A Screen-captured Image Dataset for Widget Classification on CNN

SungChul Byun[1], Seong-Soo Han[2], Chang-Sung Jeong[1],
[1] Department of Electrical Engineering, Korea University, Seoul, Republic of Korea
[2] Department of Division of Liberal Studies, Kangwon National University, Samcheok, Republic of Korea
{sungcb}@korea.ac.kr, {sshan1}@kangwon.ac.kr, {csjeong}@korea.ac.kr

Abstract. The application and UI of mobile smart devices are constantly diversifying. We use deep learning to classify widgets in screen images to promote convenience. To this end, we leverage capture images and ReDraw dataset to write deep learning datasets for image classification purposes. We validate datasets using ResNet50 and EfficientNet, and our experiments show that the dataset we write is useful for classification according to the widget's functionality. We built Widg-C dataset—a deep learning dataset for identifying the widget of smart devices—and validated them with representative CNN models.

Keywords: CNN, Image Classification, Widget, Captured Image, Dataset

1 Introduction

Mobile smart devices offer various applications to users, and due to this convenience, users of mobile smart devices are increasing worldwide. As the number of smartphone users has increased and the smartphone market has become more active, the development of applications and User Interface (UI) to use them is also rapidly growing. Android Operating System (OS) and iOS are introducing new applications from many developers to users, focusing on Google Play and Appstore, respectively.

However, the more applications are diversified, the more diverse the platform and widget forms are. It means that it can confuse users, and the test environment for smartphone production is also diversified.

We aim to implement convenience by using deep learning to identify such diverse and complex widgets. We conduct this study intending to write a dataset to classify widget according to its role in screen captured images.

2 Related Works

In this section, we introduce the dataset background that is utilized for the classification of widgets. We also introduce concepts related to the Convolutional Neural Network (CNN) used to implement them.

2.1 CNN

CNN utilizes the computation of neural networks and is applied in various computer vision and deep learning fields. To date, the image classification performance of CNNs is so superior that it is more accurate than that of humans. We use ResNet50 [1] (ResNet with 50 Layers) and EfficientNet [2] as representative CNNs to classify images of screen-capture and cropped widget in this work.

2.2 ReDraw Dataset

ReDraw dataset [3], written by Kevin Moran et al., is a deep learning dataset for classifying Graphical User Interface (GUI) for smart mobile devices. ReDraw dataset consists of Synthetic images created by Mock-up the actual widget and Organic images collected in an automated manner from the top 250 Android apps popular in each category of Google Play. Kevin Moran et al. cropped these collected screen images and classified them according to the GUI functionality. ReDraw dataset was also done with augmentation to address data imbalances and image crops to improve data diversification. The dataset is divided into 16 classes of items: Button, CheckBox, CheckedTextView, EditText, ImageButton, ImageView, NumberPicker, ProgressBarHorizontal, ProgressBarVertical, RadioButton, RatingBar, SeekBar, Spinner, Switch, TextView, and ToggleButton. It is also the same designation that is used according to the role of the widget. The Organic image consisted of 143,170 images, 29,040, and 19,090 images, respectively, Training, Validation, and Test dataset.

We will utilize the 16 classifications used in ReDraw equally. We determined that these 16 classifications were appropriate to classify its functions only with the appearance shown as an image. However, the ReDraw dataset has problems with misclassifying and image redundancy. And there are images where the image crop severely damages the feature. These problems can interfere with the convergence of Deep Learning models and cause errors. Furthermore, due to the GUI image's characteristic, the image crop can confuse the model's learning and makes the GUI feature unrecognizable.

3 Dataset Description

In this section, we introduce Widg-C dataset—a deep learning image dataset for widget classification. We will also introduce the way we wrote the dataset and its composition.

Full screen-captured image collection and cropping widgets. We guess, output, and capture a highly accessible screen that is frequently visible to users. The captured images were manually saved bounding box information using BoundingBoxerImg tool [4] and subsequently modified with an algorithm. The bounding box region was classified into seven classes–text, image, edit, navi, status, button, region–that we arbitrarily specified at first. We then crop the images as the coordination of the bounding box from the full screen-captured image. Then, we classify the cropped image into 16 widget classifications equal to ReDraw. Fig. 1 shows full screen-captured images with bounding boxes and images cropped from the screen image then reclassified to 16 widgets.

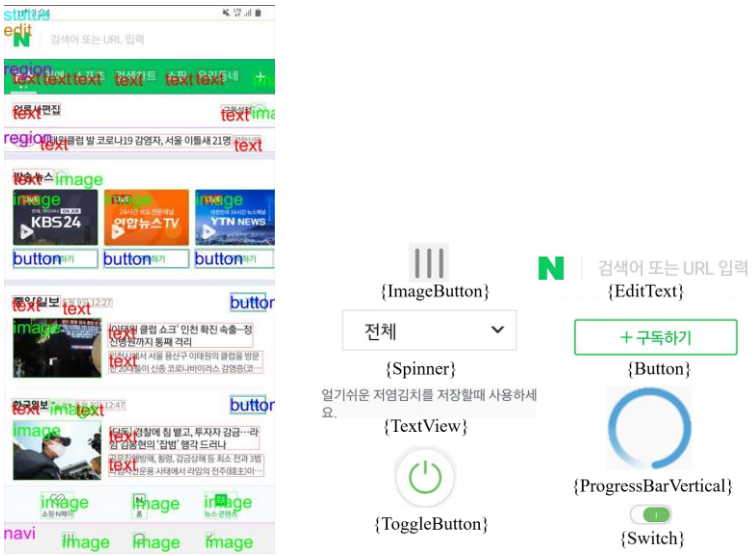

Fig. 1. Examples of full screen-captured image with bounding box and cropped widget images

Dataset for Widget Classification: Widg-C dataset. We deleted images to address redundancy and data imbalances of the same images—this reduced dataset to about half the ReDraw dataset size. As a result, the training dataset consists of 74,771 images; added 14373 images we have captured and cropped to the modified ReDraw dataset of 60,398 images. The validation dataset consists of 22,297 images; used 18,697 images of ReDraw's training and validation datasets and adding 3,600 images that we captured and cropped. Table 1 represents the configuration of Widg-C dataset.

Table 1. Configuration of 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	**74771**	**22297**

4 Experiment

We compare with ReDraw dataset using ResNet50 to validate our written dataset and implement classification accuracy using EfficientNet.

We implement CNN by leveraging Keras embedded in Tensorflow 2.3.1 Library and Tensorflow in Python 3.8.5 version for experiments. CNN models for experiments were all optimized using RMSprop introduced in Geoff Hinton's Lecture [5].

4.1 Comparison between ReDraw Dataset and Widg-C dataset

We trained ResNet50 on two datasets for 30 epochs and compared the changes in loss and accuracy. The dataset's size can affect loss and accuracy changes; we extracted the ReDraw dataset and constructed training and validation images in the number 74,771, 22,297 same as Dataset for Widget Classification.

The results of comparing the training accuracy, loss, validation accuracy, and loss of the two datasets are shown in Fig. 2.

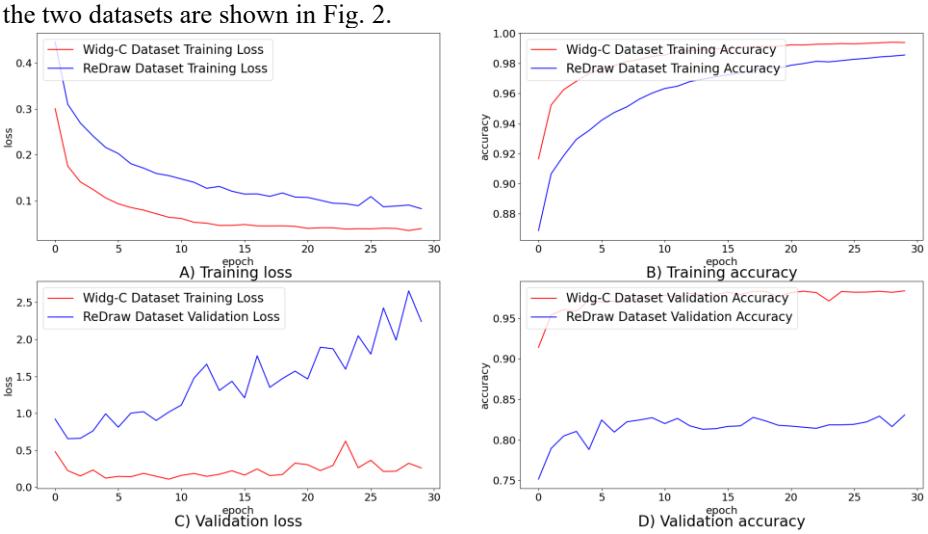

Fig. 2. The loss and accuracy in the training and validation process of ResNet50 for ReDraw and Widg-C dataset.

The training and validation accuracy graphs show that Widg-C dataset is always higher for 1 to 30 epochs than ReDraw dataset, and training and validation loss graphs are always lower and more stable. It shows that the Widg-C dataset is better refined and better classified by feature to classify widget than the ReDraw dataset.

4.2 Accuracy of CNNs trained with Widg-C dataset

We write 2,754 images of the test dataset, identical to the method we collected Widg-C dataset, to verify the classification performance on real-world screen captured images. This test data set is prepared by selecting images that are likely to be easily accessible to users, and the configuration is shown in Table 2.

Table 2. Configuration of 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**

We trained EfficientNet B0, B3 models for 30 epochs using Widg-C dataset with the same training method with ResNet50. The test results of validating the performance of ResNet and EfficientNet are as shown in Table 3.

Table 3. Accuracy of 3 CNN Models trained with Widg-C dataset.

Model	accuracy	macro average
ResNet50	95%	96%
EfficientNetB0	97%	98%
EfficientNetB3	98%	99%

As shown in Table 3, all three models performed high accuracy of more than 95% for representative widget images in 2,754. CNN models using Widg-C dataset were able to perform well in the classification of screen captured widget images.

5 Conclusion

The market for smart mobile devices and their applications is constantly changing and diversifying. It often presents complex concerns for both users and creators of smart mobile devices. We propose to solve these problems with Deep Learning using CNN. After training several representative CNNs with our Widg-C dataset, the Widg-C dataset is suitable for classifying widgets in screen-captured images.

However, Widg-C lacks the size of the dataset. For ReDraw's training dataset, it was 143,170 images, while Widg-C is half the size, 74,771, which requires supplementation of data to utilize as learning data for deep learning. Furthermore, widget images are diverse in their shape and appearance, so it will be a more versatile deep learning dataset only when supplemented with more varied kinds of images to the dataset.

Cropped screen-captured images vary too much in image size to train CNN models. It raises concerns about the failure of the feature detection while doing resize and preprocesses. Due to these problems, we propose to adjust the input of the learning model.

Henceforth, we will write collect screen image data to augment the data and write a widget classifier of the completed screen image utilizing both region proposal and image classification models. Through this, we want to make it possible to identify areas that perform functions with the only screen captured images. It will make it easier for users to access the parts of applications and the UI and make it easier for producers or developers to simulate and test various functions.

References

1. He, K., Zhang, X., Ren, S., Sun, J.: Deep Residual Learning for Image Recognition. in Proc. CVPR. pp. 770--778 (2016)
2. Tan, M., Le, Q.V.: Efficientnet: Rethinking Model Scaling for Convolutional Neural Networks. arXiv preprint arXiv:1704.04861 (2017)
3. Moran, K., Bernal-Cardenas, C., Curcio, M., Bonnet, R. Poshyvanyk, D.: Machine Learning-Based Prototyping of Graphical User Interfaces for Mobile Apps. IEEE Transactions on Software Engineering, vol. 46, No. 2 pp. 196--221. (2020)
4. BoundingBoxerImg, https://github.com/jms0923/BoundingBoxerImg
5. Hinton, G.: Lecture 6e: Neural Networks for Machine Learning, http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf