J Inf Process Syst, Vol.18, No.2, pp.197~207, April 2022
https://doi.org/10.3745/JIPS.02.0169

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

A Manually Captured and Modified Phone Screen Image Dataset for Widget Classification on CNNs

SungChul Byun[1], Seong-Soo Han[2], and Chang-Sung Jeong[1,*]

Abstract

The applications and user interfaces (UIs) of smart mobile devices are constantly diversifying. For example, deep learning can be an innovative solution to classify widgets in screen images for increasing convenience. To this end, the present research leverages captured images and the ReDraw dataset to write deep learning datasets for image classification purposes. First, as the validation for datasets using ResNet50 and EfficientNet, the experiments show that the dataset composed in this study is helpful for classification according to a widget's functionality. An implementation for widget detection and classification on RetinaNet and EfficientNet is then executed. Finally, the research suggests the Widg-C and Widg-D datasets—a deep learning dataset for identifying the widgets of smart devices—and implementing them for use with representative convolutional neural network models.

Keywords

Captured Image, CNN, Deep Learning Dataset, Image Classification, Object Detection, Widget

1. Introduction

Smart mobile devices offer various applications to users, and due to this convenience, their users are increasing worldwide. Accompanying this increase in the number of smartphone users is a growth in the development of applications and user interfaces (UIs) for the Android operating system (OS) and the iPhone operating system (iOS). However, these more diverse applications necessitate the creation of more platform types and widget forms. This diversity of applications may confuse users, and the test environment for smartphone production also becomes more complicated.

In recent years, artificial neural networks (ANNs) have been utilized in various fields. For example, convolutional neural networks (CNNs) are widely used and show high performance in computer vision fields, such as image classification [1], object detection [2], and real-time processing [3]. In addition, CNNs' fields of application are constantly growing and now include natural language processing [4] and voice detection [5].

As outstanding as CNNs' performance and utilization have been, functional smartphone widget image classification using deep learning has also been proposed For example, Moran et al. [6] wrote the ReDraw dataset and classified various graphical user interfaces (GUIs) in the Android OS by using deep learning.

Manuscript received June 2, 2021; first revision August 4, 2021; second revision September 29, 2021; accepted October 10, 2021.

* **Corresponding Author:** Chang-Sung Jeong (csjeong@korea.ac.kr)
[1] Dept. of Electrical Engineering, Korea University, Seoul, Korea (sungcb@korea.ac.kr, csjeong@korea.ac.kr)
[2] Dept. of Division of Liberal Studies, Kangwon National University, Samcheok, Korea (sshan1@kangwon.ac.kr)

Notably, the image classification model trained with the ReDraw dataset performed with 91% and 86% accuracy on validation tests using newly-captured screen images, leaving room for improvement in smart devices' GUI classification of such images using deep learning.

The primary purpose of this study is to improve convenience by identifying widgets with deep learning by writing a dataset that can detect and classify them according to their roles in screen-captured images. The research proposes a method to detect widget location and predict its functionality by implementing two CNN models on a full-screen image. As a result, the study achieved improvements in GUI image classification performance. Two datasets were written, one for widget classification and one for detection—the Widg-C and Widg-D datasets, respectively—that are more generalized and balanced than the datasets used in traditional studies. The Widg-C and Widg-D datasets are generalized by reducing redundancy and errors, a process that is reinforced by employing the template method and by collecting various captured images. These refining processes made the datasets more suitable for training CNN models.

The overall composition of this paper is as follows. Section 2 discusses the existing method and models for object detection and image classification using a CNN and then describes the existing dataset for widget classification, i.e., the ReDraw dataset. Section 3 proposes the Widg-C and Widg-D datasets, while Section 4 compares and validates the proposed scheme with the extant dataset. An experimental implementation of the proposed widget detection and classification scheme with full screen-captured images is conducted for practical uses.

2. Related Works

This section introduces the dataset background utilized in the classification of widgets and explains the concepts related to the CNN as well as representative CNN models for object detection and image classification.

2.1 Convolutional Neural Network

Currently, various CNNs are applied in several computer vision and deep learning fields. To date, CNNs' image classification performance is superior to that of humans. This study will utilize a representative CNN model to implement the detection and classification of widgets using deep learning and will adopt EfficientNet [7] as a CNN model for image classification to achieve the respective objectives of classification. Furthermore, the most representative model, ResNet-50 [8] (ResNet with 50 layers), which is used in various deep learning fields [9] due to its effective residual learning method, will be used to evaluate the classification's performance. For widget boundary detection, RetinaNet [10] using ResNet-50 as a backbone network will be implemented. Brief introductions of these three representative CNN models are presented below.

2.2 ReDraw Dataset

The ReDraw dataset, written by Moran et al. [6], is a deep learning dataset for classifying GUIs in smart mobile devices. The ReDraw dataset consists of synthetic images created by mocking up the actual widgets and organic images collected automatically from the top 250 Android apps as determined by their popularity on Google Play. Moran et al. [6] cropped these collected screen images and classified

them according to GUI functionality. The ReDraw dataset is augmented to address data imbalances and is cropped to improve data diversification. The dataset is divided into 16 item classes: Button, CheckBox, CheckedTextView, EditText, ImageButton, ImageView, NumberPicker, ProgressBarHorizontal, ProgressBarVertical, RadioButton, RatingBar, SeekBar, Spinner, Switch, TextView, and ToggleButton. The organic images comprise 143,170 training images, 29,040 validation images, and 19,090 test images. As a result, the deep learning GUI classifier using the ReDraw dataset achieved an accuracy of 91%, a performance that can be improved.

The present research will utilize the 16 classifications used in ReDraw equally, having determined that these 16 classes were appropriate for classifying widgets' functions only with the appearance shown as an image. However, the ReDraw dataset has problems with misclassifications and image redundancy. Moreover, there are images that have been cropped in a manner that severely damage the feature. These problems can interfere with the convergence of deep learning models and cause errors. Furthermore, due to a GUI image's characteristics, the image crop can confuse the model and make the GUI feature unrecognizable. From the newly collected test dataset, the classifier trained with ReDraw had an 86% accuracy performance, revealing that ReDraw has room to improve its performance by correcting the problem via a deep learning dataset.

3. Dataset Description

This section introduces the Widg-C and Widg-D datasets—deep learning image datasets for widget classification and detection, respectively. It also introduces how the datasets were created and their composition.

3.1 Full Screen-Captured Image Collection and Cropped Widgets

The datasets were collected by sorting and capturing a highly accessible screen that is frequently exposed to users. All the captured images were manually saved, and the bounding box information was defined by using the BoundingBoxerImg tool [11]. The bounding box region was separated into seven classes—text, image, edit, navi, status, button, and region—that were at first arbitrarily specified. The images were then cropped with the coordination of the bounding box from the full screen-captured image. Afterwards, the cropped images were classified into the same 16 widget classifications as in ReDraw. As an example, Fig. 1 shows a full screen-captured image with bounding boxes visualized and the cropped images from the screen images after being reclassified into the 16 widget classes.

3.2 Widg-C Dataset for Widget Classification

The original dataset has been modified by removing images to reduce redundancy and data imbalances. As a result, the size of the ReDraw dataset was reduced to half of the original. The training dataset grew to 74,771 images by adding 14,373 images captured and cropped from full screen captured images to the modified ReDraw dataset of 60,398 images. The validation dataset consists of 22,297 images that comprise 18,697 images from ReDraw's training and validation datasets plus 3,600 images that were captured and cropped manually. Table 1 represents the configuration of the Widg-C dataset.

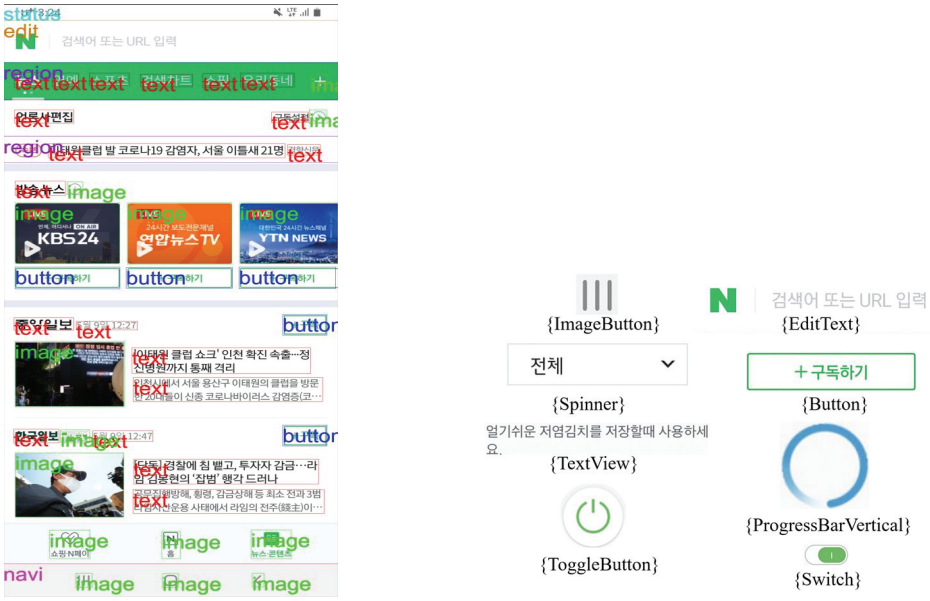


Fig. 1. Examples of full screen-captured images with bounding boxes and cropped widget images.

Table 1. Configuration of the Widg-C dataset

Classes	Training data	Validation data
Button	6,533	1,977
CheckBox	5,252	1,338
CheckedTextView	6,577	1,661
EditText	444	158
ImageButton	6,881	2,591
ImageView	3,785	1,265
NumberPicker	4,720	1,180
ProgressBarHorizontal	3,028	759
ProgressBarVertical	2,476	625
RadioButton	3,960	1,089
RatingBar	3,990	999
SeekBar	4,351	1,092
Spinner	3,459	924
Switch	4,748	1,250
TextView	10,367	4,291
ToggleButton	4,200	1,078
Total	74,771	22,297

3.3 Widg-D Dataset for Widget Detection

The training dataset for the object detection model, based on the bounding box information defined from the full screen-captured image data collected earlier, is proposed as the Widg-D dataset. This dataset uses the seven classes mentioned above. Since the BoundingBoxerImg tool provides annotation in txt format, including coordinates and labels, the training dataset was created by converting the annotation to a csv file and an xml file in PASCAL Visual Object Class (VOC) form [12]. The PASCAL VOC dataset

is a common and popular dataset in the field of object detection using deep learning. Research on object detection using the PASCAL VOC [13] is being actively conducted to improve the performance of deep learning models [14] or to more efficiently design models [15]. The Widg-D dataset was created by referring to the dataset format of the PASCAL VOC.

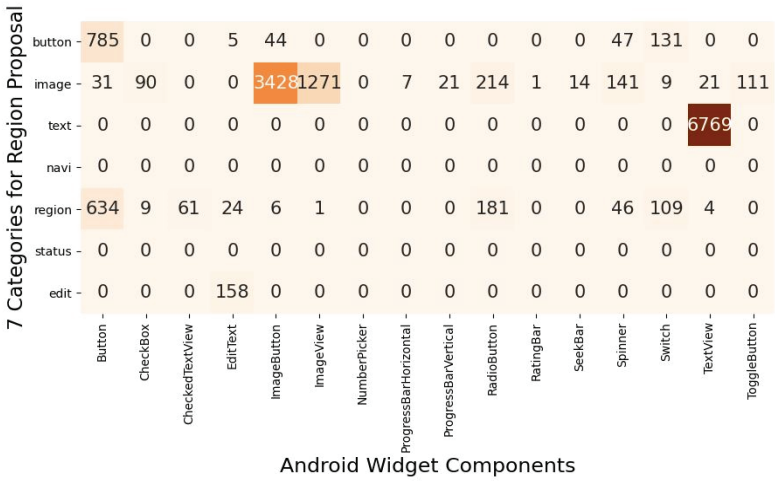


Fig. 2. A heat map of the collected screen images in the detection and classification classes.

The heat map in Fig. 2 shows the distribution of components according to the detection and classification classes of the screen images that were collected and defined with bounding boxes.

3.4 Template Image Dataset Supplementation for Better Region Proposal

The dataset needs to train the model using the intact widgets to detect them in a screen-captured image. Therefore, flip and rotation of images can harm features of the model's training. Consequently, a screen image template dataset has been composed. Such a dataset aims to solve data imbalances by reinforcing insufficient widgets. To replenish the images in the region, edit, and button classes, 100 images of three types of templates and 100 mixed templates containing all three targeted widgets were generated. As a result, 400 total images were added to the Widg-D dataset.

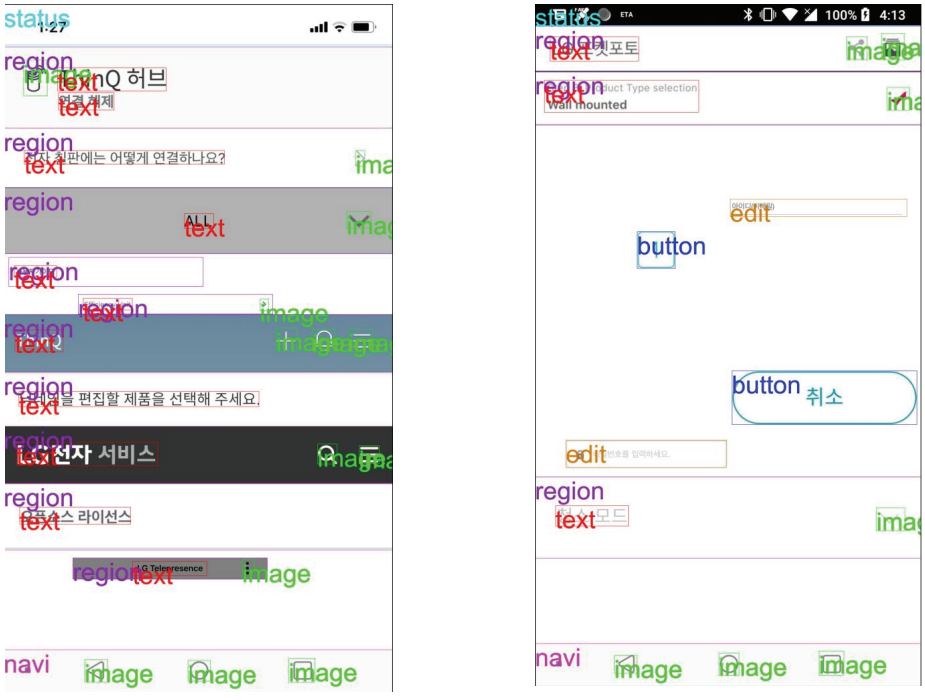
The template was composed by pasting the previously collected and cropped widget components into the background at a resolution of 1980×1080. The components were selected by non-restoring random extraction. A status bar is on top of the background, and a navigation bar is at the bottom of every template. Components which exceed the determined resolution are resized and pasted to the largest size suitable for resolution, while components that do not exceed resolution are pasted at their original size.

Each component is pasted at random intervals on the y-axis and only in an empty area. The insertion of the components is repeated until there is no available y-axis space. Since a template contains one or two columns of components from side to side, small components can be pasted into a two-parted space on the x-axis or into a non-partitioned space, and large components can be pasted without partitioning, even if there are two columns in the template. Components that include other widgets inside, such as regions, are transcribed to preserve and paste their inclusion information. The template exploits the manually written widget dataset as diversely as possible due to non-restoring randomization.

The described template writing methods can prevent detection models from training the component's location or from spacing and overfitting due to the widgets' characteristics. Fig. 3 is an example of the region and mixed templates.

There is also an additional class, named "small," to assist in the detection of smaller widgets, which are transferred to the small category if their area is less than 600 pixels. The configuration of the Widg-D dataset, including modifications and the template image dataset, is expressed in Table 2.

Categorizing and generalizing GUI components with myriad forms based on functionality is a complex subject. The modification of the ReDraw dataset was intended to increase the dataset's universality while eliminating data imbalances. In addition, multiple apps were captured, and randomly-created templates were added to the dataset to ensure generality. The Widg-C and Widg-D datasets secure the universality of the GUI image dataset in the Android OS through this data diversification process.



(a) (b)

Fig. 3. A sample image of template data: (a) region template and (b) mixed template.

Table 2. Configuration of the Widg-D dataset

Classes	Widg-D
text	8,462
image	7,535
button	2,165
region	3,560
status	1,228
navi	720
edit	1,093
small	169
Total	24,932

4. Experiment

To compare performance in widget classification trained with the Widg-C and ReDraw datasets, ResNet-50 was used, and EfficientNet was utilized to validate the Widg-C dataset's classification accuracy. RetinaNet, which uses ResNet-50 as a backbone network, was employed as a detection module by training the model with the Widg-D dataset to validate the practicality of widget classification from full-screen images. Finally, using RetinaNet's bounding box credentials, a complete implementation of widget classification was executed by classifying the detailed features of the widget with EfficientNet, which trained using the Widg-C dataset.

All experiments were implemented by leveraging Keras embedded in the TensorFlow 2.3.0 library in PyThon 3.6.13. The two models were trained using a learning rate of 10^{-5}, and the softmax function was used as the activation function. The EfficientNet for image classification was trained with a batch size of 16 using an RMSprop optimizer, introduced in Hinton's lecture [16]. The RetinaNet for widget detection was trained with a batch size of 4 using an Adam optimizer with the gradient clipping constant set to 0.001 [17].

4.1 Comparison between the ReDraw and Widg-C Datasets

To compare the classification performance, two ResNet-50 models were trained on two datasets for 30 epochs, and the changes in loss and accuracy were compared. A dataset's size can affect loss and accuracy changes; we randomly cropped the ReDraw dataset to construct training and validation set in the same size of the Widg-C dataset. The results of the accuracy and loss in training and the validation of the two datasets are shown in Fig. 4.

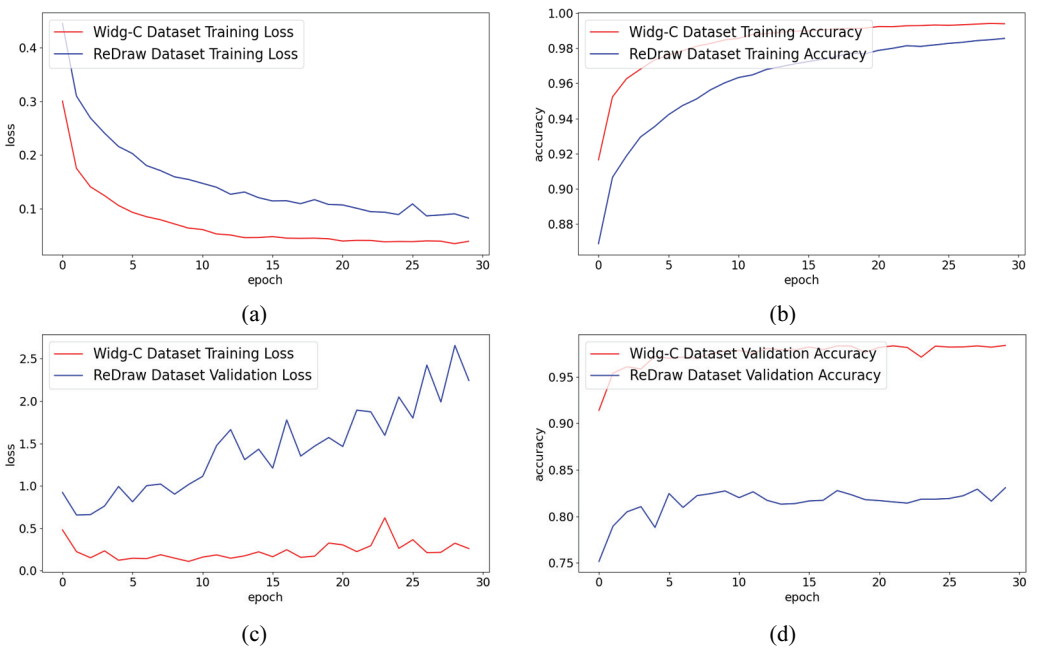


Fig. 4. The loss and accuracy in the training and validation process of ResNet-50 for the ReDraw and Widg-C datasets. (a) Training loss, (b) training accuracy, (c) validation loss, and (d) validation accuracy.

4.2 Accuracy of the CNNs Trained with the Widg-C Dataset

A total of 2,754 images are in the test dataset, collected identically to the method used to collect the Widg-C dataset, to verify the classification performance on practical screen-captured images. This test dataset was generated by selecting images that are likely to be easily exposed to users. Its configuration is shown in Table 3.

The EfficientNet B0 and B3 models were trained for 30 epochs using the Widg-C dataset with the same training method as with ResNet50. The test results validating the performance of ResNet and EfficientNet are shown in Table 4. As shown in Table 4, all three models performed with high accuracy rates of more than 95%. The CNN models trained with the Widg-C dataset were not only able to classify widget images but were also able to achieve high accuracy on functional widget classification.

Table 3. Configuration of the test dataset

Classes	Test data
Button	261
CheckBox	89
CheckedTextView	123
EditText	156
ImageButton	629
ImageView	283
NumberPicker	88
ProgressBarHorizontal	69
ProgressBarVertical	48
RadioButton	47
RatingBar	16
SeekBar	54
Spinner	42
Switch	45
TextView	759
ToggleButton	45
Total	2,754

Table 4. Accuracy of 3 CNN models trained with the Widg-C dataset

Model	Accuracy (%)	Macro average (%)
ResNet-50	95	96
EfficientNetB0	97	98
EfficientNetB3	98	99

4.3 Implementation for Full Screen Image Detection and Classification

This experiment implemented a wide-get region and feature estimation using CNNs from a real-world full screen-captured image accessible to users. The experiment employed the object detection model using the Widg-D dataset and the image classification model training using the Widg-C dataset. The bounding box information was obtained from the object detection model prediction for guessing the function of the detailed widget through the inference of the image classification model. RetinaNet was

trained using ResNet-50 as a backbone network for 60 epochs using the Widg-D dataset and EfficientNet B3, which was trained earlier for classification. Fig. 5 is a wide-get prediction result sample of full screen-captured images using RetinaNet and EfficientNet B3.

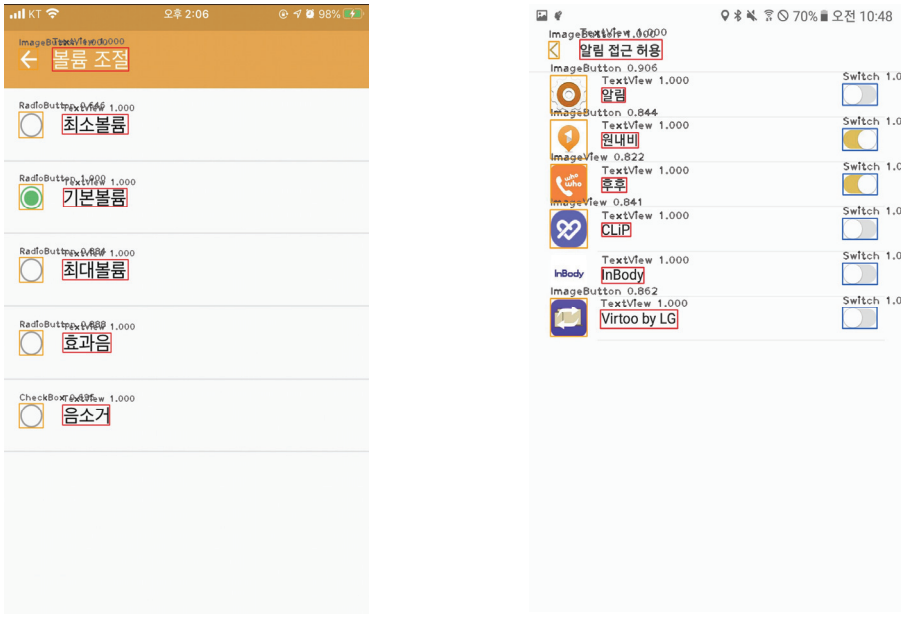


Fig. 5. Sample result images from the implemented CNN prediction.

As a result of the implementation, the detection and classification models detected regions and guess the widgets' functions accurately. However, there are also undetected regions and inaccurate boundary predictions.

5. Conclusion

The market for smart mobile devices and their applications is constantly changing and diversifying and thus often presents complexities for both users and engineers. The present research proposes to solve these problems with deep learning using CNNs. The modification of the existing dataset and manually captured and cropped images from multiple apps created a universal and diverse dataset. Furthermore, artificially-generated template images reinforced the dataset's balance. As a result, this more generalized dataset prevents deep learning models from being wrongly trained.

After training several representative CNNs with the Widg-C dataset, the Widg-C dataset was deemed suitable for classifying the widgets of screen-captured images. Deep learning models using the Widg-C dataset achieved accuracy rates of more than 95% on functional widget image classification. In addition, the object detection model demonstrated the possibility of performing the task of predicting not only functions but also locations to users using only images. Thus, the deep learning technique using the Widg-C and Widg-D datasets could provide guidelines for various screen functions without interacting with the device in question.However, the Widg-C and Widg-D datasets' size needs to be enlarged. The ReDraw

training dataset has 143,170 images, while the Widg-C dataset, with 74,771 images, is half its size and therefore requires additional data to be suitable for deep learning. Furthermore, widget images are diverse in their shapes and appearances; hence, the dataset needs to be made more versatile by being supplemented with more varied kinds of images.

Cropped screen-captured images vary too much in image size to train CNN models. Moreover, this variance raises concerns about the failure of the feature detection during resizing and preprocesses. Due to these problems, we propose adjusting the input of the learning model. Furthermore, the TextView widget is expected to enable more accurate region extraction by leveraging models aimed at optical character recognition (OCR).

The method suggested is currently problematic when dealing with the accurate extraction of regions and component losses. Deep learning models need to be trained with more classes to learn deeper for excessively small or significant features to be improved.

Currently, we are collecting more captured images to reinforce the dataset and are designing a widget classifier for full-screen images that utilizes both region proposal and image classification models. Furthermore, we aim to build an identifier for captured images from any smart mobile device that will make it easier for users to access parts of applications and for producers or developers to more easily simulate and test various functions.

References

[1] T. Akram, H. M. J. Lodhi, S. R. Naqvi, S. Naeem, M. Alhaisoni, M. Ali, S. A. Haider, and N. N. Qadri, "A multilevel features selection framework for skin lesion classification," *Human-centric Computing and Information Sciences*, vol. 10, article no. 12, 2020. https://doi.org/10.1186/s13673-020-00216-y

[2] D. Cao, Z. Chen, and L. Gao, L. (2020). An improved object detection algorithm based on multi-scaled and deformable convolutional neural networks. *Human-centric Computing and Information Sciences*, vol. 10, article no. 14, 2020. https://doi.org/10.1186/s13673-020-00219-9

[3] J. Lee and K. I. Hwang, "RAVIP: real-time AI vision platform for heterogeneous multi-channel video stream," *Journal of Information Processing Systems*, vol. 17, no. 2, pp. 227-241, 2021.

[4] S. Shokat, R. Riaz, S. S. Rizvi, A. M. Abbasi, A. A. Abbasi, and S. J. Kwon, "Deep learning scheme for character prediction with position-free touch screen-based Braille input method," *Human-centric Computing and Information Sciences*, vol. 10, article no. 41, 2020. https://doi.org/10.1186/s13673-020-00246-6

[5] S. D. You, C. H. Liu, and W. K. Chen, W. K. (2018). Comparative study of singing voice detection based on deep neural networks and ensemble learning. *Human-centric Computing and Information Sciences*, vol. 8, article no. 34, 2018. https://doi.org/10.1186/s13673-018-0158-1

[6] K. Moran, C. Bernal-Cardenas, M. Curcio, R. Bonett, and D. Poshyvanyk, "Machine learning-based prototyping of graphical user interfaces for mobile apps," *IEEE Transactions on Software Engineering*, vol. 46, no. 2, pp. 196-221, 2018.

[7] M. Tan and Q. Le, "Efficientnet: rethinking model scaling for convolutional neural networks," *Proceedings of Machine Learning Research*, vol. 97, pp. 6105-6114, 2019.

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, NV, 2016, pp. 770-778.

[9] H. Han, "Residual learning based CNN for gesture recognition in robot interaction," *Journal of Information Processing Systems*, vol. 17, no. 2, pp. 385-398, 2021.

[10] T. Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal loss for dense object detection," in *Proceedings of the IEEE International Conference on Computer Vision*, Venice, Italy, 2017, pp. 2999-3007.
[11] BoundingBoxerImg [Online]. Available: https://github.com/jms0923/BoundingBoxerImg.
[12] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (VOC) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303-338, 2010.
[13] M. Aamir, Y. F. Pu, W. A. Abro, H. Naeem, and Z. Rahman, "A hybrid approach for object proposal generation," in *The Proceedings of the International Conference on Sensing and Imaging*. Cham, Switzerland: Springer, 2017, pp. 251-259.
[14] M. Aamir, Y. F. Pu, Z. Rahman, W. A. Abro, H. Naeem, F. Ullah, and A. M. Badr, "A hybrid proposed framework for object detection and classification," *Journal of Information Processing Systems*, vol. 14, no. 5, pp. 1176-1194, 2018.
[15] Y. Guan, M. Aamir, Z. Hu, W. A. Abro, Z. Rahman, Z. A. Dayo, and S. Akram, "A region-based efficient network for accurate object detection," *Traitement du Signal*, vol. 38, no. 2, pp. 481-494, 2021.
[16] G. Hinton, N. Srivastava, and K. Swersky, "Neural Networks for Machine Learning: overview of mini-batch gradient descent (Lecture 6a)," [Online]. Available: http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf.
[17] D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization," 2014 [Online]. Available: https://arxiv.org/abs/1412.6980.

SungChul Byun **https://orcid.org/0000-0001-5847-5293**

He is currently pursuing an M.S. degree in the Department of Electrical Engineering at Korea University. He received a B.S. from the Department of Electrical Engineering at Korea University in 2020. His research interests include deep learning/computer vision and distributed parallel processing.

Seong-Soo Han **https://orcid.org/0000-0002-4915-6247**

He is a professor in the division of Liberal Studies at Kangwon National University. Before joining Kangwon National University in 2019, he was a professor at Soonchunhyang University. He received a B.S. in Information and Communication Engineering from Gyeongsang National University, an M.S. in Information and Communication Engineering from Soonchunhyang University, Korea, in 2005, and a Ph.D. in Visual Information Processing from Korea University in 2019. He was a Director of Orion Technology in 2015–2016. His research interests include computer education, AI, blockchain, deep learning, and distributed parallel processing.

Chang-Sung Jeong **https://orcid.org/0000-0001-9654-8406**

He is a professor in the Department of Electrical Engineering at Korea University. Before joining Korea University in 1992, he was a professor at POSTECH from 1982 to 1992. He was on the editorial board for the Journal of Parallel Algorithms and Application from 1992 to 2002. In addition, he was a chair of the IEEE Seoul Section and has been working as a chair of the Computer Chapter in the Seoul Section of IEEE region 10. He was chair of the EE Department in Korea University and a leader of the BK21 project. His research interests include distributed parallel computing, cloud computing, networked virtual environments, and distributed parallel deep learning for real-time image processing and visualization.