantics of mobile applications [27] as shown in Table 1.

This paper classifies GUI elements into the eight categories mentioned above. The variety of elements is complex and diverse. Moreover, there exist differences within categories and similarities between categories. This complex and unique feature brings challenges to GUI element identification.

3.2. Problems of GUI Element Recognition. In real-time GUI element recognition applications, the system needs to meet several characteristics: acceptable element recognition accuracy and fast response time. YOLOv5 has been widely used in the industry for its fast recognition and high accuracy, but two problems remain with the YOLOv5 algorithm when it comes to mobile application interface elements.

First, the recognition accuracy of GUI elements is not high. As the style of GUI elements is not unified and many similarities exist between classes, YOLOv5 may have poor recognition accuracy and incorrect recognition when recognizing GUI elements. Without a sufficient accuracy guarantee, it will lead the subsequent robot automated test to the wrong process and cannot meet the testing requirements.

Second, some tiny elements are missing. As GUI elements are generally small and dense, YOLOv5 may miss the detection of these tiny elements. For mobile application robot automated testing, if an element is missed, it will not be included in the subsequent test scripts. Thus, the interface element will not be clicked on by the robot at all.

Therefore, in this paper, the YOLOv5-MGC algorithm is designed based on the improved YOLOv5 from the above perspective.

4. Proposed Method

In this part, an improved YOLOv5 detection model based on robot vision, YOLOv5-MGC algorithm, is proposed for GUI element recognition. First, the K-means++ algorithm is used to generate suitable target anchor frames according to the mobile application interface dataset. Second, the attention mechanism is applied to enhance the focus of the network, and a microscale detection layer is added to identify tiny elements that are easily missed. Finally, the G-Bneck module is introduced to improve the recognition of GUI elements without reducing the detection speed and accuracy. The structure of the YOLOv5-MGC network is shown in Figure 1.

4.1. K-Means++ Clustering Algorithm. Faster R–CNN first proposed the concept of the anchor box to detect multiple objects in a grid unit [28]. The YOLO algorithm takes this idea and uses anchor boxes to match objects better. It uses the K-means clustering algorithm to obtain the prior anchor box to predict each grid. Although the K-means algorithm is simple and fast and the obtained anchor box value matches better than the manual setting, there still exist some drawbacks [29–31]. First, the number of K-means clustering centers K needs to be given in advance; however, in practice, the selection of K values is extremely difficult. Second, the K-means algorithm needs to determine the initial clustering centers randomly. Different initial clustering centers may lead to completely different clustering results.

The K-means++ algorithm can effectively solve the randomness problem of the K-means algorithm. The basic principle of the K-means++ algorithm to choose the initial clustering centers is that the initial clustering centers should be as far away from each other as possible. The idea of selecting cluster centers is as follows: assume that n initial cluster centers have been selected. Then, when selecting the $n + 1$ cluster center point, the more distant the points from

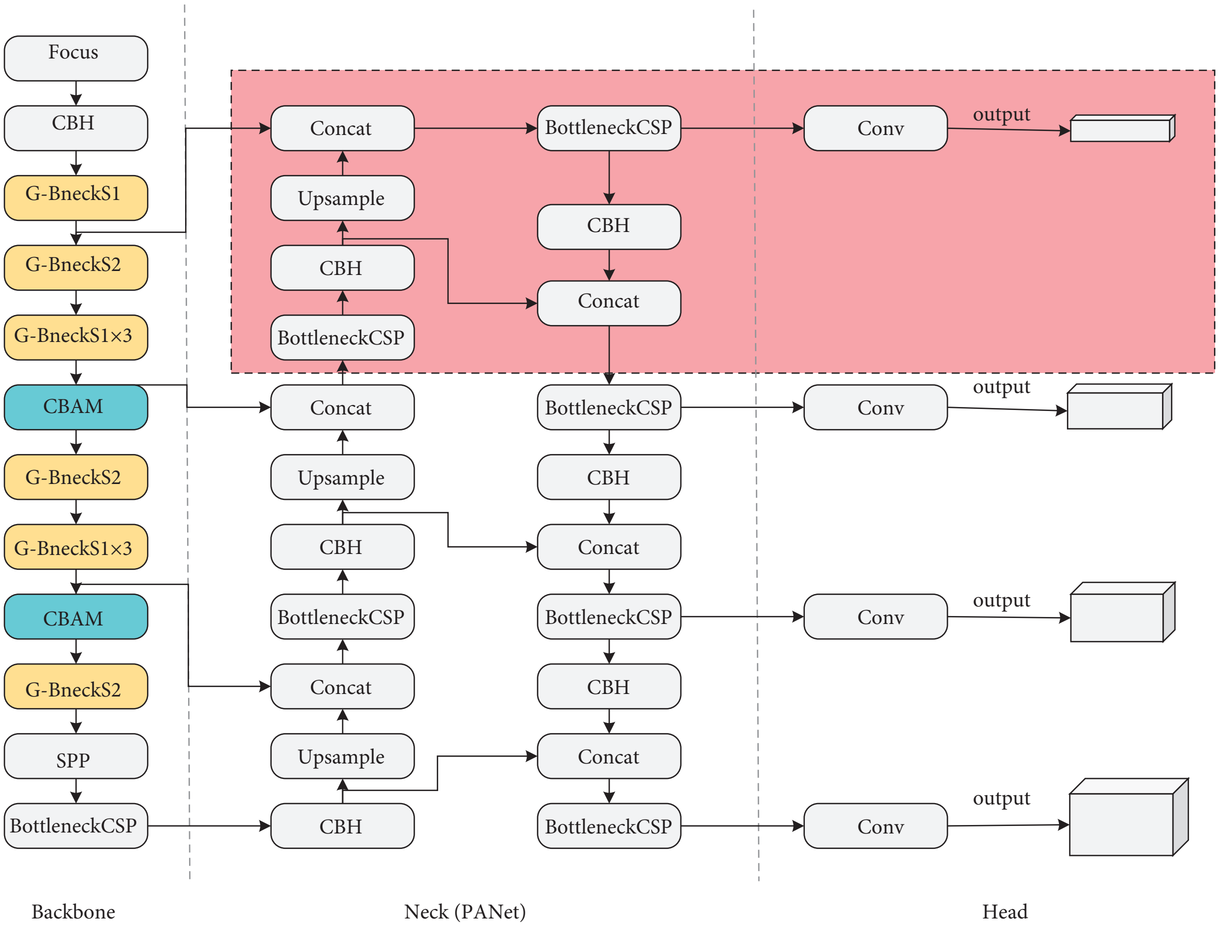


Figure 1: YOLOv5-MGC network structure diagram. It consists of three main parts: backbone, neck, and head. The red part represents the microscale detection layer, the blue part is the attention mechanism, and the yellow part represents the modified Ghost Bottleneck module.

the current n cluster centers are, the higher the probability of being selected will be.

The K-means++ algorithm is broken down as follows:

(a) Select a central point in the dataset randomly.

(b) First, calculate the Euclidean square distance $D(x)$ between each sampling point and the currently existing central point. Then calculate the probability of each sampling point $P(x)$ selected as the next cluster center. Finally, select the next cluster central point according to the roulette wheel method. The probability formula is as follows:

$$P(x) = \frac{D(x)^2}{\sum_{x\in X} D(x)^2}. \quad (1)$$

(c) Repeat step (b) until K clustering centers have been selected.

(d) Calculate the distance from each sampling point in the dataset to each of the K cluster centers and classify it into the class corresponding to the cluster center with the smallest distance.

(e) For each category, recalculate its cluster center.

(f) Repeat Steps (d) and (e) until the position of the clustering center no longer changes.

4.2. Attention Mechanism Network. CBAM is one of the representative works on attention mechanisms published by Woo et al. in 2018 [32]. The main network architecture of CBAM is relatively simple; one is the channel attention module, and the other is the spatial attention module, both of which are independent of each other. Combining the channel attention mechanism and the spatial attention mechanism for feature extraction not only saves parameters and computational power but also ensures that it is plug-and-play and easy to integrate. The CBAM module is shown in Figure 2.

To improve the accuracy of GUI element recognition algorithm and prevent category recognition errors, YOLOv5-MGC adds the CBAM attention mechanisms after the Bottleneck CSP module on Layers 5 and 7 of the YOLOv5 backbone network. By using the attention mechanism to increase the expressiveness of the network, the important features of the interface elements are focused on and unnecessary features are suppressed. These features can improve the recognition accuracy of GUI elements and make the recognition results more accurate, thereby

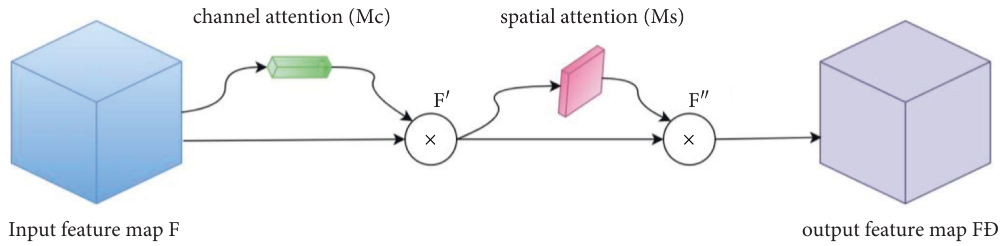


Figure 2: CBAM module.

accurately correlating test behaviors and enabling precise guidance of test execution.

4.3. Microscale Detection Layer. YOLOv5 continues the design idea of YOLOv3 by detecting through three scales, downsampling the dimensionality of the input image by 8, 16, and 32. However, this study found that relying only on the three scales for training when recognizing GUI elements, some elements were still missed. Most of these elements are small icons located at the top left corner of the interface for the backward function or at the bottom.

Therefore, to address the phenomenon of missing small targets in the process of GUI element recognition, this paper designs smaller-scale feature maps for tiny targets. On the basis of the YOLOv5 network structure, a tiny detection layer is added by downsampling the input image by a factor of 64. This microscale layer generates a feature map with a smaller sense field by extracting lower spatial features and fusing them with deeper semantic features, making the detection network structure more extensive and detailed. It also facilitates the prediction of smaller-sized targets and reduces the occurrence of missed detection.

4.4. G-Bneck Module. To improve the accuracy of GUI element recognition and reduce the miss detection rate of elements, an attention module and a microscale detection layer are added to the network, which will inevitably reduce the training and target detection speed of the network. To better serve the fully automated testing of mobile applications and support the real-time operation of deep convolutional models, using lightweight convolutional model architecture is a good solution.

In this paper, we consider adding the G-Bneck module to the backbone network of YOLOv5 to reduce the overall size of the model without increasing the network parameters. This method allows the network to understand the redundant information better and faster, thereby improving the model accuracy. The G-Bneck has two structures [33], namely, the bottleneck with Stride = 1 and Stride = 2, as shown in Figure 3. In particular, the first Ghost module in Stride = 1 is used as an extension layer, increasing the number of channels. The ratio of the number of output channels to the number of input channels is called the extension ratio. The second Ghost module reduces the number of channels to match the shortcut path. The shortcut is then used to connect the inputs and outputs of the two Ghost modules. The second Ghost module in the G-Bneck module does not use ReLU. The other layers apply batch normalization and the ReLU activation function after each layer. The case corresponding to Stride = 2 acts as downsampling in this model and implements fast paths for connectivity [34]. A G-Bneck module is used to replace part of the YOLOv5 network structure to speed up network training while ensuring recognition accuracy.

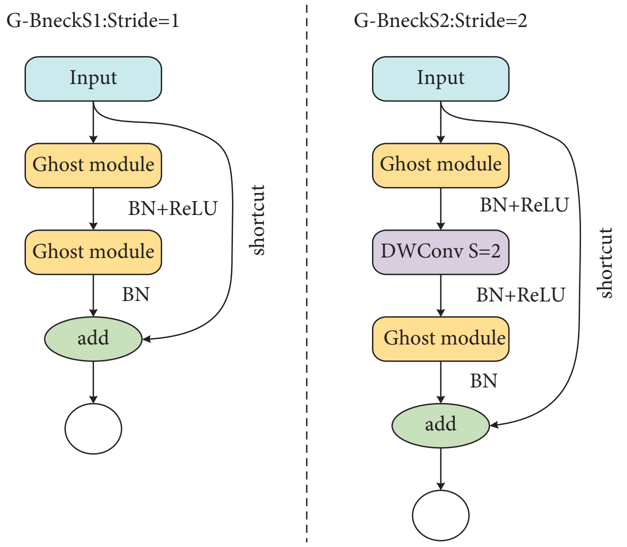


Figure 3: G-Bneck module network structure.

5. Experimental Analysis and Results

To evaluate the effectiveness and accuracy of the YOLOv5-MGC algorithm proposed in this paper on the task of GUI element recognition, this section compares the YOLOv5-MGC algorithm with the YOLOv5 algorithm for experiments. The experimental environment is built using the Python language under the Windows operating system. The relevant algorithm is also programmed based on the Pytorch framework.

5.1. Dataset Preparation. The research is based on the publicly available Rico dataset, which contains over 93,000 mobile application design data in 27 categories. Each application screenshot in the dataset contains a JSON file of the view hierarchy of application interface elements. However, as the JSON files are not annotated uniformly, the Rico dataset must be cleaned and filtered to ensure sufficiently rich and valid training samples for robot vision-based GUI recognition.

Through the analysis and statistics of the GUI element categories, we classify the elements into eight categories: Text, Image, Icon, Button, Radio Button, Check Box, Switch, and Others. A total of 4,000 images are labeled and divided into training, validation, and test sets in the ratio of 8 : 1 : 1.

5.2. Evaluation Indicators. Robotic visual recognition of GUI elements is essentially a domain-specific object detection task. Its common evaluation indicators include precision, recall, $F1$-score, average precision (AP), and frame per second (FPS). The test results of the binary classification model are represented in the data by the following four categories: true positive (TP), false positive (FP), true negative, and false negative (FN).

5.2.1. Precision. Precision rate refers to the percentage of TPs in all samples where the model predicted positive. The calculation formula is as follows:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}. \tag{2}$$

5.2.2. Recall. Recall rate describes the success rate of prediction for positive samples in the algorithm model, which is the likelihood of a positive class being recalled.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{3}$$

5.2.3. F1-Score. The $F1$-score is the harmonic average of the precision and recall rates and is generally inversely related to each other. $F1$-score ranges from 0 to 1, with larger values indicating better classification of the algorithm model.

$$F1 = 2\frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{4}$$

5.2.4. AP. The integral of the P–R curve is the area under the P–R curve. The P–R curve is a curve with recall as the horizontal axis and precision as the vertical axis.

For a continuous P–R curve,

$$\text{AP} = \int_0^1 \text{PR}\mathrm{d}r. \tag{5}$$

For a discrete P–R curve,

$$\text{AP} = \sum_{k=1}^{n} P(k)\Delta r(k). \tag{6}$$

5.2.5. FPS. FPS indicates the number of images that can be processed by the algorithm model per second. It is often used to evaluate the speed of the algorithm model.

6. Results and Discussion

The experiments trained YOLOv5-MGC and YOLOv5 under the above dataset. For the YOLOv5 algorithm, we use the idea of transfer learning and training under the officially provided weight files. Conversely, the YOLOv5-MGC algorithm changes the network structure and needs to be trained from scratch. A total of 10,000 iterations are performed throughout the training process, with a batch size of 16. The learning rate is 0.01, the stochastic gradient descent momentum value is 0.937, the weight decay is 0.0005, and the rest of the settings are kept as default.

Table 2 shows the performance indicators of the mobile application interface elements for both algorithms based on the evaluation indicators defined above. The data show that the YOLOv5-MGC algorithm improves the AP by 0.006% in the image category, 0.01% in the ico.con category, 0.03% in the Radio Button category, and 0.071% in the Others category compared with the YOLOv5 algorithm; whereas the AP in the Button and Switch categories remained the same. Overall, the YOLOv5-MGC algorithm proposed in this paper improves the recognition precision, recall, and F1-score in several categories, effectively demonstrating the accuracy of the YOLOv5-MGC.

In the field of object detection, the more complex the structure of the algorithm model is, the greater the weight of the completed algorithm model after training will be. This condition results in a slower detection speed, which is not conducive to engineering deployment and application. In the YOLOv5-MGC algorithm model, a G-Bneck module is introduced into the backbone network to mitigate the detection speed degradation caused by the expansion of the network structure. The experimental data show that the YOLOv5 algorithm model takes an average of 0.009 s to detect an image, whereas the YOLOv5-MGC algorithm model takes an average of 0.011 s. Table 3 shows a comparison of the FPS at recognition between the two algorithm models.

As shown in Table 3, although the FPS of the YOLOv5-MGC algorithm decreases somewhat compared with that of the YOLOv5 algorithm, the detection speed of YOLOv5-MGC can already meet the requirements of mobile application robot testing for GUI element recognition.

After the above analysis, to see more intuitively the difference in effectiveness between YOLOv5-MGC and YOLOv5 when performing detection, Figure 4 shows the comparison of the two algorithm models in GUI element recognition.

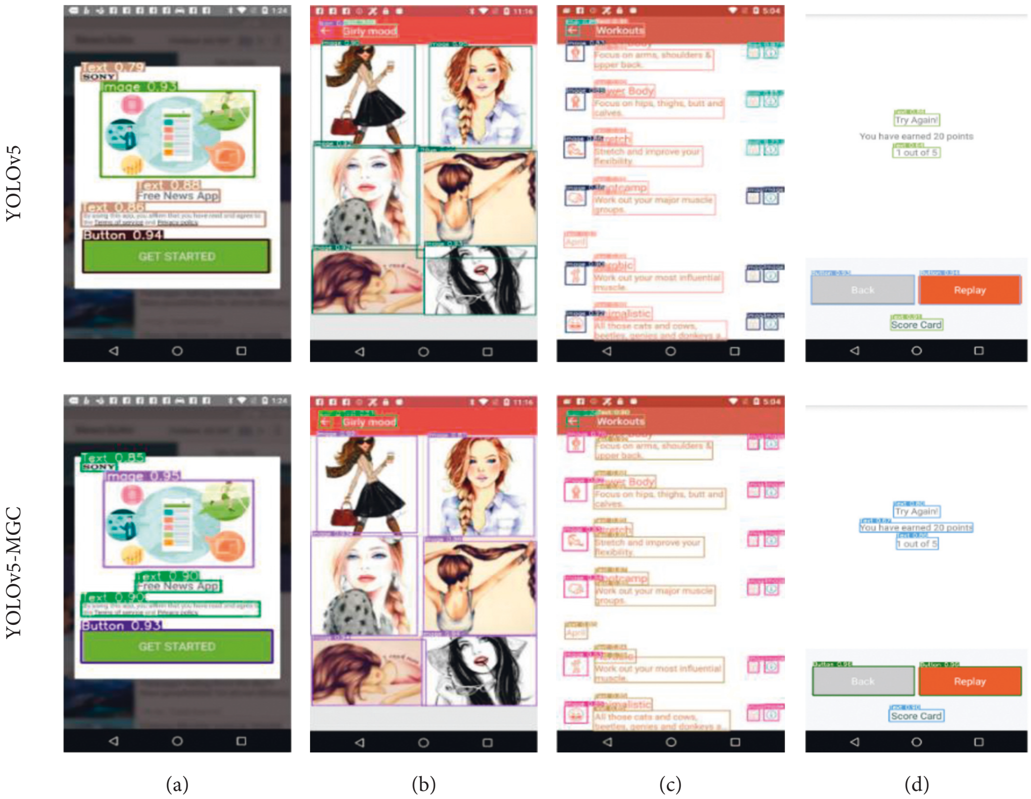
Figure 4(a) shows that the recognition accuracy of using YOLOv5-MGC for the Text, Image, and Button categories is higher than that of using the YOLOv5 algorithm. In Figure 4(b), the YOLOv5 algorithm is less effective in predicting the bounding box for multiple images, and even some of the bounding boxes overlap. Conversely, YOLOv5-MGC is more accurate in predicting the bounding boxes. Figure 4(c) shows multiple small images. The YOLOv5 algorithm incorrectly predicts the element located in the upper-right part of the interface as the Icon category, whereas the YOLOv5-MGC algorithm correctly predicts all

Table 2: GUI elements recognition results.

Category	Algorithm							
	YOLOv5-MGC (%)				YOLOv5 (%)			
	AP	Precision	Recall	*F*1	AP	Precision	Recall	*F*1
Text	91.8	86.5	89.4	87.93	92.9	84.5	89.4	86.88
Image	91.5	70.6	78.6	74.39	90.9	66.1	81.1	72.84
Icon	77.6	72.7	91.5	81.02	76.6	71.1	93	80.59
Button	94.5	88	94.5	91.13	94.5	84.7	97.2	90.52
Radio button	98.2	83.3	99.3	90.6	95.2	78.1	97.3	86.65
Check box	98.2	95.7	98.5	97.08	99.5	94.1	97.5	95.77
Switch	99.5	80.8	99.2	89.06	99.5	70.5	99.1	82.39
Others	88.3	69	60.9	64.7	81.2	48.6	63.6	55.1

Table 3: FPS comparison.

Algorithm	YOLOv5	YOLOv5-MGC
FPS	111	91



(a) (b) (c) (d)

Figure 4: Comparison of YOLOv5 and YOLOv5-MGC detection results.

of them as the Image category. In Figure 4(d), a line of text located in the middle of the interface is missed using the YOLOv5 algorithm.

The above experimental data and detection results show that the YOLOv5-MGC algorithm proposed in this paper can effectively improve the recognition accuracy of GUI elements and reduce the missed recognition rate of tiny elements. It has good performance in GUI element recognition, which provides some help for the subsequent research on mobile application automated testing.

7. Conclusions and Future Works

We propose a robot vision-based algorithm for GUI element recognition, namely, YOLOv5-MGC. This approach is applied to automated mobile application test modeling to improve the accuracy of robot vision recognition. On the basis of YOLOv5, we initially replace the K-means algorithm with K-means++ for target anchor box generation to enhance the feature extraction capability of the network model. Then, the attention mechanism and microscale detection layer are added to the backbone network to improve the element recognition accuracy and reduce the miss detection rate of elements. Finally, the G-Bneck is introduced into the network to reduce the training and detection speed due to the increase of network modules, achieving the effect of accelerating network training while ensuring recognition accuracy.

In the experiment, the convolutional network model YOLOv5 is used for training and testing. According to the comparison of detection performance, the YOLOv5-MGC network based on transfer learning proposed in this paper performs better in terms of recognition accuracy and precision. The mean AP (0.5) can reach 89.8%, and the recognition precision can reach 80.8%.

The method proposed in this paper still has some limitations. First, due to the continuous development and updating of GUI, some of the GUI styles in the RICO dataset used in this paper are already outdated and cannot meet the current training data required. Second, in this paper, the recognition categories are divided into eight categories, among which the others category contains many categories and is more complicated to recognize. Although the algorithm proposed in this paper has been greatly improved for the recognition of the others category, there still exists considerable room for improvement. Thus, we will continue to mine the GUI element classification and consider implementing unsupervised learning for GUI elements to address these limitations in future work.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no potential conflicts of interest.

Acknowledgments

This work is supported by the following projects: (1) Nature Science Basic Research Plan in Shaanxi Province of China (Grant nos. 2019JM-484). (2) Key Research and Development Project in Shaanxi Province of China (No. 2022GY-048). (3) Scientific Research Program Funded by Shaanxi Province Education Department (No. 19JK0414). (4) National Natural Science Foundation of China (No. 61901388, 62001386).

References

[1] C. Zhang, T. Shi, J. Ai, and W. Tian, "Construction of GUI elements recognition model for AI testing based on deep learning," in *Proceedings of the 8th International Conference on Dependable Systems and Their Applications (DSA)*, pp. 508–515, Yinchuan, China, December 2021.

[2] L. Mariani, M. Pezzè, V. Terragni, and D. Zuddas, "An evolutionary approach to adapt tests across mobile apps," in *Proceedings of the IEEE/ACM International Conference on Automation of Software Test (AST)*, pp. 70–79, Madrid, Spain, May 2021.

[3] X. Zhu, B. Zhou, J. Li, and Q. Gao, "A test automation solution on gui functional test," in *Proceedings of the 2008 6th IEEE International Conference on Industrial Informatics*, pp. 1413–1418, Daejeon, Korea (South), September 2008.

[4] R. Coppola, L. Ardito, and M. Torchiano, "Mobile testing: new challenges and perceived difficulties from developers of the Italian industry," *IT Professional*, vol. 22, no. 5, pp. 32–39, 2020.

[5] S. Singh, R. Gadgil, and A. Chudgor, "Automated testing of mobile applications using scripting technique: a study on appium," *International Journal of Current Engineering and Technology (IJCET)*, vol. 4, no. 5, pp. 3627–3630, 2014.

[6] S. Negara, N. Esfahani, and R. Buse, "Practical Android Test Recording with Espresso Test Recorder," in *Proceedings of the IEEE/ACM International Conference on Software Engineering: Software Engineering in Practice Google LLC*, Montreal, QC, Canada, August 2019.

[7] XCTest, "UI Tests for Your Xcode Project," https://developer.apple.com/documentation/xctest.

[8] A. Gunduz, N. Kose, and G. Rigoll, "Real-time hand gesture detection and classification using convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2613–2625, Lille, France, May 2019.

[9] T. A. Nguyen and C. Csallner, "Reverse Engineering Mobile Application User Interfaces with REMAUI," in *Proceedings of the 2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pp. 248–259, NE, USA, November 2015.

[10] K. Moran, C. Bernal-Cardenas, M. Curcio, R. Bonett, and D. Poshyvanyk, "Machine learning-based prototyping of graphical user interfaces for mobile apps," *IEEE Transactions on Software Engineering*, vol. 46, no. 2, pp. 196–221, 2020.

[11] S. C. Yu, C. R. Fang, and Y. Feng, "LIRAT:Layout and Image Recognition Driving Automated Mobile Testing of Cross-Platform," in *Proceedings of the 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pp. 1066–1069, CA, USA, November 2019.

[12] W. Y. Zhang, "Mobile application control detection method based on image recognition," *Computer Applications*, vol. 40, no. S1, pp. 157–160, 2020.

[13] T. Beltramelli, "Pix2code: generating code from a graphical user interface screenshot," in *Proceedings of the EICS'18: ACM SIGCHI Symposium on Engineering Interactive Computing Systems ACM*, NY, USA, June 2018.

[14] T. D. White, G. Fraser, and J. Guy, "Improving random GUI testing with image-based widget detection," in *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis*, pp. 307–317, NY, USA, July 2019.

[15] H. Zhang, M. Tian, G. Shao, J. Cheng, and J. Liu, "Target detection of forward-looking sonar image based on improved YOLOv5," *IEEE Access*, vol. 10, pp. 18023–18034, 2022.

[16] X. Zhang, F. Guo, Y. Liang, and X. Chen, "Summary of small target detection algorithms based on deep learning," *Softw. Guide*, vol. 19, no. 05, pp. 276–280, 2020.

[17] Y. Li, S. Li, H. Du, L. Chen, D. Li, and Y. Li, "YOLO-ACN: focusing on small target and occluded object detection," *IEEE Access*, vol. 8, pp. 227288–227303, 2020.

[18] C. Chen, S. Feng, Z. Xing, and L. Liu, "Gallery D. C. Design search and knowledge discovery through auto-created GUI component gallery," in *Proceedings of the ACM Hum-Comput Interact*, p. 22, NY, USA, November 2019.

[19] Z. Cai and N. Vasconcelos, “Cascade R-CNN: delving into high quality object detection,” in *Proceedings of the IEEE/CVF Conf. Comput. Vis. Pattern Recognit*, pp. 6154–6162, MD, USA, June 2018.

[20] K. He, G. Gkioxari, P. Girshick, and R. Girshick, “Mask R-CNN,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 2, pp. 386–397, 2020.

[21] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: unified real-time object detection,” in *Proceedings of the IEEE Conference on Computer Vision Pattern Recognition (CVPR)*, pp. 779–788, NV, USA, June 2016.

[22] J. Redmon and A. Farhadi, “YOLOv3: An Incremental Improvement,” 2018, https://arxiv.org/abs/1804.02767.

[23] D. Wu, S. Lv, M. Song, and H. Song, “Using channel pruning-based YOLO v4 deep learning algorithm for the real-time and accurate detection of apple flowers in natural environments,” *Computers and Electronics in Agriculture*, vol. 178, no. 5, Article ID 105742, 2020.

[24] J. Zhao, X. Zhang, and J. Yan, “A wheat spike detection method in UAV images based on improved YOLOv5,” *Remote Sensing*, vol. 3095, no. 13, 2021.

[25] L. Zhu, X. Geng, Z. Li, and C. Liu, “Improving YOLOv5 with attention mechanism for detecting boulders from planetary images,” *Remote Sensing*, vol. 3776, no. 13, 2021.

[26] H. Shi, J. Li, and J. Mao, “Lateral Transfer Learning for Multiagent Reinforcement Learning,” *IEEE Transactions on Cybernetics*, 2021.

[27] T. F. Liu, Mark Craft, and Jason Situ, “Learning Design Semantics for Mobile Apps,” in *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*, pp. 569–579, NY, USA, October 2018.

[28] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R.-C. N. N. Towards real-time object detection with region proposal networks,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, pp. 1137–1149, 2016.

[29] Li Jingchen, S. Haobin, and H. Kao-Shing, “Using Fuzzy Logic to Learn Abstract Policies in Large-Scale Multi-Agent Reinforcement Learning,” *IEEE Transactions on Fuzzy Systems*, 2022.

[30] S. Haobin, L. Jingchen, M. Jiahui, and H. Kao-Shing, “Lateral Transfer Learning for Multi-Agent Reinforcement Learning,” *IEEE Transactions on Cybernetics*, 2021.

[31] H. Shi, X. Li, K.-S. Hwang, W. Pan, and G Xu, “Decoupled visual servoing with fuzzy *Q*-learning, decoupled visual servoing with fuzzy,” *IEEE Transactions on Industrial Informatics*, vol. 14, no. 1, pp. 241–252, 2018.

[32] S. Woo, J. Park, J. Y. Lee, and I. S. Kweon, “CBAM: convolutional block attention module,” *Computer Vision-ECCV 2018*, vol. 11211, pp. 3–19, 2018.

[33] K. Han, “GhostNet: more features from cheap operations,” in *Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1577–1586, WA, USA, June 2020.

[34] J. Li, H. Shi, and K. S. Hwang, “An explainable ensemble feedforward method with Gaussian convolutional filter,” *Knowledge-Based Systems*, vol. 225, Article ID 107103, 2021.