Themes Validation Tool

Everlandio Fernandes(✉), Rodrigo Correia, Adriano Gil, Juliana Postal, and Mikhail R. Gadelha

Sidia Institute of Science and Technology, Manaus, Brazil
{everlandio.fernandes,rod.correia,a.gil,j.postal, mikhail.gadelha}@sidia.com
https://www.sidia.org.br/en/home/

Abstract. When themes for smartphones are created, several elements can be customized. They are usually colorful and with artistic drawings in the background. One of the main challenges in creating themes is the validation of the color contrast between the fore- and background elements, so people with visual impairments can perceive the difference between them. Human validation of low visibility issues requires a lot of effort and can generate undesirable bias. Thus, an automated bias-free system is necessary for validation of smartphone themes. In this context, we propose the use of a machine learning technique known as automatic object detection to identify and classify elements with contrast issues. We used a dataset of 1,500 artificially generated Android OS homescreens to train a Faster-RCNN, each screen with at least ten elements (icons) and a balanced distribution between four levels of contrast. Our tool obtained high performance when detecting objects and high accuracy when classifying them.

Keywords: Themes evaluation · Human computer interaction · Faster RCNN · Machine learning

1 Introduction

Users tend to customize the interfaces of their computer systems to provide a personal sense of identity, e.g., a wallpaper with their family photo or a soccer team theme. In particular in smartphones, several elements can be customized when designing a new theme such as wallpapers, widgets, icon packs, sounds, etc. Usually they are colorful and with artistic designs in the background. When designers develop a new theme, this should be sent to a theme store to be marketed, where they go through a series of validations.

One of the main challenges during the approval of smartphone themes is the validation of the color contrast between their elements fore- and background. This is required to comply with the "Web Content Accessibility Guidelines" (WCAG) 2.0 recommendations [1], which covers a wide range of suggestions to make Web content more accessible. In particular, item 1.4.3 of the WCAG 2.0 defines the recommended contrast ratio between elements, so that it can be perceived by people with moderately low vision impairments.

C. Stephanidis and M. Antona (Eds.): HCII 2019, CCIS 1088, pp. 16–22, 2019.
https://doi.org/10.1007/978-3-030-30712-7_3

Human validation of low visibility issues requires huge efforts since hundreds of operating system screens need to be validated. Furthermore, an unwanted human bias on what is low contrast will be introduced in the validation. In this context, the benefit of an automated validation system for smartphone themes is twofold: it reduces the number of designers submitting faulty themes and helps the store validator not approve a theme that has a contrast problem.

To solve this problem, we propose the use of a machine learning technique known as automatic object detection. It handles the task of detecting instances of semantic objects in digital images and videos. The automatic detection of objects in images has improved drastically in recent years with the advent of deep learning techniques. One of the most popular object-detection algorithms is Faster-RCNN, which is the basis for many derived networks. Due to its high precision, speed and simplicity, we have selected this architecture to develop our automatic theme validation tool.

We trained our tool using an artificial data set of 1,500 screenshots of the Android OS home screen. The home screens contain at least ten objects (icons) and a balanced distribution between 4 (four) levels of contrast concerning the background: very low, low, regular, and high. Our tool achieved high performance, detecting objects, and high accuracy when classifying them, reaching a mean Average Precision (mAp) of 79,91%.

2 Faster RCNN

Object detection, one of the most fundamental and challenging problems of computer vision, seeks to locate object instances from a large number of predefined categories in natural images; given an image or a frame of a video, the automatic object detection aims to obtain:

- a list of bounding boxes.
- a label assigned to each bounding box.
- a probability for each label and bounding box.

Faster RCNN [2] is an architecture for object detection in images originally presented in Twenty-ninth Conference on Neural Information Processing Systems - NIPS 2015. It is one of the most popular object detection architectures using Convolutional Neural Networks (CNN) [3] to extract features from images and videos.

The neurons of the convolutional layers function as the filters of the classic image processing, i.e., an array of values representing a particular pattern that must be searched in the image. The filter traverses the entire image searching its pattern; the filter is fired when the pattern is detected. Activating all the filters on a layer produces the activation map, also known as a feature map [4].

The feature map of one layer serves as input to the next layer, increasing the level of abstraction of the patterns sought. Usually, they begin with edge patterns in the first layer, followed by more complex patterns in deeper layers,

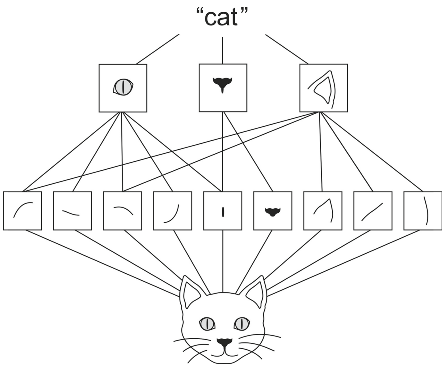

Fig. 1. Edges combine into local objects such as eyes or ears, which combine into high-level concepts such as “cat”. Image reproduced from [5]

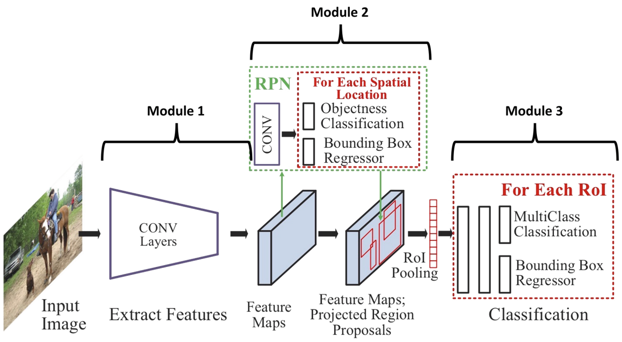

Fig. 2. Faster RCNN. Image reproduced from [6]

e.g., a cat’s eye is identified in the intermediate layers while the cat’s face is identified in a deeper layer, as shown in Fig. 1, reproduced from [5].

The architecture of the Faster RCNN, as shown in Fig. 2, consists of three modules. The input images are represented as tensors height × width × channels (multidimensional arrays), which are passed by a CNN to produce feature maps. These feature maps are consumed by a Region Proposal Network (RPN) [2]. RPN uses the feature maps to find a predefined number of regions which are highly likely to contain objects. Finally, the classification module uses the features extracted by the CNN, and the bounding boxes with relevant objects produced by the RPN to classify the image, and to predict the offset values for the bounding boxes.

3 Related Work

Nguyen et al. [7] propose a method to infer UI elements hierarchy in screenshots of mobile application screens and to generate working computer code. Authors use border detection techniques like Canny Edges [8] and Edges Dilatation to identify elements of interest in the given screenshots, find their contours, compute their bounding boxes, and use heuristics to estimate which UI component are each one based on its features. Mozgovoy et al. [9] address the challenge of hand-drawn user interface elements detection to improve GUI tests automation, and use mobile games as case of study. The experiments consisted of using several image pattern matching algorithms to compare their precision in recognizing elements of interest, and to suggest which algorithm should be used to analyse a given game screen with its particular features. Chen et al. [10] present a method that generates a GUI skeleton code from UI design images. Differently from [7], however, neural networks are used identify the visual elements in the UI images. The authors use Convolutional Neural Networks (CNN) [3] to learn features from images and a Recurrent Neural Network (RCNN) [11] to infer spatial layouts, which are used to produce GUI skeleton code. Similar to these works, our tool is aimed to find visual elements in a screenshot, but we are mainly interested in classify the elements' contrast ratio. Similar to Chen et al. [10], we use a neural network to identify and classify the elements but we are using the Faster RCNN instead.

4 Experiments

In this section, we describe the methodology used to build and train our tool. First, we describe how the dataset of Android OS Home screens was generated in Sect. 4.1, followed by the description of the parameters and configurations used by the Faster RCNN in Sect. 4.2.

4.1 Dataset

In order to have a very diversified dataset of Android OS home screen, we decided on artificially generation of images instead of an extensive and laborious manual screenshot gathering. Each synthetic image simulates a screenshot from Home Screens of Android OS. We picked a resolution of 1440×2960, which is used in last flagship smartphones, Samsung S9.

We are targetting only Home screens because of the high variance of its user interface (UI) elements, i.e, its background image, icons, labels exhibit a higher intraclass difference.

Our synthetic-based dataset generation algorithm generates a screenshot from a set of overlayed images: the full-screen wallpaper, the applications icons, and the application labels. This implementation is written in python and makes use of OpenCV [12] to read and blend the set of overlayed images. Furthermore, we induce 4-level contrast errors by hue shifting icon images to white colors, and ensuring that their luminance channel are proportional to background luminance. A greater proportional difference means a high contrast while a low difference means low contrast. Figure 3 shows an example of a generated home screen.

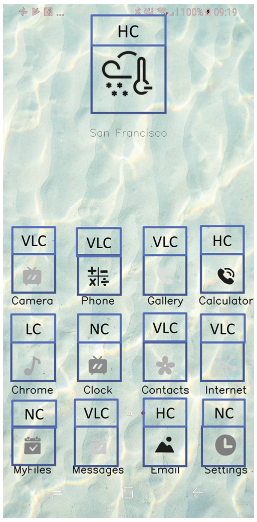

Fig. 3. Example of a synthetic home screen with icons at different levels of contrast and their labels as classified by our tool. Tags: Very Low Contrast (VLC), Low Contrast (LC), Normal Contrast (NC), and High Contrast (HC).

4.2 Faster RCNN

We used the configuration of the Faster RCNN[1] as suggest by the authors in the paper where it was first presented [2], except for the neural network that generates the feature maps: where the authors use a pre-trained VGG16 [13], we use a ResNet [14] trained from scratch for our problem.

Currently, ResNet architectures have replaced VGG as primary networks to extract features in many situations. The main advantages of ResNet over VGG is that it facilitates the training of deep models with the use of residual connections and batch normalization features not presented in the VGG architecture. Furthermore, ResNet is larger then VGG, so it has more ability to learn what is needed.

For the training, we defined the quantity of 100 epochs, each with 100 cycles. The training took place on a GeForce GTX 1080 GPU installed on a desktop device running Windows 10 Enterprise 64-bit, and had training time of approximately 26 h.

5 Results

As previously stated, we used the 1500 artificially generated screen shots to train our theme validation tool. We create the images with a balanced number of icons

[1] Keras implementation of Faster R-CNN. Available on https://github.com/yhenon/keras-frcnn/.

in 4 levels of contrast and with their respective markings (bounding boxes). The training was performed with 100 epochs, each with 100 cycles. After the training, we generated a new dataset of 300 screen shots to perform the tool tests. The images were generated with the same characteristics of the training images, in number of objects and contrast levels.

As evaluation metric, we have used mean Average Precision (mAP) [15]. This metric performs an acceptable balance between the errors of the detected bounding boxes and the classification of the object present in each bounding box. Then it is widely used in works related to object detection in images.

Our tool achieved a mAP of 79.91% in the test set. The Average Precision for very low contrast icons was 81.14%, 71.00% for low contrast icons, 73.62% for normal contrast and 93.87% for high contrast icons. This result indicates that our tool has high performance for identifying and classifying objects from low to high contrast.

6 Conclusion and Future Works

In this paper, we presented our tool for validation of smartphone themes based on a machine learning technique known as object detection. This tool was used to validate the visual problems of contrast between elements of new themes. Our tool uses Faster RCNN method to locate icons in screenshots of the Android OS Home screen, and classifies them into one of the four (4) pre-defined contrast level categories (very low, low, regular, and high). We plan to extend our tool so that it also identifies contrast problems in texts and videos that are part of customization on themes for smartphones.

References

1. Web content accessibility guidelines (WCAG) 2.0. web, December 2008
2. Ren, S., He, K., Girshick, R.B., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. In: NIPS, pp. 91–99 (2015)
3. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C.J.C., Bottou, L., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems 25, pp. 1097–1105. Curran Associates, Inc. (2012)
4. Mueller, J.P., Massaron, L.: Machine Learning for Dummies, 1st edn. For Dummies (2016)
5. Chollet, F.: Deep Learning with Python, 1st edn. Manning Publications Co., Greenwich (2017)
6. Liu, L., et al.: Deep learning for generic object detection: a survey. CoRR abs/1809.02165 (2018)
7. Nguyen, T.A., Csallner, C.: Reverse engineering mobile application user interfaces with REMAUI (T). In: 2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE), pp. 248–259. IEEE (2015)
8. Canny, J.F.: Finding edges and lines in images. Technical report, Massachusetts Inst of Tech Cambridge Artificial Intelligence Lab (1983)

9. Mozgovoy, M., Pyshkin, E.: Using image recognition for testing hand-drawn graphic user interfaces. In: 11th International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM 2017) (2017)
10. Chen, C., Su, T., Meng, G., Xing, Z., Liu, Y.: From UI design image to GUI skeleton: a neural machine translator to bootstrap mobile GUI implementation. In: Proceedings of the 40th International Conference on Software Engineering, pp. 665–676. ACM (2018)
11. Liang, M., Hu, X.: Recurrent convolutional neural network for object recognition. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2015
12. Bradski, G.: The OpenCV library. Dr. Dobb's J. Softw. Tools **25**, 120–125 (2000)
13. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: International Conference on Learning Representations (2015)
14. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778 (2016)
15. Liu, L., Özsu, M.T. (eds.): Mean Average Precision, pp. 1703–1703. Springer, Boston (2009). https://doi.org/10.1007/978-0-387-39940-9