

Deep Learning-Based Mobile Application Isomorphic GUI Identification for Automated Robotic Testing

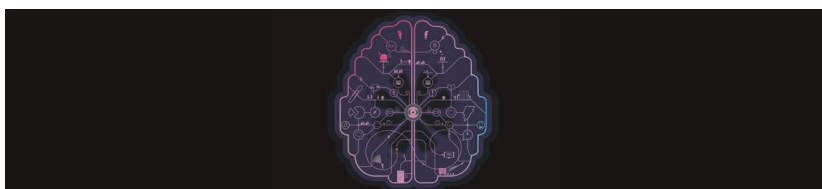
Tao Zhang and Ying Liu, Northwestern Polytechnical University

Jerry Gao, San José State University

Li Peng Gao, Northwestern Polytechnical University

Jing Cheng, Xi'an Technological University

// Fully black-box robotic testing is needed given the popularity of mobile applications. A critical constraining issue for generating graphical user interface (GUI) models is identifying isomorphic GUIs. We present a deep learning-based end-to-end trainable model to determine the similarity between GUIs and identify isomorphic GUIs. //



Digital Object Identifier 10.1109/MS.2020.2987044
Date of current version: 18 June 2020

MOBILE APPLICATIONS (APPS)

increasingly play essential roles in people's lives. Adequate and rigorous software testing is essential to ensure the security and robustness of apps. However, the releasing cycle of mobile apps is extraordinarily high, and inefficient manual labor and limited devices cannot meet the requirements. Therefore, automated testing should be efficiently deployed to decrease cost and ensure rapid feedback.

In recent years, researchers have shown an increased interest in test automation, which includes the script-based method and record and replay testing method. The script-based method relies on human test-script design at considerable cost. The record and replay testing approach relies on the technique of recording and playback of test scenarios. However, this approach is typically sensitive to graphical user interface (GUI) layout change and code change.¹ Hence, recent techniques have great limitations in real applications.

Robot testing is a new method that can reduce the labor cost during test case execution.² However, recent advanced robotic testing techniques cannot handle visual GUI testing when source codes are unavailable. Thus, our long-term goal is to develop a deep learning vision-based robot for fully black-box automatic testing. The robot emulates real users' operations and interacts only at the device level. A few studies^{3,4} have focused on deep learning-based testing, and did not solve isomorphic GUI identification.

This study focuses on our recent significant progress toward reaching our long-term goal. We solve the key issue of how to identify isomorphic GUIs. The contributions are as follows:

1. Our work integrates isomorphic GUI identification into an

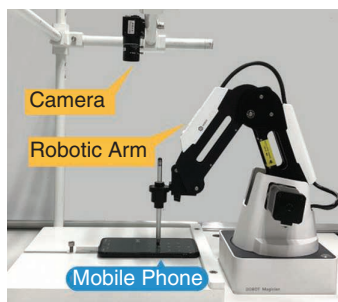


FIGURE 1. Our vision-based robot.

end-to-end trainable model, in which the inputs are the GUIs and the outputs are the identification results. Identification can be realized without using source codes.

2. Identifying isomorphic GUIs can avoid redundant test cases and prevent explosive growth in the

number of test cases by merging redundant isomorphic nodes in GUI models.

Why Perform Isomorphic GUI Identification?

Our long-term goal is to develop a robot that can analyze and report bugs for fully automated mobile application black-box testing (Figure 1). The camera captures GUI information and monitors test execution, and the robotic arm interacts with mobile devices as the test performer.

An essential step is generating a GUI model for model-based testing. The robot interacts with mobile apps at the device level. The interaction is saved both as a GUI model and as a test case. However, an important constraining issue is the determination of the isomorphic GUIs before generating GUI models.

What are isomorphic GUIs? Appearance (text, image, color, and size) differs between GUIs, but the function, structure, and internal logical relationships between the interfaces are the same. An example of isomorphic GUIs from the Google Play app⁵ is shown in Figure 2. Figure 2(d) shows the interface of the book recommendation list. Figure 2(a)–(c) shows the detailed interfaces of three books in the first row. Figure 2(a)–(c) differs only in terms of text and images; they are layout files loaded by the same activity at the Android developer level.

A finite state machine (FSM) is used to generate the GUI model.⁶ The generated model consists of nodes representing UI states and edges representing interactions. It considers the jump events of the four pages in Figure 2 but disregards the other buttons and the internal parts. The model is shown in Figure 3(a). To simplify the GUI model, S_{2a} , S_{2b} , and S_{2c} can

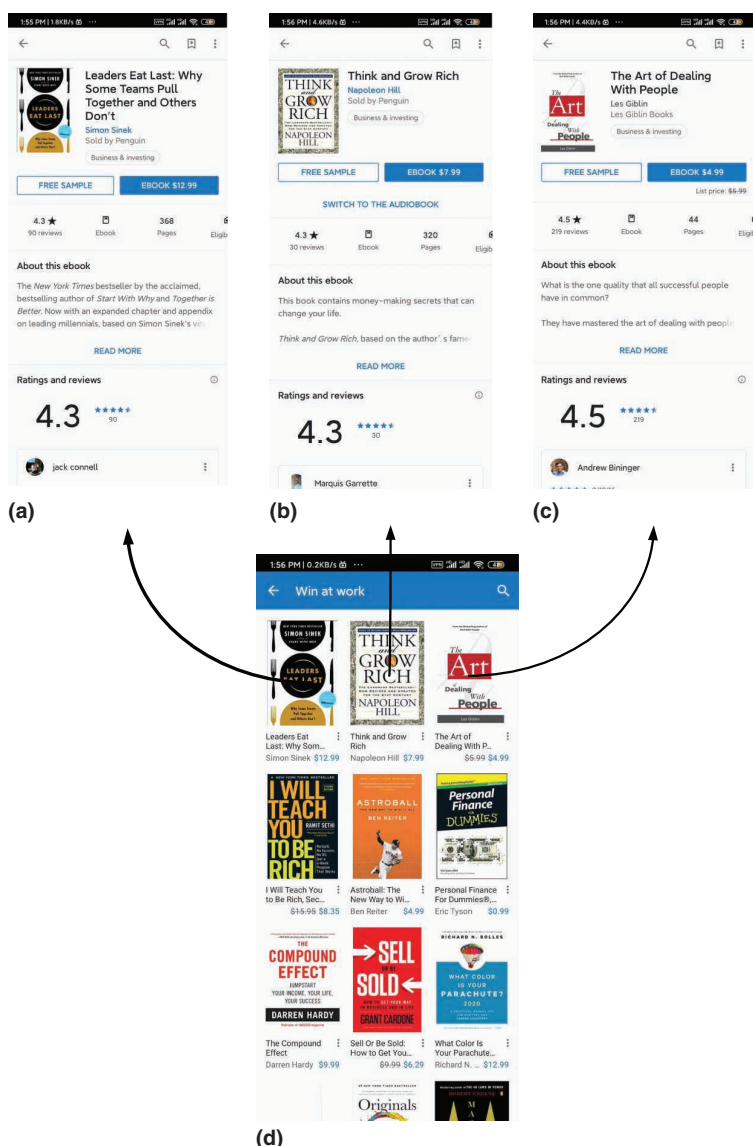


FIGURE 2. Examples of isomorphic GUIs.⁵ (a)–(c) are detailed interfaces of three books in the first row. (d) is the interface of the book recommendation list.

be regarded as the same state, and the simplified GUI model is shown in Figure 3(b). Therefore, identifying isomorphic GUIs can avoid the state-space explosion problem by recognizing and merging redundant isomorphic nodes. At this point, the necessity of identifying isomorphic GUIs in a vision-based approach becomes prominent.

Although many webpage similarity ranking methods⁷ have been proposed recently, they cannot be applied to identify isomorphic GUIs directly. Fortunately, many advances in deep learning have been achieved in recent years, thereby providing support to our work.

Proposed Method

A vision-based isomorphic GUI identification approach is proposed. First, GUI element recognition is achieved by YOLOv3.⁸ Then, a GUI skeleton⁹ is built to exclude noise from the UI style characters. An autoencoder¹⁰ is used to extract the feature vectors from the reconstructed GUI skeleton. Finally, relative entropy is used in the identification of isomorphic GUIs. The details are as follows.

GUI Element Recognition

The first step is to recognize the GUI elements of mobile apps. Deep learning-based detection methods can be divided into one-stage detectors and two-stage detectors. The former frames the detection as a “coarse-to-fine” process, whereas the latter frames the detection as “complete in one step.”¹¹

Detection speed is important in our work. Thus, the one-stage detection method, YOLOv3, is employed for real-time GUI element recognition. YOLOv3 extracts graphical features through convolutional neural networks (CNNs)

and then returns the categories and location information of the detection target.

Figure 4 shows an example of GUI element recognition. The main element categories of the interface are the texts and the checkboxes. The elements of this interface can be recognized accurately both in terms of location and category.

Constructing the GUI Skeleton

The essence of identifying isomorphic GUIs is to compare their structures and functions instead of the appearance of the GUIs. The characteristic of GUI designation can introduce noise if the feature vectors extracted from the GUI screenshots are compared directly. Here, the pixel-based depiction of GUIs is translated into a GUI skeleton to disregard the influence of visual effect.

The GUI skeleton is constructed by replacing each category of the GUI elements with a geometrical shape. An example of transforming UI images into a GUI skeleton is shown in Figure 5, and the element categories (i.e., texts and checkboxes) of the input image are enhanced.

Extracting Feature Vectors by Using An Autoencoder

An autoencoder comprises an encoder and a decoder. The encoder part maps the input figure into code, whereas the decoder part transforms the code into a reconstruction of the original input. Our autoencoder model consists of four main types of components: convolutional layer, deconvolutional layer, pooling layer, and loss function.

1. *Convolutional layer*: The convolutional layer uses convolution

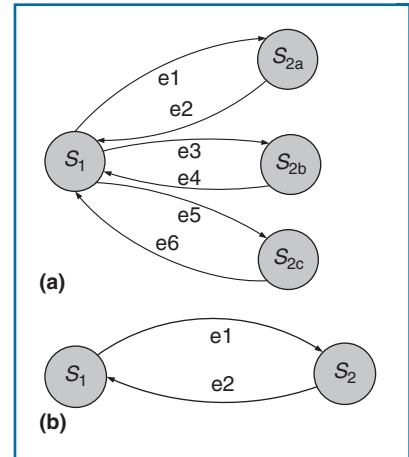


FIGURE 3. GUI models generated by FSM. (a) is the original GUI model, and (b) is the simplified version.

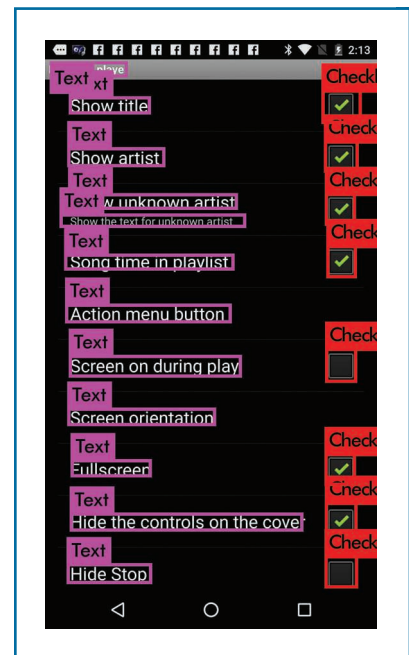


FIGURE 4. Example of GUI element recognition.

operations to drop and extract features of an input image. The extraction of features can be completed by convolution calculation. We use convolutional layers with rectified linear unit activation to extract feature

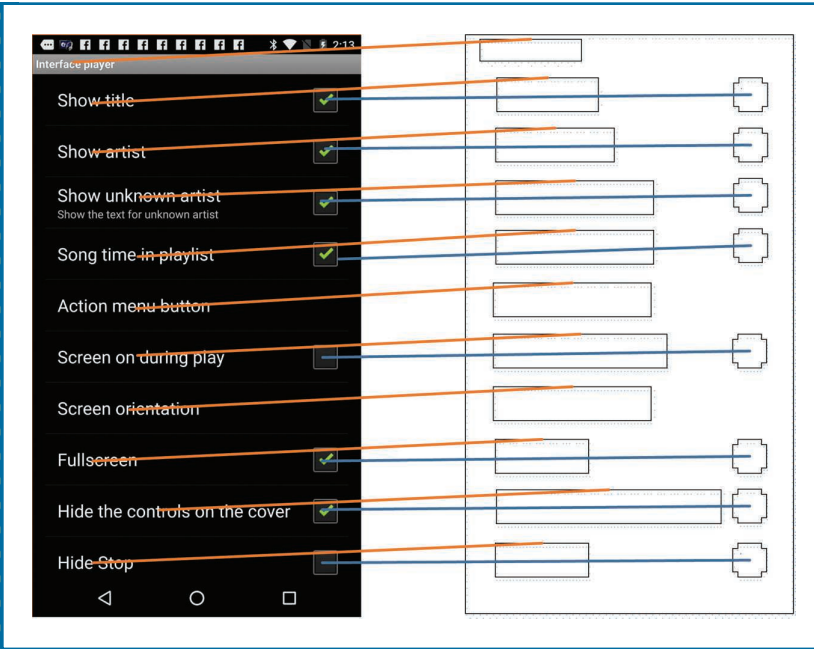


FIGURE 5. Example of constructing a GUI skeleton.

vectors from the input GUI skeleton. In addition, the padding is set to *valid* to prevent loss of information on the corners of the image.

2. **Pooling layer:** The pooling layer combines the characteristic information of the adjacent areas of the image. In this manner, image size can be compressed while valid information is retained. The pooling layer can not only reduce the complexity of calculation but also effectively alleviate the overfitting problem.
3. **Deconvolutional layer:** The calculation of the convolutional layer entails a downsampling process of extraction and compression of input data characteristics. The calculation of the deconvolutional layer involves an upsampling process of restoring input data to a feature map with the same or higher dimensions.

4. **Loss function:** The purpose of the training process is to minimize the difference between the original input and the reconstructed output. The difference is denoted as the loss function $L(x, y)$, where x is the input data and y is the reconstructed output. The loss function is defined as a cross-entropy loss function, as expressed by (1):

$$L(x, y) = \sum_{i=0}^{d_x} x_i \log y_i + (1 - x_i) \log (1 - y_i). \quad (1)$$

The loss function value is reduced constantly during the training process. When the training process is terminated, feature vectors are extracted. The process of extracting feature vectors is shown in Figure 6.

Given a GUI skeleton as the input, the encoding process enables the extraction of the feature vectors through a sequence of convolution and pooling operations, and

the middle hidden layer outputs the feature vector of the input GUI skeleton.

Identifying Isomorphic GUIs by Relative Entropy

Relative entropy is utilized to measure the distance between two random distributions. When two random distributions are the same, their relative entropy is zero. The relative entropy increases as the difference between the two random distributions increases. The relative entropy between two random distributions X and Y can be computed as

$$h(X, Y) = \sum_{i=1}^n x_i \log \frac{x_i}{y_i}, \quad (2)$$

where $X = \{x_1, x_2, \dots, x_n\}$, $Y = \{y_1, y_2, \dots, y_n\}$, $x_i \geq 0$, $y_i \geq 0$, and $i = 1, 2, \dots, n$.

In our work, the relative entropy between two GUI skeletons can be computed with (2), where $X = \{x_1, x_2, \dots, x_n\}$ and $Y = \{y_1, y_2, \dots, y_n\}$ represents the feature vectors of the two GUI skeletons. However, when $y_i = 0$, x_i/y_i does not make sense. Divergence measures based on Shannon entropy¹² are used when $y_i = 0$ to solve the above problem:

$$h(X, Y) = \sum_{i=1}^n x_i \log \frac{x_i}{\frac{x_i}{2} + \frac{y_i}{2}}. \quad (3)$$

Evaluation of Our Approach

To evaluate the effectiveness of our method, 80 images containing 54 isomorphic image pairs (because some GUIs have more than one isomorphic GUI pair) were selected from the large-scale data set Rico.¹³ The images cover 10 apps spanning three categories (book, education, and news). The perceptual hashing method¹⁴ is always applied in the image index. The performance of the proposed

method is compared with that of the perceptual hashing method.

The metric, relative entropy, is converted as the similarity to enable a direct comparison of results.¹⁵ Considering that relative entropy is not symmetry, the relevance value r of each image pair is defined as

$$r = h(X, Y) + h(Y, X), \quad (4)$$

where r_{\max} is the max value in the relevance matrix. Finally, the similarity S can be computed as

$$S = 1 - \frac{r}{r_{\max}}. \quad (5)$$

The evaluation data set contains 80 images, and they are divided into two groups. The first group is numbered 1–40 and the second group is numbered 41–80. The isomorphic image pairs are: (1,41), (2,42), ... , (40,80). Each similarity matrix containing 1,600 results is shown in Figure 7. Figure 7(a) presents the ground truth, and Figure 7(b) and (c) shows the similarity matrix of our method and perceptual hashing, respectively.

Grids are colored with different shades of blue according to their similarity. A deep shade of blue corresponds to a large similarity between the image pairs, and vice versa. The similarity $S \geq 0.9$ is regarded as isomorphic GUIs. In the similarity matrix of our method, the shades of color on the diagonal line from the top left to the bottom right are the deepest. Almost all isomorphic GUIs (45/54) can be identified by our method, whereas only about half of isomorphic GUIs (25/54) can be identified by the baseline method. The identification effect of our method is much better than the baseline method and closer to the ground truth.

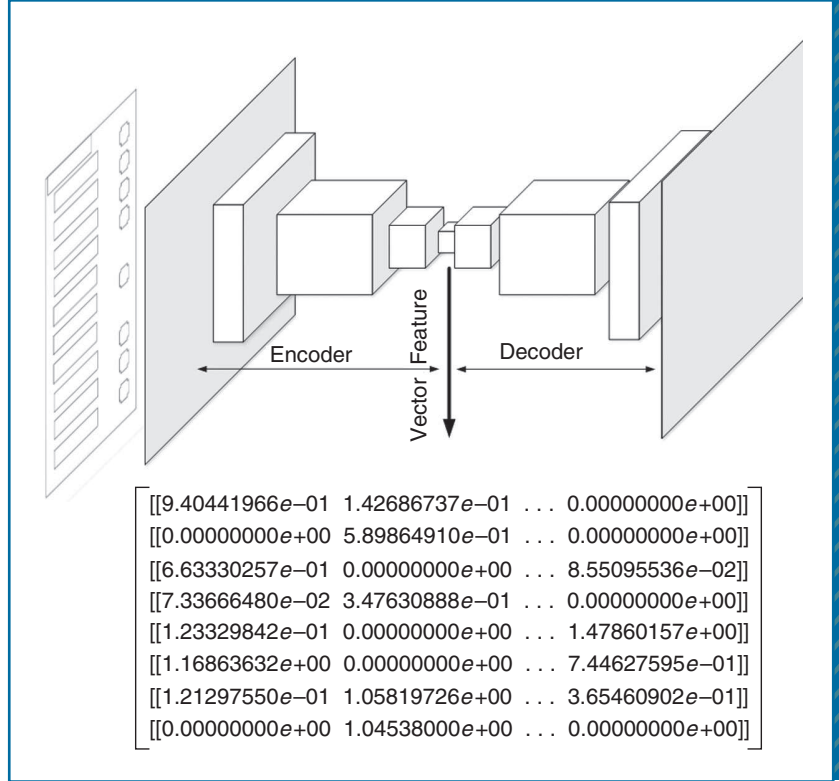


FIGURE 6. Example of extracting feature vectors.

For further evaluation, accuracy Acc , precision P , recall R , and harmonic mean F_1 are used as valuation indexes:

$$Acc = \frac{TP + TN}{TP + TN + FP + FN}, \quad (6)$$

$$P = \frac{TP}{TP + FP}, \quad (7)$$

$$R = \frac{TP}{TP + FN}, \quad (8)$$

$$\frac{2}{F_1} = \frac{1}{P} + \frac{1}{R} \Rightarrow F_1 = \frac{2TP}{2TP + FP + FN}, \quad (9)$$

where TP denotes the number of instances where positive class (isomorphic GUIs) was predicted as positive class; FP denotes the number of instances where negative class (nonisomorphic GUIs) was predicted as positive class; FN denotes the number of instances where

Table 1. Comparison results.

	Proposed method	Perceptual hashing method
TP	45	25
FN	9	29
FP	30	21
TN	1,516	1,525
Acc	97.6%	96.9%
R	0.83	0.46
P	0.6	0.54
F_1	0.7	0.5

positive class was predicted as negative class; and TN denotes the number of instances where negative

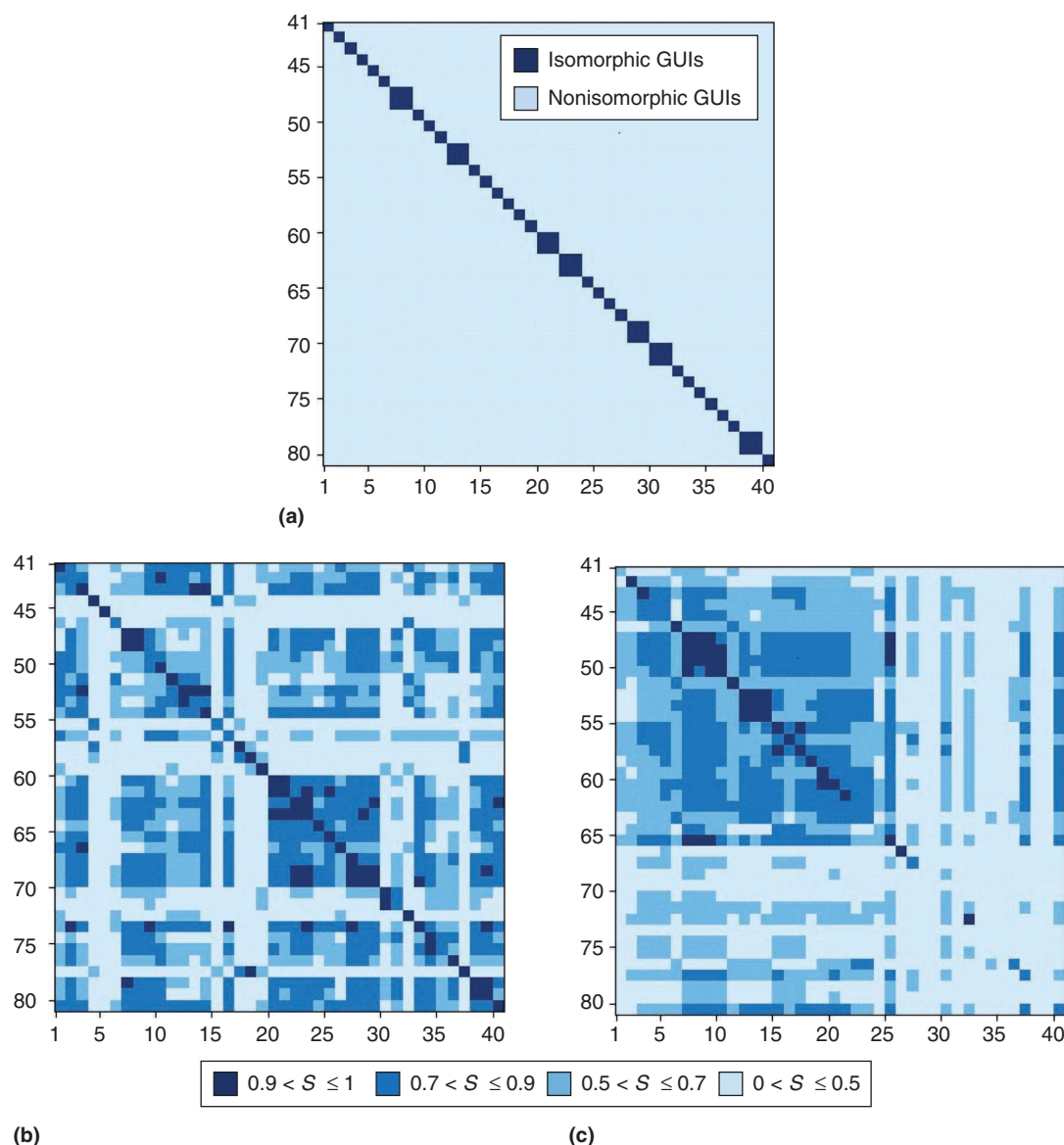


FIGURE 7. Similarity matrix: (a) ground truth, (b) similarity of our method, and (c) similarity matrix of perceptual hashing methods.


class was predicted as negative class. The comparison results are shown in Table 1.

As Table 1 shows, the proposed method performs better than the baseline method in all evaluation

indexes. The harmonic mean F_1 is the comprehensive index of recall R and precision P , as (9) shows, and the F_1 score of our approach is much higher than that of the baseline method. The high accuracy and

similarity matrix (Figure 7) also demonstrates the effectiveness of our method. In conclusion, our approach can handle isomorphic GUI identification well in a vision-based approach.

The state-space explosion problem in GUI models has become an urgent factor in automated black-box robotic testing. An important constraining issue is the identification of isomorphic GUIs as a means to generate simplified GUI models. This study solves the dilemma by proposing an end-to-end trainable model. Isomorphic GUIs can be identified by a vision-based approach without using source codes.

In conclusion, isomorphic GUI identification is sensible in furthering automated mobile application black-box testing. The proposed method provides an image-based approach for identifying isomorphic GUIs, which could be a stepping stone for higher efficiency automated testing. However, due to the limitation of our objective detection method, nondeterministic GUIs, such as AD bar or dynamic UIs, cannot be handled. We will handle nondeterministic GUIs in future work. 

References

1. E. Alegroth, R. Feldt, and L. Ryrholm, "Visual GUI testing in practice: Challenges, problems and limitations," *Empir. Softw. Eng.*, vol. 20, no. 3, pp. 694–744, June 2015. doi: 10.1007/s10664-013-9293-5.
2. K. Mao, M. Harman, and Y. Jia, "Robotic testing of mobile apps for truly black-box automation," *IEEE Softw.*, vol. 34, no. 2, pp. 11–16, Mar. 2017. doi: 10.1109/MS.2017.49.
3. Y. Li, Z. Yang, Y. Guo, and X. Chen, "A deep learning based approach to automated Android app testing, 2019. [Online]. Available: <https://arxiv.org/abs/1901.02633>
4. X. Su, D. Zhang, W. Li, and K. Zhao, "A deep learning approach to Android malware feature learning and detection," in *Proc. 2016 IEEE*

ABOUT THE AUTHORS



TAO ZHANG is an associate professor in the School of Software, Northwestern Polytechnical University, China, and a visiting scholar at the Computer Engineering Department, San José State University, California. His research interests include intelligent robots, machine learning, and software testing. Zhang received a Ph.D. from Northwestern Polytechnical University in 2006. Contact him at tao_zhang@nwpu.edu.cn.



YING LIU is a master's student in the School of Software, Northwestern Polytechnical University, China. Her research interests include mobile robots, deep learning, reinforcement learning, and software testing. Contact her at ying_liu@mail.nwpu.edu.cn.



JERRY GAO is a full professor at the College of Computer Engineering, San José State University, California. His research interests include big data testing and quality validation, cloud-based mobile testing, and smart cities. Contact him at jerry.gao@sjsu.edu.



LI PENG GAO is a lecturer with the School of Software, Northwestern Polytechnical University, China. Gao received a Ph.D. from Wuhan University, China. His research interests include object extraction, intelligent information processing, and big data. Contact him at gaolipeng@nwpu.edu.cn.



JING CHENG is a lecturer with the School of Computer Science and Engineering, Xi'an Technological University, China. Cheng received a Ph.D. from Northwestern Polytechnical University, China. Her research interests include mobile application testing, crowdsourcing testing, artificial intelligence, and software modeling. Contact her at chengjing@xatu.edu.cn.



- Trustcom/BigDataSE/ISPA*, Tianjin, China, 2016, pp. 244–251.
5. Google Play. [Online]. Available: <https://play.google.com/store>
 6. Y. Miao and X. Yang, "An FSM based GUI test automation model,"

- in *Proc. IEEE Conf. Contr. Auto. Robot. & Vis.(ICARCV)*, 2010, pp. 120–126.
7. A. S. Bozkir and E. A. Sezer, "Layout-based computation of web page similarity ranks," *Int. J.*

- Human-Comput. Stud.*, vol. 110, pp. 95–114, 2018. doi: 10.1016/j.ijhcs.2017.10.008.
8. J. Redmon, S. Divvala, and R. Girshick, “You only look once: Unified, real-time object detection,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016, pp. 79–788.
 9. Y. Chen, T. Su, G. Z. Meng, Z. C. Xing, and Y. Liu, “From UI design image to GUI skeleton: A neural machine translator to bootstrap mobile GUI implementation,” in *Proc. IEEE/ACM 40th IEEE Int. Conf. Softw. Eng. (ICSE)*, 2018, pp. 665–676. doi: 10.1145/3180155.3180240.
 10. K. G. Dizaji, A. Herandi, C. Deng, W. D. Cai, and H. Huang, “Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization,” in *Proc. IEEE Int. Conf. Comp. Vis. (ICCV)*, 2018, pp. 5736–5745.
 11. Z. Zou, Z. Shi, Y. Guo, and J. Ye, “Object detection in 20 years: A survey, 2019. [Online]. Available: <https://arxiv.org/abs/1905.05055>
 12. Lin, “Divergence measures based on the Shannon entropy,” *IEEE Trans. Inf. Theory*, vol. 37, no. 1, pp. 145–151, Jan. 1991. doi: 10.1109/18.61115.
 13. B. Deka, Z. F. Huang, Franzen, J. Hirschman, D. Afegan, Y. Li, J. Nichols, R. Kumar, “Rico: A mobile app dataset for building data-driven design applications,” in *Proc. Annu. ACM Symp. User Inter. Softw. Tech. (UIST)*, 2017, pp. 845–854. doi: 10.1145/3126594.3126651.
 14. W. Fang, H. M. Hu, Z. Hu, S. Liao, B. Li, “Perceptual hash-based feature description for person re-identification,” *Neurocomputing*, vol. 272, pp. 520–531, 2018, doi: 10.1016/j.neucom.2017.07.019.
 15. Q. Zhang, M. Li, and Y. Deng, “Measure the structure similarity of nodes in complex networks based on relative entropy,” *Physica. A*, vol. 491, pp. 749–763, 2018. doi: 10.1016/j.physa.2017.09.042.



Call for Articles

IEEE Pervasive Computing

seeks accessible, useful papers on the latest peer-reviewed developments in pervasive, mobile, and ubiquitous computing. Topics include hardware technology, software infrastructure, real-world sensing and interaction, human-computer interaction, and systems considerations, including deployment, scalability, security, and privacy.

Author guidelines:
www.computer.org/mc/pervasive/author.htm

Further details:
pervasive@computer.org
www.computer.org/pervasive

IEEE pervasive COMPUTING
 MOBILE AND UBIQUITOUS SYSTEMS

Digital Object Identifier 10.1109/MS.2020.2996916