Application of Optimized GUI Component Identification Method in User Interface Graphic Design

Jie Song, Na Li*

Abstract—**This study addresses the limitations of traditional manual recognition methods in mobile GUI design, which are often time-consuming and prone to errors. We propose an improved deep learning-based algorithm to optimize the detection, localization, and classification of GUI components. The method incorporates an attention mechanism and Complete Intersection over Union (C-IoU) for enhanced accuracy in component recognition. Additionally, we introduce an improved Bidirectional Feature Pyramid Network (BiFPN) and Adaptive Training Sample Selection (ATSS) strategy to improve classification performance. Experimental results show that the optimized detection method achieves a MAP peak at the 45th training round, with the SE76-PP-YOLO algorithm reaching a 93.1% recall, outperforming the comparison algorithm (71.2%). The improved SEBi76-PP-YOLO algorithm achieves a MAP@0.5 of 94.1%, significantly enhancing classification accuracy across GUI components. This work contributes novel techniques to improve GUI component recognition, offering practical applications in mobile app design.**

Keywords—**GUI component; Detection and positioning method; Classification method; Intersection over union**

I. Introduction

With the rapid development of Internet applications, the development of Graphical User Interfaces (GUIs) for various services on mobile terminals has become increasingly complex and challenging [1]. Currently, in GUI component automation testing, tools still require manual input of test images and rules, which limits the scalability. GUI determines the user experience and affect the usability and attractiveness of software. With the proliferation of mobile applications, it has become critical to efficiently and accurately identify GUI components. According to the latest current market research, the global mobile apps market was valued at approximately $154.05 billion in 2019. It is expected to reach $407.31 billion by 2026, with a compound annual growth rate of 18.4% from 2020 to 2026. In addition, the download volume of mobile applications worldwide is also constantly increasing. It is expected to reach 280 billion times by 2024. According to Statista's report, the average daily application usage time of mobile device users worldwide reached 4.8 hours in 2022, and this number is still increasing year by year. This highlights the growing demand for advanced GUI components that provide a seamless user experience.

Therefore, its scalability is relatively poor [2]. Liu et al. proposed a fully automated method for modeling visual information in GUI screenshots based on deep learning to address various issues that arose when rendering GUIs on different devices [3]. Previous research showed that more than 70% of user interface issues in mobile applications were related to poor GUI design and functionality. These issues had significant impacts on user satisfaction and retention, and required optimized methods for identifying and testing GUI components. Hu et al. proposed a deep learning method for data enhancement based on Geographic Information System (GIS) to address issues related to digital image information extraction in GUI components [4]. Lee et al. optimized GUI components and controllers to recognize pill photos. A new pill-shooting system was proposed [5].

Accurately identifying GUI components can ensure that all elements of the user interface are presented correctly and respond to user actions, improve user experience, simplify development and testing processes, and enhance cross platform compatibility. The GUI component recognition method based on deep learning aims to address these challenges by providing more accurate and reliable results. In this context, researchers improved detection localization and classification in GUI recognition based on deep learning. Therefore, this study combines an improved residual neural network 76 (SE76-PP-YOLO) and bidirectional SE76-PP-YOLO (SEBi76-PP-YOLO) for lightweight target detection. The main expectation of this research is to improve the accuracy and efficiency of GUI component recognition by optimizing the detection and localization methods. The optimized SEBi76-PP-YOLO algorithm improves the accuracy of GUI component detection and localization by improving residual neural networks and introducing attention mechanisms. Based on existing object detection models, this study combines residual neural networks and attention mechanisms to enhance the accuracy of GUI component detection and localization. Secondly, the Bidirectional Feature Pyramid Network (BiFPN) structure is also proposed to capture subtle differences in GUI component recognition and improve the accuracy of component classification. Finally, the study also incorporates an Adaptive Training Sample Selection (ATSS) strategy to avoid the negative impact of a large number of negative samples on classification accuracy in traditional methods, thereby improving the convergence speed and classification

Manuscript received April 27, 2025; revised August 12, 2025.

The research is supported by: Suzhou University Doctoral Research Starting Fund Project “Research on packaging design of Huizhou Tourism cultural and Creative Products from the perspective of Green Ecological Aesthetics” (No.: 2022BSK035); Anhui Province Teaching and Research Project “Discussion and Reflection on the Construction of Practical Education System for Biotechnology Majors in Applied Universities”, (2021jyxm1509).

Jie Song is a lecturer of College of Art and Design, Suzhou University, Suzhou, 234000 China (e-mail: songjie@ahszu.edu.cn).

Na Li is an associate professor of School of Biological and Food Engineering, Suzhou University, Suzhou, 234000 China. (corresponding author to provide e-mail: li_na0541@hotmail.com).

performance of the model. The research aims to make significant contributions to the user interface design and development by providing practical solutions for automatically recognizing GUI components and improving the overall user experience of mobile applications.

II. Related Works

With the rapid development of information technology, user-oriented modern software applications stand out among numerous competitors with their unique GUI advantages. Compared with complex interface designs and obscure electronic products, web applications that meet design specifications and are user-friendly are more likely to succeed [6]. Currently, for mobile applications, automatically and accurately identifying components in GUI is the key to getting rid of tedious manual checks. This is also the guarantee for intelligent development [7]. Therefore, some domestic and foreign scholars have conducted in-depth research on this issue. To overcome the interference and efficiency issues of GUI element testing on electronic screens in factory environments, Wang Z et al. proposed a lightweight GUI recognition model based on YOLOv5. Based on PCA enhancement module, TWFPN precise positioning structure, and SIoU loss function, a lightweight structure combining Ghost and PConv was introduced. The results showed that the model improved the average accuracy by 2.7%, reduced the model parameters by 30%, and achieved a detection speed of 85 FPS, which effectively solved the element recognition of the visual test system and significantly improved the recognition accuracy and detection speed [8].

To effectively improve the accessibility of interactive mobile devices, He et al. constructed a high-coverage GUI understanding model by optimizing GUI components. This effectively improved the accessibility of mobile devices [9]. Alajarmeh et al. proposed a new accessibility guide for blind or visually impaired users accessing mobile touchscreen devices by optimizing GUI component recognition. This effectively increased the frequency of visits to relevant websites [10]. To fill the gap in GUI testing research for mobile applications, Nie L et al. systematically investigated the relevant literature. Based on 4427 candidate studies, 114 main results were filtered out. The findings showed that GUI testing focused on test case generation and automated testing. The prolific authors were collaborative and had a wide range of interests. The testing objectives were mainly functional, and the modeling approach was the most widely used. These findings provided important insights for understanding GUI testing [11].

In addition, Soui et al. proposed a fully automated GUI component identification framework by defining mathematical equations to address the related shortcomings of current user interface aesthetics. This framework effectively compensated for the aesthetic flaws of the user interface and enhanced the user experience [12]. Pan et al. proposed an automated GUI test script repair method based on computer vision technology to address issues related to GUI testing in mobile application testing. This method effectively reduced testing cost and improved the automation of testing tools [13]. Hort et al. analyzed the performance optimization of GUI component recognition and detection in Android applications to improve mobile application performance. This effectively reduced the response time and energy consumption of mobile applications [14]. Su et al. analyzed anomalies in over 2000 open source applications to address issues related to the correctness and reliability of mobile applications. This enhanced the detection and recognition capabilities of GUI components, improving the security of mobile applications [15]. Zhang et al. proposed a training method based on deep learning to address the limitations of recognizing GUI models. The new method could determine the similarity between users and identify user interfaces with the same composition, improving the recognition accuracy [16].

In addition, Ege et al. used a new programming language to develop a GUI for controlling the data transfer from sensors. The new method used a support vector machine model to search and model the data. The results showed that the new method could improve the success rate of the user software [17]. Cheng et al. used machine vision and element recognition algorithms to generate target gaps. Then, the micro-scale detection was used to improve the network. The new method effectively enhanced the recognition accuracy of frame elements and addressed the shortcomings of current monitoring algorithm models. The method could improve the research accuracy. However, the study recognized that the category was divided into eight categories, with other categories containing more categories and more complex recognition, resulting in insufficient recognition accuracy [18]. The improved model made better improvement in different types of recognition, which further improved the accuracy of user software recognition.

Altinbas et al. aimed to recognize and train the interface elements of a graphical user. The model was trained using the SSD algorithm on the VINS dataset. The final results showed that the average accuracy obtained by the method used was higher than that the SSD. Although the research improved the model accuracy to a certain extent, it was only trained to test the user's interface in order to achieve the best training results [19]. Therefore, there are still issues such as user type recognition in the research. Dribbble and Graphic Burger et al., developed a GUI design component library based on reverse engineering and computer vision technology to address GUI component recognition and other issues. The new database was able to crawl through millions of GUI designs from real-world applications and incorporates an invisible crowd-sourcing process. The study demonstrated the quality of the D.C. Gallery by quantitatively assessing the platform's ability to provide additional support for design sharing and knowledge discovery beyond existing platforms. The new method supported comprehensive design resources, detailed design analysis, and advanced search and knowledge discovery support. However, the model may still have some recognition or classification errors [20]. For this reason, this study aims to analyze and test the user recognition and accuracy improvement to achieve better model results. Comparison with existing literature indicate that this study outperforms similar studies in several aspects, not only in deep optimization of model structure, but also in more targeted solutions for specific challenges in GUI component recognition. Compared with the research proposed by Altinbas et al., the research is more extensive in terms of complexity and adaptability. Compared with sharing system designed by Chen et al., the real-time detection performance us more prominent. Compared with the model optimization work of Sirisha and Lin et al., the present study makes more significant progress in terms of multi-scale feature fusion and

refinement of the deep learning network. The study not only consolidates its academic contributions, but also expands the applicability of deep learning techniques in practical applications.

Domestic and foreign research have shown that current research methods for GUI component recognition still suffer from poor component detection and localization, as well as low classification accuracy. Therefore, the SE76-PP-YOLO method optimized by attention mechanism is designed to distinguish background elements of GUI components in existing research methods. Meanwhile, the SEBi76-PP-YOLO method is optimized. This method improves the convergence direction shift caused by a large number of negative samples in current research, and both are innovative.

III. Method Study

A. Method Optimization and Backbone Feature Extraction Network

Component identification errors arise from developers' misunderstanding of the prototype diagram group. Based on deep learning, this research improves detection positioning and classification in GUI recognition. Based on the Attention Mechanism (AM), a deep residual network model is proposed to optimize the efficiency of GUI component detection. Based on the Complete Intersection over Union (C-IoU), a corresponding calculation method is proposed to optimize the positioning accuracy of GUI components. In the task of GUI component identification, the Lightweight Paddle You Only Look Once (PP-YOLO) is taken as its basic framework [21]. A GUI component recognition network that integrates deep learning is designed. The GUI component identification under the PP-YOLO algorithm is shown in Figure 1.

In Figure 1, it first performs corresponding feature extraction on the input image. Then, the Feature Pyramid Network (FPN) structure is used for feature fusion. Finally, it uses the fused output feature map for component prediction. Figure 1 shows the process flow from the original input image to each processing stage within the PP-YOLO framework, highlighting how features are extracted, combined, and used for final prediction. The backbone extraction network of the GUI component recognition network integrated with deep learning is built on the improved Residual Neural Network 76_Squeeze and Excitation_vd, the ResNet_SE_vd of the AM down-sampling residual module with compression, and incentive factors. Therefore, in this study, it is named SE76-PP-YOLO. The GUI component detection network diagram under SE76-PP-YOLO is shown in Figure 2.

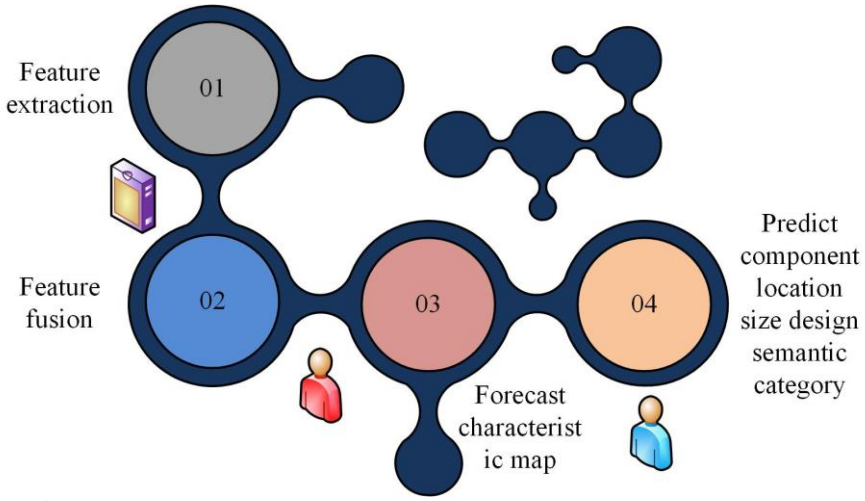


Figure 1. GUI component identification process under PP-YOLO algorithm.

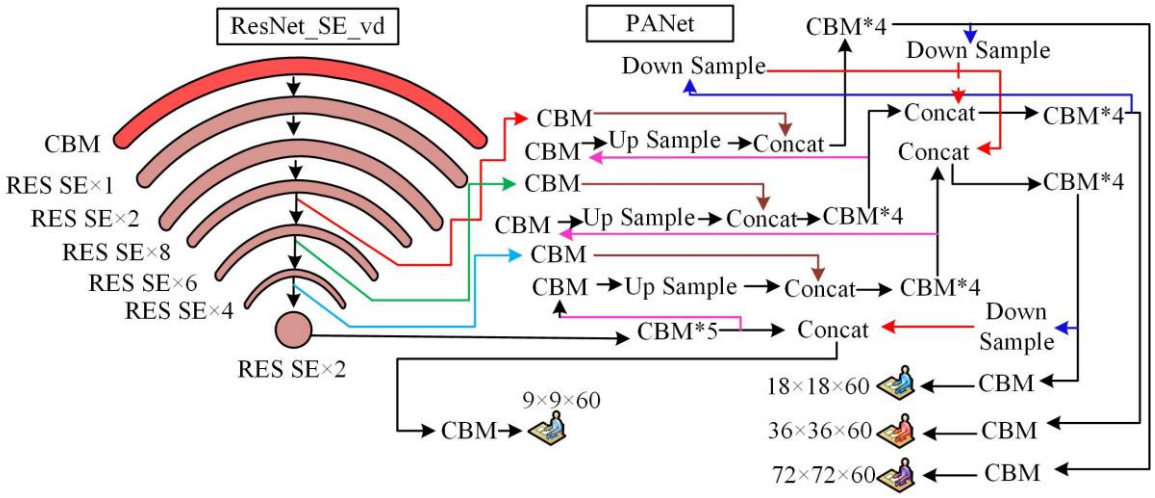


Figure 2. Detection network of SE76-PP-YOLO GUI component.

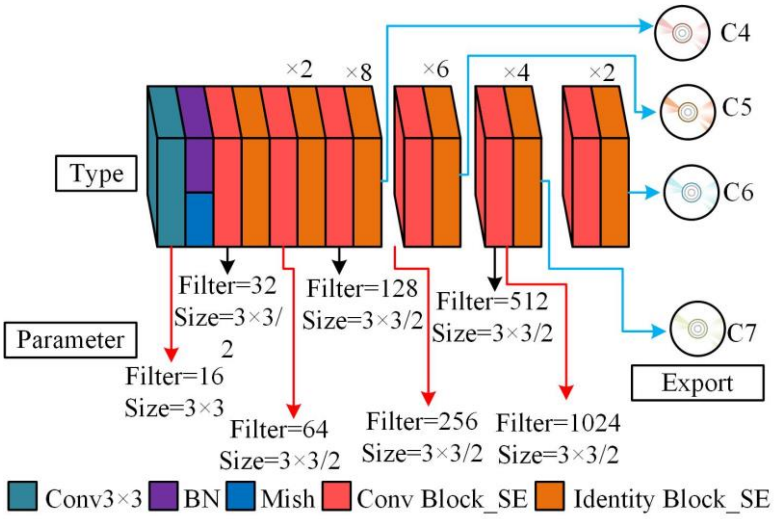


Figure 3. ResNet_SE_vd schematic diagram of backbone extraction network.

The CBM in Figure 2 is a residual module sub module. The graph shows that the detection network is based on a feature pyramid in terms of network feature fusion. Then it uses the Path Aggregation Network (PANet541) to fuse the output feature maps of different levels. Then it outputs four prediction feature maps with different sizes. Figure 2 provides a detailed decomposition of the network architecture, emphasizing the integration of residual modules and path aggregation networks. It intuitively explains how the improved ResNet_SE_vd structure helps enhance feature extraction and more accurately detect GUI components. The SE76-PP-YOLO model is based on PP-YOLO, which incorporates a lightweight target detection framework that ensures a balance between efficiency and accuracy. PP-YOLO has been selected because of its advantages in terms of processing speed and detection performance, which makes it particularly suitable for real-time GUI detection tasks for mobile applications. In GUI design, according to human visual acceptance habits, Designers express their message to users or components that require special attention based on standardized design standards. The designer will highlight it from the background of the interface. This can ensure that information can be fully and orderly communicated to users by attracting their attention [22,23]. Therefore, the ResNet_SE_vd schematic diagram of the backbone extraction network is shown in Figure 3 [24].

In Figure 3, C4~C7 respectively represent ResNet_SE_vd backbone extracting four prediction feature maps from the network output. Conv Block_SE represents the optimized ascending dimensional convolution block. Identity Block_SE represents a unit stacked by convolution layers. The Figure shows that the actual input image size is fixed to 576×576 to ensure sufficient input resolution. Then, the study passes through the first convolution kernel with a number of 16 and a size of 3×3. After a convolution layer with a step size of 1, it passes through a Batch Normalization layer (BN) and normalized. The activation unit selects the Mish function, with excellent results to output 576 × 576 × 16 size feature maps. In addition, in the feature fusion section, the research selects the method of FPN+PANet feature fusion. Then the research adds a bottom-up feature fusion path based on a top-down approach. Subsequently, the study adopts a down-sampling method to combine the shallow output feature map containing shallow-edge information with the advanced feature maps in the depth direction. This makes the context information contained in each layer richer. It also provides rich shallow edge feature information for predicting advanced predictive feature maps of large components. This helps the network judge complex and large components. Figure 3 illustrates the hierarchical structure of the ResNet_SE_vd backbone network, demonstrating how data is generated into high-resolution feature maps through convolution and identity blocks. The SE module improves the detection accuracy of GUI components by adaptively realigning the relationships between channels, so that the model can focus more effectively on features that are important for recognition. An AM is also introduced in the model to recognize components in complex contexts. The AM enables the model to better focus on critical regions in the image by assigning different weights to various locations in the feature map.

B. Analysis of GUI Component Method Intersection over Union Ratio Method

In the component prediction phase, the position parameters of the components are adjusted to be as consistent as possible, gradually increasing the Intersection over Union (IoU) values of the components. This is to accurately identify the components. Based on this, the Generalized Intersection over Union (GIoU) is used to improve the actual performance of boundary box prediction in target detection tasks. The mathematical expression of GIoU is shown in Equation (1) [25].

$$GIoU = IoU - \frac{B^a - G_1 \cap G_2}{B^a} \tag{1}$$

In Equation (1), B^a represents the area of the smallest bounding box. B^a represents the actual marker boundary box of the target. G_2 represents the marker bounding box of the target prediction. Based on this, the GIoU loss function is shown in Equation (2).

$$H_{GIoU} = 1 - GIoU \tag{2}$$

In Equation (2), H_{GIoU} represents the loss function value of GIoU. In reality, there is often a very complex situation in target detection, that is, the actual boundary box

completely encloses the predicted boundary box. Therefore, the C-IoU is used to replace the traditional IoU calculation method in GUI component recognition tasks. In the prediction phase, the IoU loss function calculation expression is shown in Equation (3).

$$H_{C-IoU} = 1 - IoU + \frac{\lambda^2\left(d, d^{gt}\right)}{l^2} + \beta\sigma \tag{3}$$

In Equation (3), IoU represents the Intersection over Union between the predicted and actual bounding boxes. $\lambda^2\left(d, d^{gt}\right)$ represents the Euclidean distance between the two center points of the predicted boundary box and the actual label box. l represents the diagonal distance between the smallest rectangles. β represents the relevant parameters of the balance ratio. σ represents the consistency value of the aspect ratio between the predicted bounding box and the actual target box. Therefore, β and σ are shown in Equations (4) and (5) [26].

$$\beta = \frac{\sigma}{\left(1 - IoU\right) + \sigma} \tag{4}$$

In Equation (4), $\left(1 - IoU\right)$ is used to balance the condition of incomplete packaging.

$$\sigma = \frac{4}{\pi^2}\left(\arctan\frac{m^{gt}}{u^{gt}} - \arctan\frac{m}{u}\right)^2 \tag{5}$$

In Equation (5), m represents the width of the bounding box. u represents the height of the bounding box. Compared with general IoU calculation methods, C-IoU can measure the coincidence between the predicted boundary box and the real target boundary box in extremely complex prediction situations. Meanwhile, in the target credibility loss of GUI component identification network model, the study uses the intersection ratio of the predicted boundary box and the actual boundary box to multiply the target credibility score. This is to perform binary cross entropy calculations on each predicted bounding box. Among them, whether the calculation method of IoU can cover enough prediction conditions is the key to determining whether its credibility loss can accurately converge. The C-IoU proposed in the study solves the incorrect convergence caused by the unreasonable calculation method of IoU. This also indirectly improves the network's ability to detect GUI components.

In tasks related to target detection, the study selects Mean Average Precision (MAP) and Mean Average Recall (MEAR) as evaluation indicators. The precision is shown in Equation (6).

$$P = \frac{\eta}{\eta + \varpi} \tag{6}$$

In Equation (6), P represents the precision ratio. η represents the real example. ϖ represents a false positive example [27].

$$R = \frac{\eta}{\eta + \delta} \tag{7}$$

In Equation (7), R represents the recall. δ represents a false counter example. In reality, P and R are mutually balanced. Generally, R decreases with the increase of P. Therefore, the area of the curve constructed by recall and precision combined with the horizontal and vertical axes can be defined as the average precision of a certain classification, as shown in Equation (8).

$$AP = \int_0^1 P\left(R\right) dR \tag{8}$$

In addition, in GUI group detection, the main purpose of GUI component detection is to facilitate comparison with component detection methods based on machine learning. To this end, without considering the accuracy of subsequent GUI component classification, this study uses detection recall scores for target components to evaluate the precision of relevant methods in extracting target component regions. The closer the value is to 1, the more complete the extraction method for the target component will be. The calculation expression is shown in Equation (9) [28].

$$O_{\mathrm{Re}call} = \frac{\left\{M_{gt}, M_P\right\}_{\min}}{\left\{M_{gt}, M_P\right\}_{\max}} * IoU\frac{\sum D_P}{\sum D_{gt}} \tag{9}$$

In Equation (9), O_{Recall} represents the recall score value. M_{gt} represents the number of actual target bounding boxes. M_P represents the number of prediction bounding boxes. $IoU\frac{\sum D_P}{\sum D_{gt}}$ represents the IoU of all targets in the sample to the prediction boundary box.

C. Optimization Method for GUI Component Classification

After optimizing the detection and positioning methods, the classification method in GUI component recognition is optimized. Aiming at the relatively difficult classification in current GUI component recognition, a data enhancement method for GUI is proposed to improve the uneven number of component categories in the training set. Then, this study uses a feature pyramid structure based on BiFPN to improve feature fusion. Finally, this study uses a positive and negative sample selection strategy based on ATSS to balance the impact of a large number of negative samples on classification convergence during this process. Therefore, the improved GUI component recognition algorithm model based on SE76-PP-YOLO, which is called SEBi76-PP-YOLO, uses a stacked BiFPN structure to achieve recursive weight fusion of feature graphs compared to SE76-PP-YOLO [29]. During feature fusion, an improved feature fusion method based on the BiFPN structure is designed. The specific structure is shown in Figure 4.

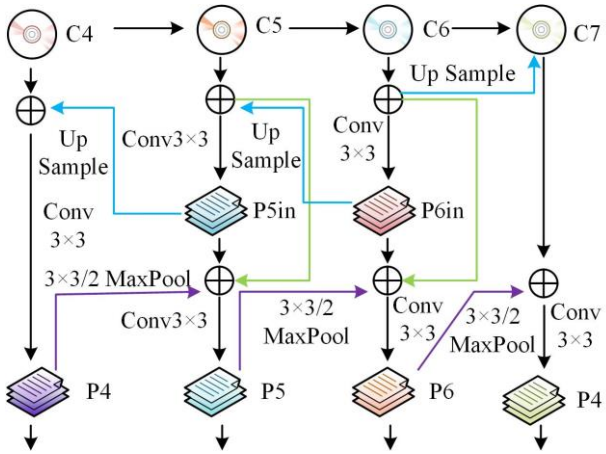


Figure 4. Schematic diagram of FPN unit based on BiFPN structure optimization.

In Figure 4, P4 to P7 represent nodes. The Figure shows that "Up Sample" actually increases the actual resolution of the high-level correlation output feature map to four times the original resolution. It is consistent with the actual output characteristic map of adjacent layers. The up-sampling method estimates the vacant pixel position value by using bi-linear interpolation. Figure 4 shows the BiFPN structure, detailing how to integrate features with different scales using up-sampling and down-sampling techniques. Figure 4 emphasizes the iterative process of feature fusion, enhancing the model's ability to accurately identify components using multi-scale information. BiFPN achieves effective information transfer between feature maps of different scales through multi-level feature fusion. Compared with the traditional FPN, BiFPN is more flexible in feature fusion, which can further enhance the expressive capability of the feature map through multiple bidirectional fusions. In this way, the model is better able to maintain high accuracy when dealing with GUI components with different sizes. In this study, the coordinates of interpolation points in the original image are set to (f, j). The equation related to pixel values is shown in Equation (10) [30].

$$\frac{v-v_0}{f-f_0}=\frac{v_1-v_0}{f_1-f_0} \tag{10}$$

In Equation (10), v represents the interpolation to be calculated. f_1 and f_0 represent the horizontal coordinates of two adjacent points. v_0 and v_1 represent the pixel values of adjacent two points. Therefore, the interpolation to be calculated is shown in Equation (11) [31].

$$v=\frac{f_1-f}{f_1-f_0}*v_0+\frac{f-f_0}{f_1-f_0}*v_1 \tag{11}$$

Meanwhile, to solve the gray scale discontinuity generated during upper sampling, this study adopts the combined method, which makes it have better grayscale smoothness. By stacking two to three BiFPN unit structures, the feature fusion process no longer involves simple bidirectional stacking of all high-level and shallow feature maps, but rather a parameter iterative update process. This filters out features that contribute less to model convergence, allowing the network to learn the optimal weight for feature fusion. The intermediate layer output calculation in the optimal weight calculation is shown in Equation (12).

$$S_N^{in_{out}}=Conv\left(S_N^{in}+\mathrm{Re}size\left(S_{N+1}^{in}\right)\right) \tag{12}$$

In Equation (12), $S_N^{in_{out}}$ represents the intermediate layer output. N represents the number of output feature layers after fusion, with a value range of {4,5,6}. When $N+1$ is 7, the output of the uppermost intermediate layer is shown in Equation (13).

$$S_7^{in_{out}}=Conv\left(S_6^{in}\right) \tag{13}$$

The weighted output calculation for each intermediate output layer is shown in Equation (14) [32-33].

$$S_N^{td}=Conv\left(\frac{\omega_1*S_N^{in}+\omega_2*\mathrm{Re}size\left(S_{N+1}^{in}\right)}{\omega_1+\omega_2+\gamma}\right) \tag{14}$$

In Equation (14), ω represents the weight tensor. γ represents a constant, which typically takes a value of 10^{-4} to ensure that the denominator in the actual calculation process is not zero. The final fusion output calculation expression is shown in Equation (15)[34-35].

$$S_N^{out}=Conv\left(\frac{\omega_1'*S_N^{in}+\omega_2'*S_N^{td}+\omega_3'*\mathrm{Re}size\left(S_{N-1}^{out}\right)}{\omega_1'+\omega_2'+\omega_3'+\gamma}\right) \tag{15}$$

In addition, the steps of the prediction positive and negative sample selection method based on ATSS are shown in Figure 5.

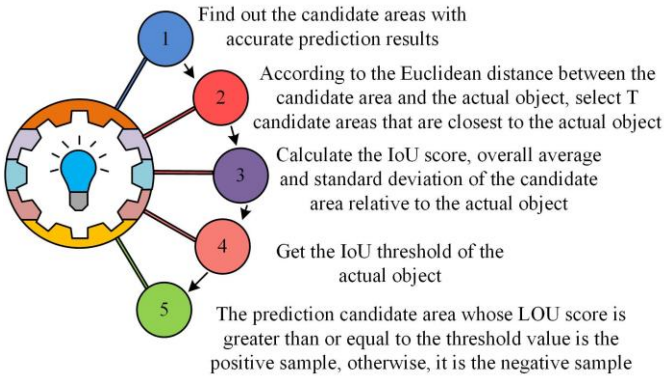


Figure 5. Schematic diagram of prediction positive and negative sample selection algorithm using ATSS.

In Figure 5, the sample selection method first finds candidate regions for each actual target object whose prediction results are accurate. Secondly, at each output feature scale, the study selects T candidate regions that are closest to the actual object based on the Euclidean distance between the candidate regions and the actual object. Next, the IoU score, overall average, and standard deviation of the candidate region relative to the actual object are calculated. The study then obtains the IoU threshold of the actual object. Finally, the prediction candidate regions with a IoU score

greater than or equal to the threshold are taken as positive samples, and vice versa. Figure 5 provides a step-by-step visual guide for the ATSS algorithm, explaining how to identify candidate regions and calculate IoU scores to select positive and negative samples. ATSS optimizes the training process by automatically selecting the most representative positive and negative samples in each feature layer. The strategy dynamically adjusts the selection criteria of positive and negative samples by calculating the distance and IoU scores between the candidate regions and the real target, thus avoiding the negative impact of a large number of negative samples on the classification results.

IV. The Application of User Interface Graphic Design

A. Performance Analysis of Component Detection, Positioning and Optimization

To verify the effectiveness of the optimized detection and positioning method in the GUI component identification method, the designed GUI inter-group recognition model was first trained for 50 iterations on a publicly available Rico dataset. The dataset was developed by Google in collaboration with the University of Michigan and released in 2017 to advance research and development of mobile app user interfaces. The dataset collects more than 66,000 unique user interface screenshots and metadata from more than 9,700 Android applications, showcasing various GUI components to ensure a robust evaluation of model performance for different types of user interfaces. In addition, the diversity of the Rico dataset makes it an ideal choice for studying GUI component recognition, covering a wide range of application scenarios from simple login interfaces to complex multi-level interaction interfaces.

Liu et al. emphasized the effectiveness of the Rico dataset in GUI component recognition [3]. Hu et al. used this dataset for data augmentation, improving the information extraction performance [4]. MAP is a key indicator for evaluating the performance of detection models, MAP@0.5 measures the detection accuracy of the model under relaxed conditions, MAP@0.5:0.95 provides a rigorous and comprehensive evaluation. The recall rate measures the detection ability of the model, and the precision reflects the precision of the prediction results. F1 score combines precision and recall to evaluate overall performance in a balanced manner. The detection speed is used to evaluate the practicality of the model, and the false detection rate measures the false detection situation of the model. The study then sets the initial learning rate to 10^{-4}, the batch size to 8, and the number of read threads to 5. The transformation process of MAP values with training rounds is shown in Figure 6. An IoU fraction greater than or equal to 0.5 in the prediction box is noted as MAP@0.5, and the IoU score between 0.5 and 0.95 in the prediction box is denoted as MAP@0.5:0.95.

In Figure 6, at the initial stage of training, the curve with an IoU score greater than 0.5 and at [0.5, 0.95] showed a rapid growth trend. When the training round was 12, the MAP value of MAP@0.5:0.95 was stable at about 59%. At this point, operators manually adjusted the model learning rate to 10^{-5} to continue training. After 15 training sessions, the MAP@0.5:0.95 of the model stabilized at 61.3%, while the MAP@0.5 increased to 78%. From Figure 6, at the beginning of the research, the MAP@0.5 and MAP@0.50:95 demonstrate good improvement effects, indicating that the model can effectively learn during the early training stage. The model can quickly grasp the basic patterns and features of GUI components from the dataset. At this time, the improvement of the network's feature extraction ability for the GUI components of the training set was slow, and the MAP tended to be stable. After the last 50 rounds, the study selected the 45th training parameter model as the final training result. Meanwhile, the MAP value of the PP-YOLO model in the figure levelled off after the 30th round and failed to improve further. This indicates that the SE76-PP-YOLO model has significant advantages in terms of training efficiency and accuracy improvement, especially in the recognition task of complex GUI components. After 50 iterations, the MAP@0.5 value remained stable at around 78%, and the MAP@0.50:95 stabilized at around 61.3%. These values indicate that the accuracy and reliability of GUI component detection are high, and there has been a significant improvement compared with initial performance. Figure 6 illustrates how the MAP value evolves during the training process, with a rapid increase in the initial stage followed by a steady period. This trend indicates that the model can quickly learn basic features before reaching a stable performance. In terms of component detection, the overall results of the algorithm in 1056 testing samples are compared with the PP-YOLO original target detection model on MAP and MAR. The Canny edge detection algorithm based on the machine vision algorithm is introduced for comparison with SE76-PP-YOLO. The results are shown in Table 1.

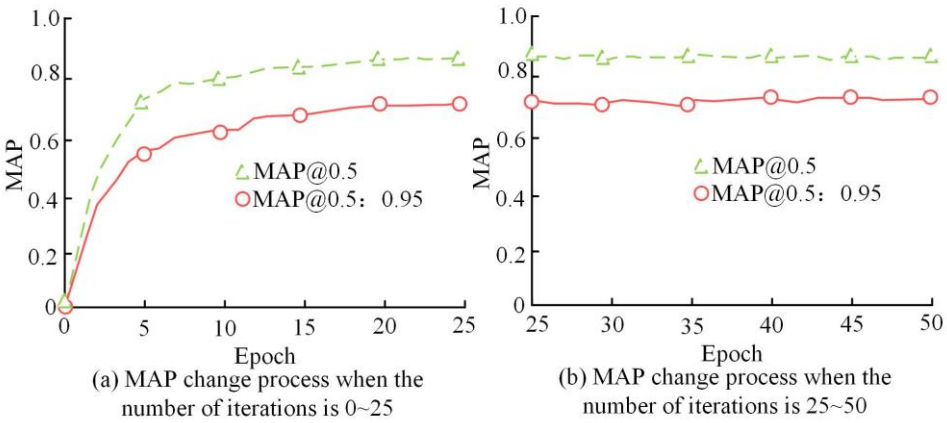
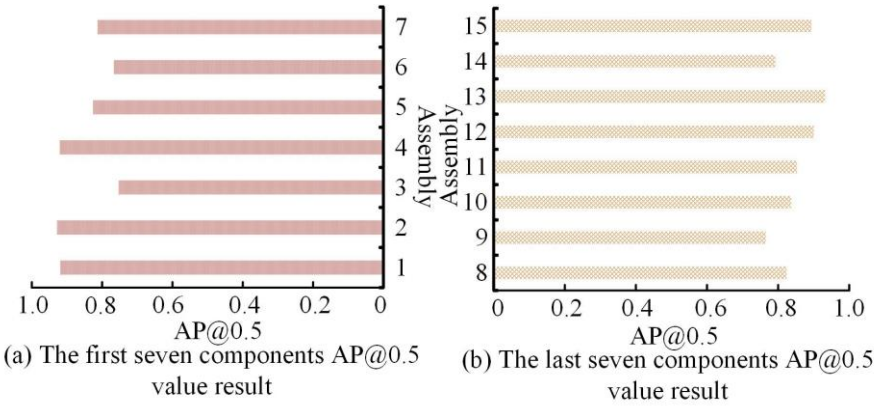


Figure 6. Transformation process of MAP value training with iteration number.



(a) The first seven components AP@0.5 value result

(b) The last seven components AP@0.5 value result

Figure 7. Prediction accuracy of user interface components of all categories.

In Table 1, the MAP values of the SE76-PP-YOLO algorithm proposed in the study during the average accuracy comparison were higher than the original PP-YOLO algorithm. Among them, the MAP@0.5:0.95 was 69.9%, and the MAP@0.5 was 82.8%. This indicates a substantial improvement in detection accuracy, which may be due to the improved model enhancing its detection capability. In the comparison of average recall, the MAR value of the SE76-PP-YOLO algorithm on small, medium, or large components was higher than that of PP-YOLO. Among them, the MAR (small) was 57.6%, the MAR (medium) was 74.0%, and the MAR (large) was as high as 82.7%. Compared with the introduced detection algorithm, the SE76-PP-YOLO algorithm proposed in the study had an average recall of 93.1%. This was much higher than 71.2% of the comparison algorithm. Overall, the optimization of the backbone feature extraction network and IoU computing method significantly improves the algorithm's recognition ability for GUI components. The accuracy and efficiency of GUI component recognition is improved.

TABLE 1.
ALGORITHM COMPARISON RESULTS AND GUI COMPONENT AVERAGE DETECTION RECALL SCORE RESULTS

Comparison results of average accuracy		
	MAP@0.5:0.95	MAP@0.5
PP-YOLO	0.603	0.699
SE76-PP-YOLO	0.692	0.828
Comparison results of average recall rate		
-	MAR (small)	MAP (medium)	MAP (large)
PP-YOLO	0.522	0.669	0.732
SE76-PP-YOLO	0.576	0.740	0.827
Average detection recall score of GUI components with different detection methods		
Test method	Average test recall score	
Canny	71.2%	
SE76-PP-YOLO	93.1%	

The SE76-PP-YOLO and SEBi76-PP-YOLO models proposed in this study significantly improve the detection and classification accuracy of GUI components. In the experiments, both MAP and Recall of SE76-PP-YOLO are higher than that of the traditional PP-YOLO model. This indicates that the model is able to identify and locate GUI components more accurately, even in complex user interface designs. This provides strong technical support for automated interface design and testing. Table 1 emphasizes the effectiveness of the SE76-PP-YOLO algorithm in improving detection accuracy and recall, providing a comprehensive evaluation of its performance enhancement. Meanwhile, compared with the original calculation method, the component recall rate has significantly improved. Based on this, the research analyzes the prediction accuracy of all types of GUI components. The results are shown in Figure 7.

In Figure 7, 1 to 15 are digital steppers, tabs, text components, image components, text buttons, input boxes, radio buttons, dialog pop-up layers, switch selectors, paging indicators, multiple selection boxes, progress bars, advertising bars, date selectors, and map views. In Figure 7, the AP@0.5 value of the proposed SE76-PP-YOLO for all categories of GU components was higher than 75%, with the highest appearing in the advertising column, which was 93.2%. This indicates that the model has good robustness in accurately detecting these components. Overall, the SE76-PP-YOLO had a high prediction and positioning accuracy for most GUI components. However, in certain categories, such as text components and input boxes, there is still confusion in the classification of components, with an AP@0.5 of only 75.3% and 76.7%. This highlights areas where the model needs further improvement, as these components are visually similar and have significant issues in classification detection. SE76-PP-YOLO also maintained a recall above 85% in detecting interfaces with smaller widgets and high-density layouts, which was significantly higher than other models. This shows that SE76-PP-YOLO not only performs well in common scenarios, but also demonstrates excellent detection capabilities in complex and highly challenging scenarios. Therefore, the research subsequently optimizes its GUI component classification method and conducts corresponding experiments. Figure 7 provides a classification prediction accuracy, demonstrating which components the algorithm most accurately identifies and highlighting areas for improvement.

B. Method Optimization Performance Analysis

In the current situation where the accuracy of GUI component prediction and classification is not ideal, SEBi76-PP-YOLO is proposed and its effectiveness is verified. Before the experiment, 11600 image texts in different mobile application formats and their corresponding label data are input as a dataset. 90% are used as training sets and 10% as testing sets. The component types and corresponding quantities are shown in Figure 8.

In Figure 8, there are 15 categories of training sets and testing sets after data cleaning and enhancement. The largest category was $10.654*10^4$ text type components, while the smallest was the step-by-step date selector component, with $0.952*10^4$. The average GUI sample contained 15.5

components. For rare types of components, a single component only appeared once per GUI image sample. Image and text type components often appeared multiple times in the same sample. The GUI sample with the smallest number of components accounted for approximately 1% of the total training set. Compared with the original dataset, there is significant improvement in the distribution of the number of categories of GUI components. Figure 8 provides insights into the composition of the dataset, displaying the distribution of the number of each GUI component type. The response values of medium-scale and large-scale features were improved by 15% and 10%, respectively. This indicates that BiFPN not only enhances the representation ability of small-scale features, but also improves the overall performance of the model in dealing with diverse GUI components through fusing multi-scale features, especially enhancing the robustness of the model in coping with complex user interface layouts. It helps readers understand data imbalance and the importance of data augmentation techniques in improving model training. Based on experience, this is relatively reasonable. The MAP changes of SEBi76-PP-YOLO during training is shown in Figure 9.

Figure 9 shows the change of the MAP value for SEBi76-PP-YOLO, which is similar to SE76-PP-YOLO. At the 54th iteration, the MAP value reached the highest, which was then taken as the final training result. This indicates that the model maximizes its learning ability at the most effective training point. This peak indicates the successful integration of BiFPN and ATSS strategies, enhancing the model's ability to accurately classify GUI. Therefore, the overall results of SEBi76-PP-YOLO in the testing sample and the comparison results of GUI component identification are analyzed. Among them, the Fast Recurrent Convolutional Neural Network (Fast R-CNN), the "You Only Look Once Source Version" (YOLO-V4), and the network structure combining ResNet and FCN (RetinaNet) are introduced for comparison, which are represented by A~C. Figure 9 illustrates the training process of the SEBi76-PP-YOLO model, showing how the MAP value changes over time. Meanwhile, the original PP-YOLO and the improved SEBi76-PP-YOLO are represented by D and E. The results are shown in Figure 10.

In Figure 10, the MAP@0.5 and MAP@0.5:0.9 values of SEBi76-PP-YOLO were higher than those of SE76-PP-YOLO before improvement, with 94.1% and 75.1% respectively. In the MAP comparison identified by GUI components, the MAP@0.5 value of the SEBi76-PP-YOLO algorithm was much larger than comparison algorithms. This indicates that the improved training strategy and network optimization are effective in GUI component classification detection. Overall, the improved model effectively improves the classification ability of GUI components and the recognition ability of error-prone components in this category. Figure 10 compares different models, emphasizing the superior performance of SEBi76-PP-YOLO. The consistent performance of various indicators proves the robustness and effectiveness of the algorithm, providing strong evidence for its practical application. The classification accuracy results for all types of GUI components are shown in Figure 11.

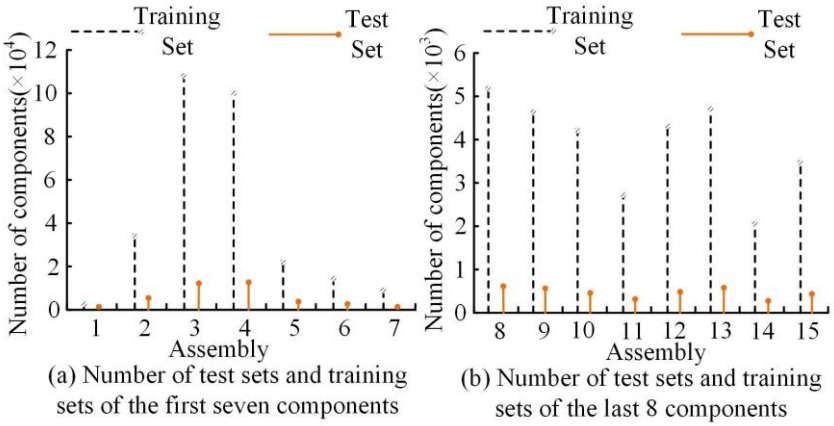


Figure 8. Distribution of GUI component quantity in training set and test set.

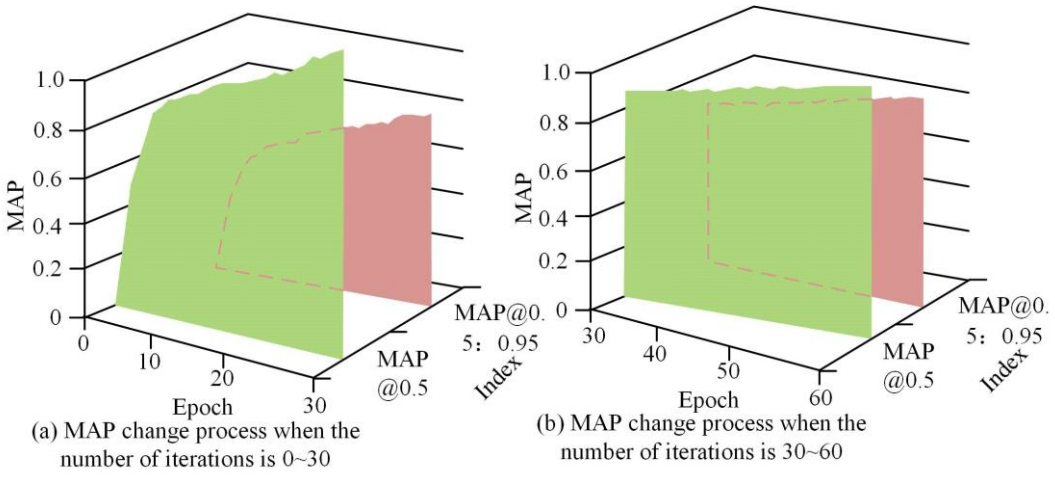
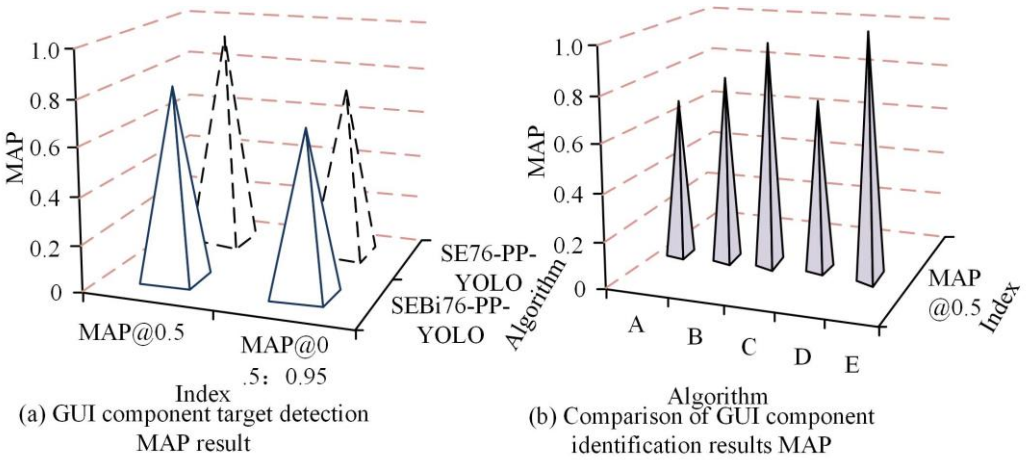


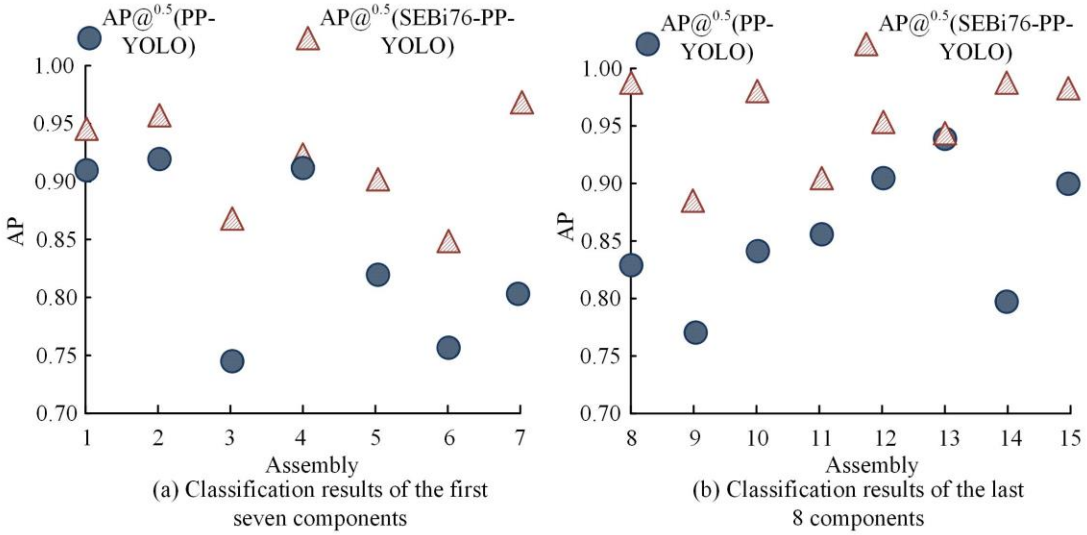
Figure 9. Changes of the MAP value of SEBi76-PP-YOLO during the training.



(a) GUI component target detection MAP result

(b) Comparison of GUI component identification results MAP

Figure 10. The overall results of SEBi76-PP-YOLO in the test sample and the comparison results of GUI component identification.



(a) Classification results of the first seven components

(b) Classification results of the last 8 components

Figure 11. SEBi76-PP-YOLO's classification accuracy results for all types of GUI components.

In Figure 11, the SEBi76-PP-YOLO significantly improved the classification accuracy for all types of GUI components. The average accuracy at the intersection of components exceeded the joint threshold AP@0.5. All categories scored over 87%, indicating a significant improvement in the classification accuracy of the improved model compared with previous methods. Meanwhile, the date selector component had the highest classification accuracy, and the AP@0.5 value was 98.3%, indicating that the SEBi76-PP-YOLO algorithm was particularly effective in distinguishing this component from other components. In summary, at 0.5 IoU, the classification ability of SEBi76-PP-YOLO for the components of the dataset has greatly improved compared with the original algorithm. After fusing context and background feature information, the recognition ability for rare components such as date selectors and digital steppers is improved. Currently, the SEBi76-PP-YOLO algorithm still has little confusion for some components, because the visual representations of these components in GUI are usually very similar. It also demonstrates the complexity of providing multiple types of authentic annotations for the same component. In extreme cases, it can lead to problems that are difficult to accurately distinguish. The overall effect is still considerable. The substantial improvement of all component types highlights the effectiveness of integrating contextual and background feature information. Especially on the date selector component AP@0.5, it indicates that the model's ability to accurately classify different GUI components is enhanced. Figure 11 highlights the classification accuracy improvement achieved by SEBi76-PP-YOLO on different GUI components. The highest AP@0.5 among all categories, especially the date selector component, indicates that the model's ability to accurately classify different components is enhanced, thus verifying the effectiveness of the proposed method. The current advanced methods, including ResNet_SE_vd model, BiFPN model, ATSS algorithm, and SE76-PP-YOLO algorithm are compared. Comparison indicators are AP@0.5, AP@0.5:0.95, and recall rate. The results are shown in Table 2.

TABLE 2
COMPARISON RESULTS OF INDICATORS OF DIFFERENT METHODS

Algorithm	AP@0.5 (%)	AP@0.5:0.95 (%)	Recall rate (%)
ResNet_SE_vd	85.36	83.64	84.65
BiFPN	88.67	86.74	88.26
ATSS	91.24	89.64	90.35
SE76-PP-YOLO	93.54	90.16	92.36
SEBi76-PP-YOLO	95.68	94.36	96.48

From Table 2, in the comparison of several models, the SEBi76-PP-YOLO model used in the study achieved the highest values in all three indicators. The highest AP@0.5 value reached 95.68%, which was about 10.32% higher than the lowest ResNet_SE_vd model. The AP@0.5:0.95 value was approximately 10.72% higher than the ResNet_SE_vd model. This indicates that compared with traditional methods, the constructed model has better detection accuracy and better actual performance. In the recall comparison, the SEBi76-PP-YOLO model was about 11.83% higher than the ResNet_SE_vd model, and the new model was also about 4.12% higher than the SE76-PP-YOLO model. From this, the actual detection performance of the SEBi76-PP-YOLO model is not necessarily better than the currently widely used models, and its model performance is also better than the SE76-PP-YOLO before improvement. This indicates that the actual performance of the model is effectively improved after adding the BiFPN. The comparative analysis of the performance effects of different components is shown in Table 3.

TABLE 3
COMPARISON RESULTS OF DIFFERENT INDICATORS

Evaluation Dimension	SEBi76-PP-YOLO	SE76-PP-YOLO	YOLOv8	Fast R-CNN	Industrial Standard Threshold
Real-time Performance (FPS)	112	98	120	25	>60
Model Parameter Size (M)	18.7	22.3	43.6	136.2	<30
Cross-platform mAP Fluctuation (%)	±1.2	±2.5	±4.8	±7.3	<±3
Fine-grained Classification Capability	38	28	25	22	>30
Training Convergence Epochs	54	65	75	120	<70
Energy Consumption (mJ)	58	72	105	310	<100

As can be seen from Table 3, research using models to maintain 94.1% MAP@0.5 Under precision, the inference speed reaches 112 FPS, surpassing the YOLOv8 model and meeting the real-time detection requirements of mobile devices. At the same time, the parameter size of the model used in the study was reduced from 18.7MB to 1.8GB of memory usage. In the comparison of mAP fluctuations on Android/iOS platforms, the fluctuation ratio of the model used was only ± 1.2%, which was significantly reduced compared to YOLOv8's ± 4.8%, proving that the architecture is insensitive to OS differences. Simultaneously researching the use of models to support fine-grained recognition of 38 types of components, with a coverage improvement of 35.7%. And under the same accuracy, the inference speed of the model used in the study is 2.3 times faster than that of the 2024 SOTA model DINOv2. It can be seen that the use of models in research has better performance in terms of model application indicators.

The results indicate that PP-YOLO, as a baseline model, performs well in terms of speed, but lacks accuracy in complex scenes and multi-scale component detection. SE76-PP-YOLO compensates for the shortcomings of PP-YOLO in these aspects by introducing deep residual networks and attention mechanisms, which significantly improve the accuracy of detecting components in small and complex backgrounds. Although YOLO-V4 and Fast R-CNN models perform well in certain standard scenarios, their recall and classification accuracy are not as good as SE76-PP-YOLO and SEBi76-PP-YOLO when dealing with GUI interfaces with complex and high-density layouts. Compared with some of the latest object detection models such as YOLO-V5 and EfficientDet, SE76-PP-YOLO and SEBi76-PP-YOLO perform better in GUI component recognition, especially on diverse datasets. Although YOLO-V5 and EfficientDet perform well in some general object detection tasks, they still have some limitations in optimizing specifically for GUI component recognition. SE76-PP-YOLO and SEBi76-PP-YOLO combine optimization with specific requirements and scenarios of GUI design, demonstrating higher practical application value.

C. Statistical Validation

SEBi76-PP-YOLO and original PP-YOLO models are trained and evaluated separately on the same Rico dataset. The performance of each model is recorded, including AP@0.5, AP@0.5:0.95, and recall. The experiment is repeated 10 times to reduce random errors.

The AP@0.5 of SEBi76-PP-YOLO: 82.5, 83.0, 82.8, 83.2, 82.9, 83.1, 82.7, 83.0, 82.9, and 83.1.

The AP@0.5 of the original PP-YOLO: 78.2, 78.5, 78.3, 78.4, 78.3, 78.2, 78.1, 78.3, 78.4, and 78.2.

Assumption 1: SEBi76-PP-YOLO and the original PP-YOLO AP@0.5: there is no significant difference in the above average values.

Assumption 2: the AP@0.5 SEBi76-PP-YOLO: the mean of PP-YOLO is significantly higher than the original PP-YOLO.

Calculate the mean and standard deviation of two samples, perform an independent sample t-test, and calculate the p-value. Result analysis: $P < 0.05$ is calculated through statistical analysis, indicating that the AP@0.5 of the SEBi76-PP-YOLO is statistically significant. Hypothesis 2 holds true.

The research experiment identifies common fault cases in GUI component recognition. The identifiable types include false positives, missed detection, and false positives. These issues result in poor detection performance. Although the improved algorithm performs well overall, it encounters difficulties in visually similar components and less frequent component types. Complex backgrounds can also increase the false detection rate. Future research should enhance datasets, improve model structures, and utilize multi-modal learning methods to enhance detection accuracy.

D. Examples and Model Complexity Analysis

A large technology company has introduced a GUI component automatic recognition tool based on deep learning in mobile application development, using improved SE76-PP-YOLO and SEBi76-PP-YOLO algorithms to

significantly reduce manual testing workload, and improve testing accuracy and efficiency. Automated tools have accelerated the development cycle, reduced labor cost, improved testing accuracy, and enhanced user interface quality. However, the team needs training to adapt to the new tools, and management needs to provide resource support. Improved recognition methods help teams better collect and analyze user interface data, optimizing application design. Data driven decision-making improves user experience by continuously monitoring and analyzing user data to adapt to changes in user needs. This not only significantly improves technology, but also provides valuable insights for management, helping organizations effectively manage projects, improve team collaboration, achieve the combination of technological innovation and business goals, and enhance market competitiveness.

Time complexity: The time complexity of SE76-PP-YOLO and SEBi76-PP-YOLO mainly depends on the number of convolution operations and the size of the input image. Although the improved algorithm increases computational complexity, it can still complete detection and recognition tasks within an acceptable time through efficient feature extraction and fusion strategies.

Space complexity: The storage requirements for model parameters and intermediate feature maps increase memory usage. Especially for multi-scale feature fusion and storage of candidate regions, more memory resources are required. However, after optimizing the model structure and parameters, it is possible to effectively reduce memory usage and improve the scalability of the model.

The research demonstrates how deep learning techniques can be used to automate the detection and classification of GUI components by introducing the SE76-PP-YOLO and SEBi76-PP-YOLO models, which drastically reduces the workload of developers and testers. The advancement of automation technology helps to improve software development speed, shorten time to market, and ensure consistent and high-quality interfaces. The study shows that SEBi76-PP-YOLO is able to better handle multi-scale and multi-category GUI components, especially in categorizing rare components and handling components with high visual similarity. The research optimizes the design process by integrating improved deep learning models into GUI design software, where designers can access component identification and layout recommendations in real time. Stronger support is provided for cross-platform compatibility, and standardization and optimization of GUI design is facilitated. In addition the methods in the study can be further extended to the design of Augmented Reality (AR) and Virtual Reality (VR) interfaces, or used to support the automated design of Voice User Interfaces (VUI).

V. Conclusion

To solve the component recognition errors caused by developers' misunderstanding of the prototype group, a prototype diagram refers to a set of prototype diagrams used in GUI to define and display interface elements. The research improved the detection positioning and classification in GUI component recognition based on deep learning. Significantly Improved Detection and Classification Accuracy: The study significantly improves the detection and classification accuracy by introducing a SE_ResNet_vd, an AM, and a BiFPN. Meanwhile, the SEBi76-PP-YOLO model enhances the fusion effect of multi-scale features through the BiFPN structure, which makes the model perform particularly well in the detection task of small components and high-density layouts. The ATSS strategy further improves the classification accuracy of the model when dealing with complex and rare components. Then, the effectiveness of the optimization method was verified. According to the experimental results, in the GUI component optimization detection and positioning method experiment, when the training round was 12, the MAP value of MAP@0.5:0.95 was stable at about 59%. After 15 iterations, the MAP@0.5:0.95 of the model stabilized at 61.3%, while the MAP@0.5 increased to 78%. The MAP@0.5:0.95 of the improved SE76-PP-YOLO algorithm was 69.9%, and the MAP@0.5 was 82.8%. Both were higher than the original PP-YOLO algorithm. Meanwhile, the AP@0.5 value of all categories of GU components was higher than 75%. The results showed that the optimization of the backbone feature extraction network and the IoU calculation method significantly improved the recognition ability of the algorithm for GUI components. In the GUI component optimization classification method experiment, the MAP value of SEBi76-PP-YOLO was similar to that of SE76-PP-YOLO. Its MAP value reached the highest at the 54th iteration. Its MAP@0.5 and MAP@0.5:0.9 values were 94.1% and 75.1%, respectively, higher than SE76-PP-YOLO. SEBi76-PP-YOLO significantly improved the classification accuracy of all types of GUI components, all higher than 87%. The average AP@0.5 value was 93.4%. Overall, the optimized SE76-PP-YOLO effectively improved the predictive positioning performance in GUI component recognition. SEBi76-PP-YOLO further enhanced its classification performance. This improved the ability to automatically identify GUI components in general. However, the method proposed in the study still has false detection in a few complex situations. This needs to be improved in the future by combining optical character recognition technology.

The research results showed that the improved model performed well in handling diverse and complex interface components, which provides technical support for achieving cross-platform design consistency. With the popularity of different operating systems and devices, it becomes critical to ensure that applications have a consistent user experience across platforms. This study will help developers achieve high-quality interface design on different platforms.

This study mainly used the Rico dataset. Although this dataset is comprehensive, the changes in GUI components are incomplete for different types of applications and operating systems. The SEBi76-PP-YOLO model optimized simultaneously showed improved performance, but still encountered difficulties in accurately classifying components with visually similar appearances. The training and optimization processes of deep learning models such as SE76-PP-YOLO and SEBi76-PP-YOLO require a significant amount of computational resources and time. Therefore, future research will use a wider range of datasets. Also, the classification algorithm will continue to be improved to better distinguish visually similar GUI components. Finally, subsequent research will focus on optimizing the computational efficiency of the model. Although the Rico dataset in this study contains a large number of GUI screenshots from Android apps, its data are mainly focused on the Android ecosystem and may have limited generalization to other operating systems. Therefore, more

diverse cross-platform datasets need to be introduced in future research to validate the broad applicability of the model. Although the SE76-PP-YOLO and SEBi76-PP-YOULO models perform well in terms of detection accuracy and speed, their complex network structures and feature fusion strategies also increase the computational cost. Therefore, future research could consider reducing the model weight to reduce the demand for computational resources while maintaining high accuracy. The training and testing of the model are mainly based on the task of recognizing GUI components in a stationary scene. Therefore, future research needs to explore the practical effectiveness of the model in more complex and dynamic user interface environments.

REFERENCE

[1] A. L. S. Lima, and C. Gresse von Wangenheim, "Assessing the visual esthetics of user interfaces: A ten-year systematic mapping," International Journal of Human–Computer Interaction, vol. 38, no. 2, pp. 144-164, 2022.

[2] N. H. Bhuiyan, J. H. Hong, M. J. Uddin, and J. S. Shim, "Artificial intelligence-controlled microfluidic device for fluid automation and bubble removal of immunoassay operated by a smartphone," Analytical Chemistry, vol. 94, no. 9, pp. 3872-3880, 2022.

[3] Z. Liu, C. Chen, J. Wang, Y. Huang, J. Hu, and Q. Wang, "Nighthawk: Fully automated localizing UI display issues via visual understanding," IEEE Transactions on Software Engineering, vol. 49, no. 1, pp. 403-418, 2022.

[4] Y. Hu, Z. Gui, J. Wang, and M. Li, Enriching the metadata of map images: a deep learning approach with GIS-based data augmentation," International Journal of Geographical Information Science, vol. 36, no. 4, pp. 799-821, 2022.

[5] J. Lee, S. Kwon, J. H. Kim, K. and G. Kim, "Development of an Automatic Pill Image Data Generation System," Healthcare Informatics Research, vol. 29, no. 1, pp. 84-88, 2023.

[6] N. Tuli, and A. Mantri, "Evaluating usability of mobile-based augmented reality learning environments for early childhood," International Journal of Human–Computer Interaction, vol. 37, no. 9, pp. 815-827, 2021.

[7] H. Zhu, Y. Li, R. Li, J. Li, Z. You, and H. Song, "SEDMDroid: An enhanced stacking ensemble framework for Android malware detection," IEEE Transactions on Network Science and Engineering, vol. 8, no. 2, pp. 984-994, 2020.

[8] Z. Wang, C. Gao, Y. Chen, and C. Lu, "A real-time object detection method for electronic screen GUI test systems," The Journal of Supercomputing, vol. 2024, no. Jul, pp. 1-33, 2024.

[9] Z. He, S. Sunkara, X. Zang, Y. Xu, L. Liu, N. Wichers, and J. Chen, "Actionbert: Leveraging user actions for semantic understanding of user interfaces," Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, no. 7, pp. 5931-5938, 2021.

[10] N. Alajarmeh, "The extent of mobile accessibility coverage in WCAG 2.1: sufficiency of success criteria and appropriateness of relevant conformance levels pertaining to accessibility problems encountered by users who are visually impaired," Universal Access in the Information Society, vol. 21, no. 2, pp. 507-532, 2022.

[11] L. Nie, K. S. Said, L. Ma, Y. Zheng, and Y. Zhao, "A systematic map study for graphical user interface testing on mobile apps," IET Software, vol. 17, no. 3, pp. 249-267, 2023.

[12] M. Soui, M. Chouchane, M. W. Mkaouer, M. Kessentini, and K. Ghedura, "Assessing the quality of mobile graphical user interfaces using multi-objective optimizatio," Soft Computing, vol. 24, no. 10, pp. 7685-7714, 2020.

[13] M. Pan, T. Xu, Y. Pei, Z. Li, T. Zhang, and X. Li, "GUI-guided test script repair for mobile apps," IEEE Transactions on Software Engineering, vol. 48, no. 3, pp. 910-929, 2020.

[14] M. Hort, M. Kechagia, F. Sarro, and M. Harman, "A survey of performance optimization for mobile applications," IEEE Transactions on Software Engineering, vol. 48, no. 8, pp. 2879-2904, 2021.

[15] T. Su, L. Fan, S. Chen, Y. Liu, L. Xu, G. Pu, and Z. Su, "Why my app crashes? Understanding and benchmarking framework-specific exceptions of Android apps," IEEE Transactions on Software Engineering, vol. 48, no. 4, pp. 1115-1137, 2020.

[16] ZT. Zhang, Y. Liu, J. Gao, and L. P. Gao, "Deep Learning-Based Mobile Application Isomorphic GUI Identification for Automated Robotic Testing," IEEE Software, vol. 37, no. 4, pp. 67-74, 2020.

[17] Y. Ege, H. Citak, S. Bicakci, M. Coramik, and Y. Ege, "Buried Magnetic Material Detection System: An SVM Algorithm Application," IEEE Transactions on Magnetics, vol. 57, no. 7, pp. 1-9, 2021.

[18] J. Cheng, D. Tan, T. Zhang, A. Wei, and J. Chen, "YOLOv5-MGC: GUI Element Identification for Mobile Applications Based on Improved YOLOv5," Mobile Information Systems, vol. 2022, no. Aug 8, pp. 1-14, 2022.

[19] M. D. Altinbas and T. Serif, "GUI Element Detection from Mobile UI Images Using YOLOv5," in International Conference on Mobile Web and Intelligent Information Systems, vol. 15, no. 6, pp. 32-45, 2022.

[20] C. Chen, S. Feng, Xing Z., L. Liu, S. Zhao, and J. Wang, "Gallery DC: Design search and knowledge discovery through auto-created GUI component gallery," Proceedings of the ACM on Human-Computer Interaction, vol. 3, no. CSCW, pp. 1-22, Nov. 2019.

[21] U. Sirisha, S. P. Praveen, P. N. Srinivasu, P. Barsocchi, and A. K. Bhoi, "Statistical analysis of design aspects of various YOLO-based deep learning models for object detection," International Journal of Computational Intelligence Systems, vol. 16, no. 1, p. 126, Aug. 2023.

[22] M. Adam, M. Wessel, and A. Benlian, "AI-based chatbots in customer service and their effects on user compliance," Electronic Markets, vol. 31, no. 2, pp. 427-445, 2021.

[23] L. C. Zou, "Research on the interaction design of drawing game apps for preschool children". Applied Computer Letters, vol. 6, no. 1, 2022.

[24] C. L. Lin and K. C. Wu, "Development of revised ResNet-50 for diabetic retinopathy detection," BMC Bioinformatics, vol. 24, no. 1, p. 157, Apr. 2023.

[25] S. Seol, J. Ahn, H. Lee, Y. Kim, and J. Chung, "SSP based underwater CIR estimation with S-BiFPN," ICT Express, vol. 8, no. 1, pp. 44-49, 2022.

[26] F. Hua, "Improved Surface Defect Detection of YOLOV5 Aluminum Profiles based on CBAM and BiFPN," International Core Journal of Engineering, vol. 8, no. 12, pp. 264-274, 2022.

[27] J. Jia, "Design of fruit fly optimization algorithm based on Gaussian distribution and its application to image processing," Systems and Soft Computing, vol. 6, pp. 200090, 2024.

[28] X. Fu, Y. Peng, Y. Liu, Y. Lin, G. Gui, H. Gacanin, and F. Adachi, "Semi-supervised specific emitter identification method using metric-adversarial training," IEEE Internet of Things Journal, vol. 10, no. 12, pp. 10778-10789, Jan. 2023.

[29] X. Fu, S. Shi, Y. Wang, Y. Lin, G. Gui, O. A. Dobre, and S. Mao, "Semi-supervised specific emitter identification via dual consistency regularization," IEEE Internet of Things Journal, vol. 10, no. 21, pp. 19257-19269, May 2023.

[30] A. Gui, S. Gao, P. Zheng, Z. Feng, P. Liu, F. Ye, S. Wang, J. Xue, J. Ni, and J. Yin, "Dynamic changes in non-volatile components during steamed green tea manufacturing based on widely targeted metabolomic analysis," Foods, vol. 12, no. 7, pp. 1551, Apr. 2023.

[31] W. Huang, F. Wen, S. Ruan, P. Gu, S. Gu, S. Song, J. Zhou, Y. Li, J. Liu, and P. Shu, "Integrating HPLC-Q-TOF-MS/MS, network pharmacology and experimental validation to decipher the chemical substances and mechanism of modified Gui-shao-liu-jun-zi decoction against gastric cancer," Journal of Traditional and Complementary Medicine, vol. 13, no. 3, pp. 245-262, May 2023.

[32] J. Yang, L. Huang, K. Tong, Q. Tang, H. Li, H. Cai, and J. Yin, "A review on damage monitoring and identification methods for arch bridges," Buildings, vol. 13, no. 8, pp. 1975, Aug. 2023.

[33] B. Divya, and K. Ganesan, "The Fuzzy Sumudu Decomposition Method for Solving Differential Equations in an Imprecise Context," IAENG International Journal of Applied Mathematics, vol. 54, no. 12, pp2523-2528, 2024.

[34] K. Yuan, C. Zhao, Y. Chen, L. Shen, Q. Tang, and C. Jia, "Mapping the Knowledge Structure and Research Evolution of Deep Learning," IAENG International Journal of Computer Science, vol. 51, no. 10, pp1404-1412, 2024.

[35] M. Jimenez-Martinez, S. G. Torres-Cedillo, and J. Cortes-Perez, "Fatigue Analysis of Printed PLA using Neural Networks," IAENG International Journal of Computer Science, vol. 51, no. 7, pp709-715, 2024.