IJARIIT INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

ISSN: 2454-132X
Impact Factor: 6.078
(Volume 7, Issue 3 - V7I3-1699)
Available online at: https://www.ijariit.com

Automation of UI design testing using Machine Learning

Divya Gupta
guptadk19.comp@coep.ac.in
College of Engineering, Pune, Maharashtra

Jagannath Aghav
jva.comp@coep.ac.in
College of Engineering, Pune, Maharashtra

Ashwini Chaudhari
ashwini.chaudhari@emerson.com
Emerson Innovation Center, Pune, Maharashtra

ABSTRACT

The buzz around SDLC and the ever-increasing need for testing each analytical component of a software component has incurred time losses in the entire cycle. The flaws in the design of any UI hurt the presentation and ease of access to users. Manually testing these entities might end up adding complexity, manual errors and unnecessary duration. Taking into consideration the dynamic elements of UI, this work presents a study on how the testing of these design elements can be carried out using different methods. It also addresses the technical challenges in the tasks carried out for the detection of objects and widgets in these dynamic interfaces. Motivated by machine learning concepts like Regions of Interests this work sheds light on the limitations and highlights of these methods.

Keywords— *Object Detection, CNN, Preprocessing, UI*

1 INTRODUCTION

Along with the surfacing of World Wide Web and the outbreak of technology and the ever-increasing buzz around the development cycle of any software (SDLC), software testing became a major concern. The testing phase being such a significant part in any development is been categorized broadly into GUI based testing, integration testing and logical testing. Testing the GUI provides a more intricate path to interact with the software. The amount of time and complexity that goes into testing any UI has been increasing over time exponentially. The testing thus needs to be carried out in a way that it offers efficiency, effectiveness, good path coverage and an increased fault detection rate.

GUI being the front end of any software needs to be convenient to access and track along with being user friendly. Every organization thus creates a basis called their standard guidelines as per the requirement and ease of clients/users of their products. Along with all the processing that goes in the back end the aesthetics and ease of front end is equally important and sometimes overlooked at. GUI testing not only confirms the standardized format of the interface but also adds a layer of additional corroboration for developers. Automating this process is a smart option for the duration margins of software development and it's testing to be in proportion.

2 BACKGROUND

'Image classification' consists of activities like prediction of an object class in an image. Pinpointing the location of objects in an image and sketching an abounding box in the circumference of its extent is done by 'object localization'.

Object detection basically combines the functions of these two methods and collectively is localized and a certain number of objects in the image are classified. Whenever a practitioner or user refers to the word 'object recognition', it usually means 'object detection'. [4]

It is possible to roughly break down the entire process of object detection into 3 methodologies of which the first method is extracting features locally wherein local models such as 'Histogram of Oriented Gradients' and 'Scale invariant feature transform' are used. Further a classifier is constructed to detect exact items.[6] The second method takes into consideration deformable part models techniques where a suitable method gives information about the location of the components of the items which useful for training purpose. In the same way, the third category is called Convolutional Neural Networks (CNN) wherein the attributes of images are defined by feature vectors which are consecutive. CNNs depict a higher degree of precision when it comes to object boundary detection or other such features.

At a basic level there are two broadly classified approaches for object detection models namely top-down approach and bottom-up approach. As a result of the features being extracted and matched locally, the number of false positives suffered in top-down approach is more. In the same way there is a necessity of some effective techniques for segmentation in the approaches which follow the bottom-up pattern. Therefore, a third and better approach is created which merges the above-mentioned approaches in order to preserve the consistency and balance in segmentation tasks which in turn reduces the tasks of grouping.

Color image segmentation methods can be viewed as an extension of the gray image segmentation method in the color images, but many of the original gray image segmentation methods cannot be directly applied to color images. We analyzed proposed a color image segmentation method of automatic seed region growing on basis of the region with the combination of the watershed algorithm with seed region growing algorithm which based on the traditional seed region growing algorithm.

2.1 Diagnosed Techniques

2.1.1 R-CNN: For finding a solution to the issue of choosing a large number of regions while processing an image, Ross Girshick put forth a method which uses the selective search in order to fetch just 2000 regions from the image and he called them region proposals. Thus, instead of trying to classify the huge number of regions, 2000 regions are worked upon. The extraction of these 2000 region proposals is done by the following algorithm called selective search. [3]

Selective Search :

- Generating the first sub-segmentation
- Using the greedy algorithm to recurrently combine similar regions into wider ones
- Using generated regions to build the final concluding region proposals.

The two thousand candidate regions which are also termed as proposals are twisted into a square and given as input to a CNN which outputs a feature vector of dimension 4096. Here the role of CNN is like a feature extractor and the dense layered output comprises of the extracted features from the image and these features which are extracted are given as an input to a Support Vector Machine for classifying the availability of the object inside that candidate region proposal. [3]

2.1.2 Fast R-CNN: Ross Girshick later developed an algorithm to get something faster by avoiding the shortcomings of R-CNN. This method is identical to the 2 R-CNN algorithms. However, rather than submitting regional proposals to CNN, the input image is directly fed to CNN in order to generate a convolution feature map. We can recognize the region of the proposals from the convolutional feature map and bend them into square. Further with the help of an RoI pooling layer we rearrange them to a fixed size to be used as an input to a fully connected layer. It is possible to use the softmax layer from the feature vector or the RoI indicator so that we can predict the class of proposed region and the offset value for the bounding box. [3]

2.1.3 YOLO: YOLO first takes a picture and later splits the image into an SxS grid. Inside each grid it takes m bounding boxes. Within each bounding box, the network offers an offset value and a class probability for the bounding box. A sole convolutional network thus simultaneously predicts several bounding boxes and class probabilities for those boxes. The bounding boxes which have class probability above a pre-established threshold value is selected and used to detect the object inside the image. YOLO not only understands the generalized object representation but also works faster as compared to other methods (45 frames per second).

One of the constraints of YOLO algorithm is the detection of tiny objects in the image, for example, it will be difficult for YOLO to identify a swarm of bees or a herd of birds. It is because of the constraints on the spatial features of the algorithm. [3]

2.2 Object Detection for GUI

It is domain specific job in object detection to identify GUI elements in GUI images. It is possible to recognize elements in GUI using intrusive methods or pixel-based (non-intrusive, generic, domain-specific) methods. The elements in any UI can be roughly classified into text elements and non-text elements. Detecting text and non-text UI elements using a sole prototype has a worse performance as compared to using models of non- text components and text components dedicated separately. The text present in any UI must be considered as scene text and performed upon rather than considering it as document text.

2.2.1 Non-text Detection: For region detection and classification of UI elements various traditional and non-traditional methods exist of which Faster R-CNN and YOLO v2/YOLO v3 work considerably well. These are the anchor box methods which work by generating bounding boxes from the predefined anchor boxes. CenterNet being an anchor free, single stage object detection technique predicts the positions of bottom-right and top-left corners along with the center of objects in an image and later merges them to obtain the bounding box of those objects. Thus, it does not generate bounding box based on anchor box which are decided priorly. [1]

2.2.2 Text Detection: The fact that UIs are inclined towards scene text instead of document text, the traditional methods for detecting characters mostly don't seem to work effectively. One of the OCR tools called Tesseract is created for texts in documents. It generally flows in two actions : 'text line/boundary detection' and 'text recognition'. For this project, the only part which is relevant is the former part i.e. the text line/boundary detection. The text line detection carried out by Tesseract is an out of fashion methodology. First it transforms the image that is given into a binary map then later executes a connected component analysis in order to find the borders of each element in the image. The outlines thus produced are then clustered into blobs. These blobs are further joint together. At last, it integrates all those text lines which overlap at least half horizontally. [1] A technique in deep learning called EAST is used to recognize text in natural scenes. Firstly, an image is given as an input to a feature pyramid grid. Then, EAST calculates 6 values related to each point bottomed on the finalized feature map. These values include : a rotation angle, bottom/top/right/left offsets, an objectness score etc. In favor of this as a baseline, the pre-trained model is used directly to perceive texts in UIs irrespective of any adjustment or calibration. [5]

2.3 Motivation

UI testing gets tedious to a high extent as the development code keeps changing throughout the SDLC. The manual effort that goes behind the testing and updating of UI to provide an optimal framework decreases the overall efficiency and increases the delivery cycles of a software product. Manually testing the elements of GUI is also tiresome, boring and thus error prone. Automating this exercise will considerably reduce the effort of processing these test cases by providing an intelligent test design.

3 METHODOLOGICAL APPROACHES

The implementation of this work started with understanding the 3 broad dimensions of this project :

- UI image Classification
- Object Detection of the UI elements and Text Detection

- Test case execution

Any suitable testing requires a test case and its generation mechanism. One of the suitable tests with fewer infeasibilities described by Xun Yuan is the Dynamic Adaptive Automated test generation. In due consideration of the dynamic behavior based on various states of UIs, it is essential and important to create test cases which will be based on feedbacks and have a thorough review. [2]

3.1 Object Detection of the UI elements

This phase started off with recognizing the dynamic elements of the device UI. The number, area and layout of some elements differed with a change in the device type of the organization. The overview page which is the default startup page of devices is divided into two sections which is the A-level menu and the B-level menu. The object detection is to be carried out in the B-level menu which contains tabs and graphics giving information about devices. The tabs are clickable boxes and graphics just display specific measurements/status of devices for which the UI is being made.

The region detection of these graphics and text boxes was carried out by different methods as mentioned below :

3.2 ROI extraction by color filter

Extracting regions of interest from images based on colors using masking is a great technique to identify areas of specific colors. In our B-level menu there are graphics with RGB color spaces. The first approach towards detecting these graphics was to use these color filter masks like green screens and extract these areas. By using Morphological transformation, the unnecessary edges were removed. Instead of RGB values the hue and saturation values help in pinpointing the actual area of interest in this approach.

Challenge : The main challenge in using this method was the complexity of using different color masks for different graphic box. Also, this method was not quite inclusive of the text box area (tabs) in the UI.

3.3 ROI extraction using coordinates

This method works by giving the model a standard set of input coordinates which enclose the regions of interest of the UI. The model in turn extracts these regions and give it as an output to the further task. This can be done either by using center coordinates of the object or by the top-left and bottom-right coordinates.

Challenge : The elements of UI don't follow a standard positioning system and the number of tabs along with their location keeps changing as per the device for which the UI is being made. Using coordinates for detecting and extracting these regions is not the right way to do it as many chunks of region might go unnoticed by the model.

Ideation : We also tried using a combination function for grouping these coordinates according to the type of devices. The set of coordinates corresponding to a device UI was grouped and given to its corresponding conditioning loop. But this created a lot of confusion and cost.

3.4 ROI using Key points

Some techniques to find key points for ROI extraction as SIFT, SURF, ORB. Most of these tools are not open source so we used the ORB technique. ROI feature extraction is done by this method by max pooling.

Challenge : The accuracy of this method is too low for the kind of dataset we are working on.

3.5 ROI using rectangular contours

The conclusion that was settled on was the fact that our regions of interest were all following the same shape patter i.e. rectangular. We started extracting regions of graphics and tabs by using different areas, height and width of rectangle. Since the rectangular areas were very well defined, this method worked well.

Challenge: This method started considering text areas as ROIs. So, to exclude this we came up with the following ideation.

Sliding Window: Use a sliding window for parsing the image and extracting the rectangular regions by excluding the text areas or headings.

4 PROPOSED METHODOLOGY

With a CNN model in place having two convolution layers and additional activation layers (ReLU and sigmoid) and max pooling layers, the proposed approach starts with classifying the input UI image based on the protocols for which the organization creates devices. These protocols are namely Hart, WiHART, Profibus, FF depending upon the functionalities. To compensate against the unavailability of a large dataset in order to have higher accuracy, the model takes help of data augmentation approach taking into consideration the rescaling and sheer range factors. This helps in making the dataset balanced by creating variations of original samples in the dataset.

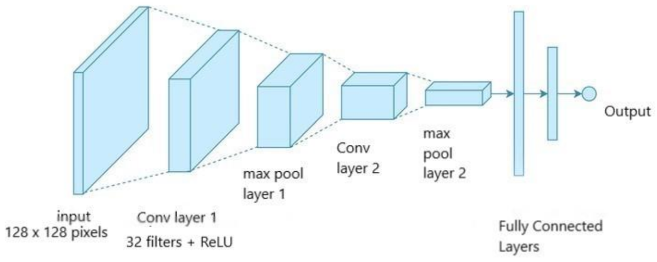
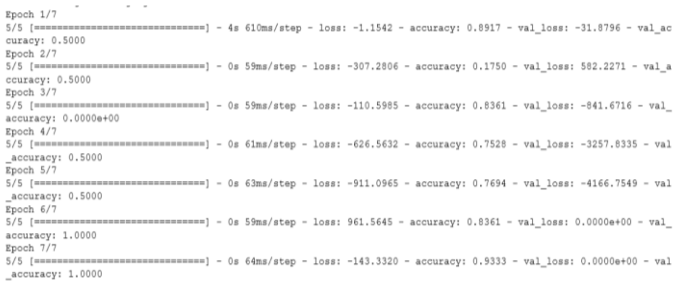


Figure 1 : Model representation

The validation accuracy of this model reaches a good 100% with the training accuracy touching 93% within seven epochs.



```
Epoch 1/7
5/5 [==============================] - 4s 610ms/step - loss: -1.1542 - accuracy: 0.8917 - val_loss: -31.8796 - val_ac
curacy: 0.5000
Epoch 2/7
5/5 [==============================] - 0s 59ms/step - loss: -307.2806 - accuracy: 0.1750 - val_loss: 582.2271 - val_a
ccuracy: 0.5000
Epoch 3/7
5/5 [==============================] - 0s 59ms/step - loss: -110.5985 - accuracy: 0.8361 - val_loss: -841.6716 - val_
accuracy: 0.0000e+00
Epoch 4/7
5/5 [==============================] - 0s 61ms/step - loss: -626.5632 - accuracy: 0.7528 - val_loss: -3257.8335 - val
_accuracy: 0.5000
Epoch 5/7
5/5 [==============================] - 0s 63ms/step - loss: -911.0965 - accuracy: 0.7694 - val_loss: -4166.7549 - val
_accuracy: 0.5000
Epoch 6/7
5/5 [==============================] - 0s 59ms/step - loss: 961.5645 - accuracy: 0.8361 - val_loss: 0.0000e+00 - val_
accuracy: 1.0000
Epoch 7/7
5/5 [==============================] - 0s 64ms/step - loss: -143.3320 - accuracy: 0.9333 - val_loss: 0.0000e+00 - val
_accuracy: 1.0000
```

Figure 2: Model Parameters

Once the images are classified based on their protocols, they are put through pre-processing. The images are converted to grayscale representations to store less pixel information. Further these grayscale images are passed through functions of thresholding, morphing and dilation. Once the image is processed into a binary transform, extraction of contours takes place. The ROI is extracted based on rectangular contours using OpenCV python. This extraction takes place in accordance to a scale based on the size of varied elements in the UI. These contours are then sorted based on their position on the UI. The contours are stored for extracting the text out of it and detecting the same based on the English language diction.

The sorted contours are also matched with a set of standard graphics that need to be present on the UI based on different protocols. These graphics are matched using Key Points and Brute Force Matching Algorithm.

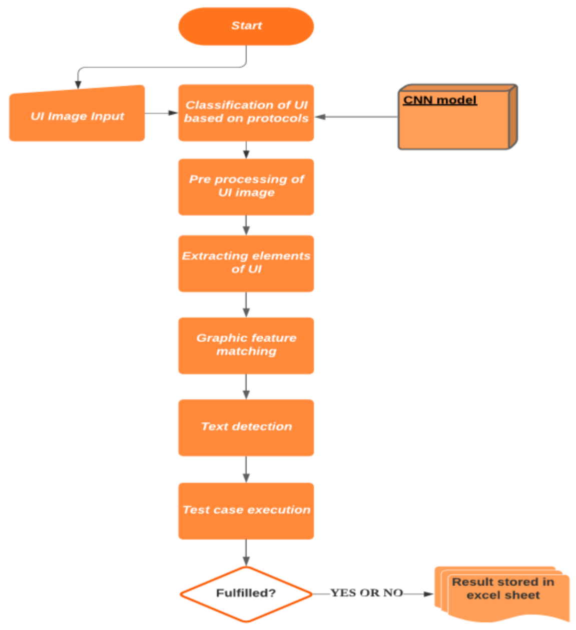

Start
UI Image Input
Classification of UI based on protocols
CNN model
Pre processing of UI image
Extracting elements of UI
Graphic feature matching
Text detection
Test case execution
Fulfilled?
YES OR NO
Result stored in excel sheet

Figure 3: Proposed Methodology

The test case execution of various standard guidelines of the organization are done to detect any kind of flaw in the design of the UI which does not follow the guidelines mentioned.

Finally, the results are stored in an excel sheet for easy access to the development team.

5 CONCLUSION

The design testing of any UI mainly requires the correct extraction of the variable elements of an UI. Creating a tool to do so makes it easier to automate this process which in turn is advantaging to the Software Development Life Cycle. Machine learning adds dynamism and a self-learning capability to the idea of automation. This work can also be expanded to other similar protocols and UI designs.

6 REFERENCES

[1] Isabella and Emi Retna. Study paper on test case generation for GUI based testing. ABC, 2012. URL https://arxiv.org/ftp/arxiv/papers/1202/1202.4527.pdf.
[2] Jieshan Chen, Mulong Xie, Zhenchang Xing, Chunyang Chen, Xiwei Xu, Liming Zhu and Guoqiang Li. Saliency guided faster-rcnn (sgfr-rcnn) model for object detection and recognition. JKL, 2020. URL https://arxiv.org/abs/2008.05132.
[3] Maxime Oquab, L´eon Bottou, Ivan Laptev, Josef Sivic. Weakly supervised object recognition with convolutional neural networks. HAL archive, 2014. URL https://hal.inria.fr/hal-01015140v1/document.
[4] Mukkamala Rohith Sri Sai and Sainagesh Veravalli. Object detection and identification, a project report. GHI, 2019. URL https://www.researchgate.net/publication/337464355_OBJECT_DETECTION_AND_IDENTIFICATION_A_Project_Report/link/5dd8c7bd92851c1feda8e2fd/download.
[5] Vipul Sharma, Roohie Naz Mir Department of Computer Science & Engineering, National Institute of Technology, Srinagar, India. Object detection for graphical user interface: Old fashioned or deep learning or a combination? Journal of King Saud University – Computer and Information Sciences, 2019. URL https://doi.org/10. 1016/j.jksuci.2019.09.012
[6] William Hasling and Rajesh Subramanyam. Automation of gui testing using a model driven approach. DEF, 2006. URL http://users.csc.calpoly.edu/~gfisher/ work/specl/documentation/related-work/9jul08-scan/p9-vieira.pdf.
[7] Yuhang Zhang, Hao Sun, Jiawei Zuo, Hongqi Wang, Guangluan Xu and Xian Sun. Aircraft type recognition in remote sensing images based on feature learning with conditional generative adversarial networks. MDPI, 2018. URL https://www.researchgate.net/publication/326427598_Aircraft_Type_Recognition_in_Remote_Sensing_Images_Based_on_Feature_Learning_with_Conditional_Generative_Adversarial_Networks