UI Components Recognition System Based On Image Understanding

Xiaolei Sun*
Wuhu Institute of Technology
Wuhu, China
75sun@whit.edu.cn

Tongyu Li
Nanjing Mooctest Inc.
Nanjing, China
lity980@163.com

Jianfeng Xu
Nanjing Mooctest Inc.
Nanjing, China
xujianfeng@mooctest.com

***Abstract*—Before the release of mobile application products, a lot of repeated testing is often required. In the process of mobile application testing, the core problem is to locate the UI components on the mobile application screenshots. There are many methods to automatically identify UI components, but in some cases, such as crowdsourcing testing, it is difficult to use automatic methods to identify UI components. In view of this, the APP UI components recognition system based on image understanding provides new solutions and methods for application scenarios that are difficult to automatically locate components. We investigate Android UI component information, use image understanding analysis to extract component images on screenshot, design and implement a convolutional neural networks,and then use trained CNN to classify these images. The classification accuracy is up to 96.97%. In the end, we get the component information contained in screenshot.**

***Index Terms*—UI Component, Image Processing, CNN**

I. Introduction

With the rapid growth of economy and the rapid development of mobile Internet, mobile applications have penetrated into all aspects of people's lives, and the number of them is growing explosively. Therefore, higher requirements are placed on the quality of mobile application products. Before the release of mobile application products, a large number of repeated tests are often needed to ensure the quality of products. In the process of mobile application testing, the core problem is to locate the UI components on the mobile application page. For a button or input box on mobile app, if we want to click or input it, the premise is to find the component object first. Therefore, it is particularly important to identify the component object on the current page.

Graphical user interface (GUI) testing is the focus of regression testing, and its efficiency will directly affect the cost of testing. In the GUI testing process [1], automated testing is widely used. According to the different degree of automation, the current GUI automation test methods can be divided into two categories, manually writing test scripts and record and playback methods [2]. The former requires testers to write test scripts manually, which results in higher test costs. The latter records the interaction between the user and the application under test and converts it into script by recording and playing back, and then executes the content of script playing back interaction [3]. Among them, the most important is the recognition method, that is to automatically identify the corresponding visual components of interactive operations. Common component recognition methods include coordinate-based component recognition, source code-based component recognition and component tree traversal based component recognition.

For crowdsourcing testing, the positioning of components cannot be automated. Crowdsourcing is a distributed problem-solving model that integrates unknown people on the Internet to complete tasks that are difficult for computers to complete [4], and crowdsourcing testing refers to all crowdsourcing activities related to software testing [5]. In the crowdsourcing test report, screenshots and descriptive text are usually included, as well as evaluations by staff that the software behaves correctly or incorrectly [6]. During the crowdsourcing test, only screenshots can be obtained, so it is difficult to locate UI components based on the frame. In addition, UI component identification is the most basic part of the crowdsourcing testing. Only when the UI components are recognized, can we do such tasks as bug screenshot processing, scene understanding, and test report classification and clustering during the testing process.

In view of the important value of UI components recognition, we investigate Android UI component information for the location of component elements, train deep learning model, use image understanding analysis technology to extract UI component images on screenshot, and use trained neural networks to classify these images. Finally, we get the UI component information contained in the screenshot.

In terms of image processing, based on the factors of the screenshot itself, we performed a series of processing such as image gray gradient, image binarization, denoising, and image segmentation on the screenshot. After dividing the image into blocks, we detect the area containing UI components in the block, and then extract and label the component in the area. After going through this part, we can get a screenshot of the UI components marked. In terms of component recognition, we design a convolutional neural network, and use the trained network for component recognition after image processing, with the accuracy of 96.97%.

In summary, this paper mainly makes the following contributions:

*Corresponding author

978-1-7281-8915-4/20/$31.00 ©2020 IEEE
DOI 10.1109/QRS-C51114.2020.00022

- We mainly design and implement an UI components recognition system based on image understanding.After taking the screenshot, we perform gray gradient, binarization, image segmentation and other operations on the screenshot, detect the area containing UI components, and then extract and annotate the components in the area.We design and implement a CNN, and use the trained network to recognize the component after image processing.with an accuracy rate of 96.97%.
- For the application scenarios where it is difficult to automatically locate UI components, such as in the process of crowdsourcing test or screenshot contain component drawn by the user, some complex composite component, etc., we provide new solutions and methods.

II. Background

In this part, we introduce the relevant knowledge of Android UI components, image processing technology, and related concepts of Convolutional Neural Network.

A. Android UI components

Android provides a large number of UI components, which can be used properly to easily write a pretty good interface.In Android, View is the basis of all the content displayed on the screen. Buttons, toolbars, input boxes, etc. are all Views. As the most basic component of UI, view is responsible for drawing UI components and monitoring interface actions. It can be considered as a button, text field and other interface elements or other View containers.We consulted the Android API, made a comprehensive summary of the Android UI components, and selected 14 commonly used Android UI components as the classification criteria. They are: Button, CheckBox CHeckTextView, EditText, ImageButton, ImageView, ProgressBarHorizontal, ProgressBarVertical, RadioButton, RatingBar, Switch,SeekBar, Spinner, TextView.

B. Image Processing

Researchers have done some work on image understanding testing technology. Tuan Anh Nguyen and others proposed REMAUI, which implements the function of inputting a screenshot of the mobile terminal and generating corresponding source code and resource files. On a given input picture, REMAUI identifies user interface elements such as images, text, containers, and lists through computer vision and Optical Character Recognition (OCR) technology [7]. Kevin Moran and others proposed ReDraw, which realizes GUI precise prototyping through three tasks: detection, classification and assembly. First, the logical components of GUI are detected from model artifacts using computer vision technology or model metadata. Then, GUI components are classified into specific types accurately by using software knowledge base mining, automatic dynamic analysis and deep convolution neural network. Finally, a data-driven K-nearest-neighbors algorithm generates a suitable hierarchical GUI structure from which prototype applications can be automatically assembled [8].Chunyang et al. introduced a neural network machine translator [9], which combines the latest advances in computer vision and machine translation to convert UI images into GUI skeletons.

But none of the above work regards the UI components in the recognition image as a single work, and we separate it separately. Our UI component recognition technology refers to the method in the above reference and proposes our own method.

C. Convolutional Neural Network (CNN)

In recent years, with the rapid development of deep learning technology, Convolutional Neural Networks(CNN) are significantly superior to traditional methods in accuracy, and have become the latest research hotspot, such as AlexNet, VGGNet, GoogLeNet, ResNet and other neural network models.AlexNet [10] deepens the structure of the network on the basis of LeNet. For the first time, it has successfully applied such tricks as ReLU, Dropout, and LRN in CNN to learn richer and higher dimensional image features. The ResNet [11] model is an efficient neural network proposed by Microsoft, and a 152-layer neural network was successfully trained by using ResNet Unit. Its core is to solve the side effects (degradation problem) caused by increasing the depth.In this way, the network performance can be improved by simply increasing the depth of the network. To a certain extent, the problem of gradient disappearance and gradient explosion in the deep neural network is solved.

Due to the rapid development of CNNs [10] [12] [13], there has been a huge breakthrough in large-scale image classification tasks.After UI components are identified, they need to be classified, such as button, textview, progress bar, etc. This is actually an image classification task. Relying on the rapid development of deep Convolution Neural Network, we use CNN to classify the image of UI components into specific types.

III. Approach and Implementation

We first use image processing technology to extract of the suspected UI components from the screenshot and mark it in the screenshot. In order to classify the detected UI components into appropriate types, we design and implement a convolutional neural network. We train CNN by using the data set in the literature [8] [14] after random sampling. Then, use the trained CNN to classify the detected UI components after image processing, and mark the predicted component categories on the screenshot.The whole process of the system is shown in Fig. 1.

A. Image Processing and detect UI components

After obtaining the screenshot, we need to process the image to get the suspected UI component image. The image obtained by the screen capture tool is a color image, and we need to convert the color image into gray image. When mean filter is used to reduce image noise, it will bring side effects of image blur. Therefore, we take the gradient of the gray image to make the gray change of the contour edge more obvious.

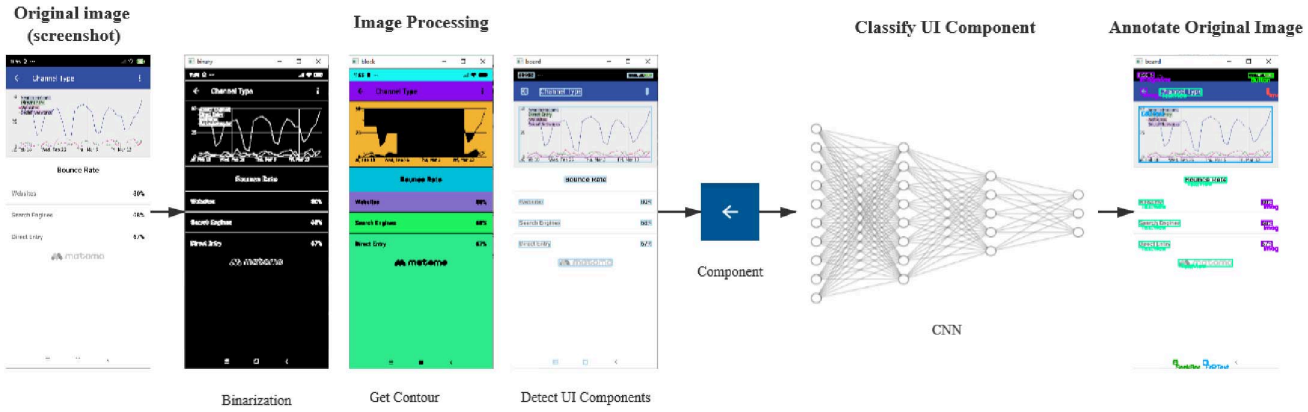

Fig. 1: System Process

After the gray gradientation of the screenshot, the obtained gray image needs to be binarized. We use the cv2.threshold () function provided in OpenCV to get the binary image from the gray image. For binarization, the purpose is to classify the target user background and prepare for the identification of subsequent UI components. In order to remove the noise that may exist in the binary image, we use cv2.morphologyex() function provided in OpenCV to close the image,first perform the dilation operation, and then perform the erosion operation to get a smoother binary image.

We segment the binary image, cut the image into blocks, detect whether the block contains UI components, and mark the components. We use the Flood Fill algorithm to mark the location of layout blocks in the image.The FloodFill algorithm is widely used, among which the ”reverse color” of drawing is prominent. The principle is to start with a point and fill the nearby pixels with a new color until all pixels in the closed area are filled with the new color. In the image, it is likely that the layout blocks are nested or overlapped. Therefore, after detecting the layout block, we also need to calculate the level of the block to label it. For the calculation of the block level, we calculate the level relationship by comparing the relative sizes of two blocks. For the hierarchical relationship between blocks, we divide into the following four cases:

- Case 1: Block A is in Block B.
- Case 2: Block A and Block B are not intersected.
- Case 3: Block B is in Block A.
- Case 4: Block A and Block B are identical or intersected.

We detect whether the divided blocks contain components. First, we check whether the block itself is a UI component based on its relative size. If it meets our standards, it is considered a UI component. Then, the binary image is processed and cut according to the relative size of the block to obtain the area containing the UI component in the block. Note that in this step of processing, due to the nesting relationship between the blocks, it is necessary to delete the blocks containing sub blocks before cutting. Finally, the components in region are extracted. The binary image is used as input to calculate the connection areas, obtain their boundaries, and check whether these areas are rectangular. After segmenting the image into blocks, there are still some non block components in the image, and we still need to process them. The detection method is the same as above.

We optimize the identified components and mark the components in the original image at the same time, which will lay a foundation for the subsequent classification of UI components using neural networks.

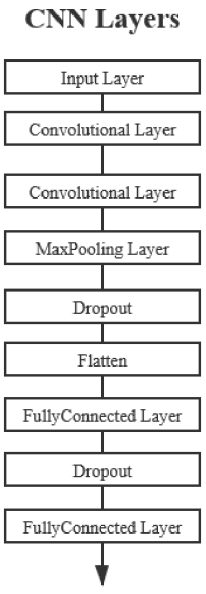

Fig. 2: Network Architecture

Fig. 3: Image after labeling UI components

Fig. 4: Image after labeling UI component type

B. *Identify and classify UI components*

After clipping out the suspected UI component images from the screenshot, the next step is to classify these images into specific types, such as button, textview, progress bar, etc.In order to achieve this goal, we have implemented a CNN, which can classify the target images of UI components into one of the 14 commonly used Android UI components.The network structure is shown in Fig. 2. Because what we need is to classify 14 common UI components, this simple network structure can meet our needs.

In order to verify the performance of our neural network, a deep learning framework based on Tensorflow was built on the GTX 1080 Ti GPU server. The operating environment is Ubuntu 16.04, the programming language is Python 3.6.8, and the main function libraries used include Open CV , Keras, etc.

The experimental data of our CNN training comes from the ReDraw data set and the data set in [14]. ReDraw is a data set of CNN and KNN machine learning techniques used to train and evaluate reproducible papers in [8]. In order to train our CNN, we randomly sampled 2.5K images from each class of the ReDraw data set and the data set in [14] to construct a small data set.Some examples in the data set are shown in Fig. 5. We divide the data into training set (70%), validation set (20%) and test set (10%). We record the accuracy on the validation set, and record the final accuracy on the test set. We adjusted the relevant parameters and structure of the neural network until the test accuracy reached the peak value.

In order to evaluate the accuracy of CNN, we tested the Recall rate and Precision rate of all classes on the test set:

$$Recall = \frac{TP}{TP + FN}$$

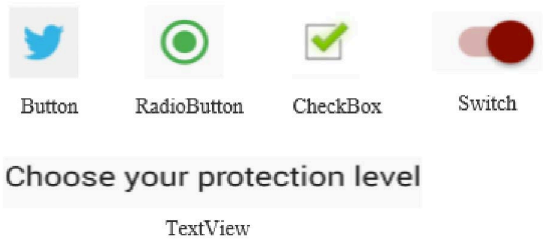

Fig. 5: Component Image in Data Set

$$Precision = \frac{TP}{TP + FP}$$

where TP means that the true category of the sample is positive, and the result of model prediction is also positive; FN means that the true category of the sample is positive, but the model predicts it as a negative example; FP means that the true category of the sample is negative, but the model predicts it as a positive example.In order to illustrate the classification ability of CNN, we give a confusion matrix with precision in Table 1.

After training CNN, we use the trained neural network to classify the UI component images processed in 3.1.The result of classification is then annotated into the original image.

IV. Evaluation

Our system first needs to detect the UI components in the screenshot. The main goal of this process is to use image processing techniques to extract the suspected UI components

TABLE I: Confusion Matrix

	Bt	CB	CTV	ET	IB	IV	PBH	PBV	RB	RtB	St	SB	Sp	TV
Bt	82.30%	0.56%	0.75%	1.88%	1.51%	6.03%	0.00%	0.00%	0.75%	0.00%	0.19%	0.00%	0.00%	6.03%
CB	0.19%	96.35%	0.00%	0.77%	0.19%	1.15%	0.00%	0.00%	0.19%	0.00%	0.00%	0.00%	0.00%	1.15%
CTV	1.15%	0.00%	93.27%	1.15%	0.00%	0.38%	0.00%	0.00%	0.38%	0.00%	0.00%	0.00%	0.00%	3.65%
ET	6.91%	0.58%	1.92%	74.66%	1.15%	3.07%	0.58%	0.00%	0.58%	0.19%	0.38%	0.00%	0.00%	9.98%
IB	5.19%	0.19%	0.00%	0.38%	76.73%	14.62%	0.19%	0.00%	0.58%	0.00%	0.00%	0.19%	0.00%	1.92%
IV	3.09%	1.82%	0.36%	1.09%	8.00%	74.36%	0.73%	0.00%	2.55%	0.00%	1.27%	0.00%	0.00%	6.73%
PBH	0.19%	0.00%	0.00%	0.00%	0.58%	0.19%	98.65%	0.00%	0.00%	0.00%	0.19%	0.00%	0.00%	0.19%
PBV	0.18%	0.00%	0.00%	0.00%	0.18%	0.91%	4.19%	94.35%	0.00%	0.00%	0.00%	0.00%	0.00%	0.18%
RB	1.75%	0.39%	0.19%	0.78%	0.19%	0.00%	0.39%	0.00%	94.17%	0.00%	0.00%	0.00%	0.00%	2.14%
RtB	0.00%	0.00%	0.00%	0.19%	0.00%	0.57%	0.00%	0.00%	0.00%	99.05%	0.00%	0.00%	0.00%	0.19%
St	0.19%	0.00%	0.19%	0.19%	0.19%	0.00%	0.00%	0.00%	0.00%	0.00%	98.85%	0.19%	0.00%	0.19%
SB	0.00%	0.19%	0.19%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	99.23%	0.00%	0.38%
Sp	0.19%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	99.81%	0.00%
TV	6.39%	1.32%	3.57%	3.76%	0.75%	3.38%	0.19%	0.19%	1.32%	0.00%	0.19%	0.00%	0.00%	78.95%

[a] The abbreviation for the line title of the UI component type: Button(Bt), CheckBox(CB), CheckedTextView(CT), EditText(ET), ImageButton(IB), ImageView(IV), ProgressBar(PB), ProgressBarHorizontal(PBH), ProgressBarVertical(PBV), RadioButton(RB), RatingBar(RtB), SeekBar(SB), Switch(Sw), Spinner(Sp), TextView(TV).

image from the screenshot. We process the screenshot with gray gradientation, binarization, denoising, image segmentation and other processing. We get the bounding box of the suspected component and label it on the original image. The detected UI component image is shown in Fig. 3.As we can see, our method detects almost all UI components in the screenshot.

After detecting the UI component bounding box contained in the screenshot, we input these images into the trained CNN. The network will predict the category to which it belongs and select the category with the highest probability as its category. We label the predicted component categories on the original image, as shown in Fig. 4. Table 1 shows the confusion matrix with precision. Each row of the confusion matrix represents the actual category, and each column represents the predicted result, and the diagonal corresponds to the correct classification. The accuracy of our CNN on the training set reached 96.97%. And the precision rate of CNN is 86.4%, and the recall rate is 86.4%.

Combined with Fig. 4 and Table 1, we can see that our CNN has some deviations in the predictions of certain categories. This is because of the multiple styles of component types, and the component images of different apps are different.Although CNN may occasionally misclassify components, the confusion matrix indicates that these misclassifications tend to be similar classes. For example, imagebutton is mainly misclassified as ImageView. In the process of practical application, experienced developers can manually correct the components that predict errors.

V. Related Work

Mobile applications are widely used in different tasks in people's daily lives, so higher requirements are put forward on the testing of mobile applications. At present, there are various tools to automatically identify mobile application page elements, such as UI Automator Viewer and Appium, etc. However, for test scenarios that are difficult to use automatic identification tools, such as the processing of screenshots in crowdsourcing testing, these tools cannot be useful. Our work is aimed at this. In this section, we introduce some research related to our work.

A. UI component data set

To train our convolutional neural network, we need a data set containing a large number of UI component images. Recently, more and more related datasets have been open sourced.

Rico is a mobile application design repository that was created to support five types of data-driven applications: design search, UI layout generation, UI code generation, UI code generation, user interaction modeling, and user perception prediction [15]. It contains more than 66k unique UI screens and 3M UI elements with visual, text, structure and interactive design attributes.In reference [14], they uses app explorer to automatically browse different screens in the application by clicking, editing and other actions, taking screenshots of the application GUI, dumping the run-time front-end code. After deleting all duplicates by checking the screenshots, there are 278234 screenshots containing ImageView and ImageButton from 10408 applications in the collected data.

In addition, reference [8] proposes a method for automatically creating labeled training data, which consists of images from specific GUI components of full screenshots and labels corresponding to their category using fully automated dynamic program analysis. After filtering the data set, they obtained a data set containing 14,382 unique screens with 431,747 unique components from 6,538 applications.

The data sets in our work refer to the data sets in [8] and [14], and are filtered and added to form the data sets we use.

B. Identify user interface elements

In recent years, it is a research hotspot to identify the elements in user interface, and some researchers have made relevant work in this field.

REMAUI identifies user interface elements such as images, text, containers, and lists through computer vision and OCR technology on a given input bitmap[7]. In terms of identifying page elements, REMAUI first applies off the shelf OCR detection on a given input image. Because there will be recognition errors in optical character recognition, it will post process OCR results. Later in computer vision, it uses the Canny algorithm to detect the edges of each image element, merge similar elements with each other, and merge with the surrounding noise, and close the almost closed contour, REMAUI expands its detected edges. Finally, REMAUI merges the results of OCR and computer vision to heuristically combine the best results.

Kevin Moran et al. Proposed Redraw, a system which implements accurate prototype of GUI through three tasks, detection, classification and assembly. It converts the raphical user interface (GUI) model into code. In the first step of detecting the logic components of GUI, Redraw uses computer vision technology to automatically infer the boundary box of components from the static image. It reproduces the CV method described in [7], and makes innovation on this basis. Then, it uses software repository mining, automatic dynamic analysis and deep convolution neural network to classify GUI components accurately.

In view of the difficulty for visually impaired users to obtain information on the screen intuitively, Jieshan Chen et al. developed a model called LabelDroid, which based on deep learning. It automatically predicts the label of image-based buttons [14]. CNN is used to extract image features, and transform model is used as encoder and decoder, then the extracted informative features are encoded into tensor by encoder module. Based on the encoding information, the decoder module generates a word sequence output based on the tensor and the previous output.

Xusheng Xiao et al. proposed an application analysis framework, ICONINTENT. Through statically analyzing the UI layout files and code of the application, automatically associating UI widgets and icons, and then using computer vision technology to classify the relevant icons into eight sensitive data categories [16].

The above work is not completely similar to our work. We process the screenshot, extract the component images, and then use the trained neural network to classify these images and mark the results on the original images.

C. Image Classification

After identifying the components in the screenshot, we need to classify them into their corresponding category, which is actually an image classification task. The traditional method of image classification is feature description and detection. This kind of traditional method may be effective for some simple image classification, but because the actual situation is very complicated, the traditional classification method is overwhelmed. Relying on the rapid development of neural networks, the use of machine learning methods to deal with image classification problems has gradually become the mainstream method.

Alex Krizhevsky proposed a CNN model called AlexNet, which was won the ILSVRC2012 championship, and its effect greatly exceeds the traditional method. This model is called AlexNet [10]. This is also the first time that deep learning is used in large-scale image classification. After AlexNet, a series of CNN models emerged, such as VGG [17], GoogleNet [18], and ResNet[11], which constantly updated their performance on Imagenet. As the model becomes more and more deep and sophisticated in structural design, the error rate of Top-5 is getting lower and lower.

VI. Conclusion and Future Work

Aiming at the positioning of UI components, we investigates Android UI component information, trains deep learning models, uses image understanding analysis to extract key component images on screenshot. After getting the component image, we use the trained CNN to classify these images, Finally we get the component information contained in the screenshot and mark it on the origial image. This system helps testers locate UI components without using automated tools to identify components and improves test efficiency.

We hope to cover as many UI components as possible and some user-defined components in the follow-up research process. In the complex screenshot, the accuracy of component recognition in our system is not very ideal. To further strengthen the image processing part, and the segmentation of screenshot needs to be more accurate, so as to improve the accuracy of control recognition.

Acknowledgements

This work is partially supported by the National key R&D program of China (2018YFB1403400) and the Key Project of Natural Science Research in Anhui Higher Education Institutions (KJ2019ZD67).

References

[1] Silva D B , Endo A T, Eler M M and V. Durelli, “An analysis of automated tests for mobile Android applications,”2016 XLII Latin American Computing Conference (CLEI), pp.1-9, 2016.

[2] Jin Hou, Naijie Gu, Shiju Ding, Yunkai Du, “UI automating test method for cross-device based on widget path,” Computer Systems and Applications, pp.240–247, 2018.

[3] Shiju Ding, Naijie Gu, Zhangjin Huang and Jin Hou, “APP control recognition algorithm based on text recognition and page layout,” Computer Engineering, pp.89-95, 2019.

[4] Jianhong Feng, Guoliang Li and Jianhua Feng, “ A survey on crowdsourcing,” Chinese Journal of Computers, pp.1713-1726, 2015.

[5] Xiaofang Zhang, Yang Feng, Di Liu, Zhenyu Chen and Baowen Xu, “Research progress of crowdsourced software testing. Journal of Software, pp.69-88, 2018.

[6] Yang Feng, James J, Zhenyu Chen and Chunrong Fang, “Multi-objective test report prioritization using image understanding ,” 2016 31st IEEE/ACM International Conference on Automated Software Engineering (ASE), pp.202-213, 2016.

[7] Nguyen T A and Csallner C, "Reverse engineering mobile application user interfaces with REMAUI," 2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE), pp. 248-259, 2015.
[8] Moran K, Bernal-Cardenas C, Curcio M, Bonett R, and Poshyvanyk D, "Machine learning-based prototyping of graphical user interfaces for mobile apps," IEEE Transactions on Software Engineering, pp.196-221, 2018.
[9] Chunyang Chen, Ting Su, Guozhu Meng, Zhencang Xing and Yang Liu, "From UI design image to GUI skeleton: a neural machine translator to bootstrap mobile GUI implementation," 2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE), pp.665–676, 2018.
[10] Krizhevsky A, Sutskever I, Hinton G, "ImageNet classification with Deep Convolutional Neural Networks," Advances in neural information processing systems, 2012.
[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun, "Deep residual learning for image recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp.770-778, 2016.
[12] Lecun Y, Bottou L, Bengio Y, and Haffner P, "Gradient-based learning applied to document recognition," Proceedings of the IEEE, pp.2278–2324, 1998.
[13] Simonyan K and Zisserman A, "Very deep convolutional networks for large-scale image recognition," Computer ence, 2014., 2014.
[14] Jieshan Chen, Chunyang Chen, Zhenchang Xing, et al,"Unblind Your Apps: predicting natural-language labels for mobile GUI components by deep learning," arXiv:2003.00380.
[15] Deka B, Huang Z, Franzen C, et al. "Rico: A mobile app dataset for building data-driven design applications,"2017 30th Annual Symposium on User Interface Software and Technology, 2017.
[16] Xusheng Xiao, Xiaoyin Wang, Zhihao Cao, Hanlin Wang and Peng Gao, "IconIntent: automatic identifification of sensitive UI widgets based on icon classifification for Android Apps," 2019 41st IEEE/ACM International Conference on Software Engineering (ICSE), 2019.
[17] Simonyan K and Zisserman A, "Very Deep Convolutional Networks for large-scale image recognition," arXiv 1409.1556.
[18] Szegedy C, Liu W, Jia Y, et al. "Going deeper with convolutions," 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp.1-9, 2015.
[19] Shengcheng Yu, Chunrong Fang, Yang Feng, Wenyuan Zhao, Zhenyu Chen, "LIRAT:layout and image recognition driving automated moblie testing of cross-platform," 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE), 2019.