Performing Automated UI Test Using Convolution Neural Network, Recurrent Neural Network, and Optical Character Recognition

Subas Thapa

Abstract

Convolution Neural Network(CNN) has emerged as one of the most powerful deep neural network in the field of image processing and recognition. I aim to tackle this solution by using artificial intelligence algorithms along with automated tools to detect elements. Captured screenshots of the Graphical User Interface(GUI) are passed on to a Convolution Neural Network (CNN), Recurrent Neural Network (RNN) and Optical Character Recognition (OCR). CNN consists of convolution layers, pooling layers, and dense layers. These interconnected layers identify patterns to recognize objects in an image. The convolution layer extract features from the input image while the pooling layer reduces the size of the feature map. The dense layer is also known as hidden layer which consists of nodes connected to a subsequent hidden layer. Sequences of inputs are memorized using a Recurrent Neural Network. A dataset, consisting of icons and buttons is created to train the CNN. Text in the GUI is extracted from image using OCR. The aim of this project is to create a system to detect an element in the GUI with minimal effort.

I. Introduction and Motivation

Users interact with the software through a Graphical User interface(GUI). GUI is the key component of software, and it should be user-friendly. Testing and maintenance need to be performed in order to improve the user experience of software. GUI testing is one of the main challenges in software industries where requirements change frequently. As a consequence, testing has to be performed rigorously to make software bugs and error-free. Manual and automated tests are performed in order to find bugs in a GUI. Manual testing consumes time and resources in most companies. Some visual bugs might not be captured during the manual tests. Automated tests reduce the time and resource needed during the software development process and reduces human errors too. However, automated tests also consume more time in maintaining test cases due to frequent changes in user interface and dynamic properties of a webpage. This report will present a solution for problems using a Convolution Neural Network (CNN), Recurrent Neural Network (RNN), and Optical Character Recognition (OCR). We feed screenshots of a webpage to the CNN, RNN, and OCR algorithms in order to identify the elements in a given GUI.

II. Background

This report consists of a detailed review of how automated tests can be performed to test a graphical user interface. Hubel (1957) was successful in presenting crucial insights into the structure of the visual cortex[1]. Hubel and Wiesel (1958) were successful in demonstrating that most of the neurons in the visual cortex contain a small local receptive field, and respond to some specific patterns[2][3]. In order to construct the whole visual field, receptive fields of various neurons may overlap. They also found some neurons react to the horizontal line while others react to the vertical line in an image[2][3]. Additionally, they noticed that some neurons have larger receptive fields, and can identify complex patterns [2][3]. The observation further concluded that higher-level neurons are related to the output of lower-level neurons and are able to identify the complex pattern of any visual field [2][3]. Kunihiko Fukushima (1980) proposed a neural network model, Neocognitron, a self-organized network, inspired by a visual nervous system by Hubel and Wiesel[4]. The purpose of this system is to recognize a stimulus pattern based on Gestalt(Geometrical similarity)[4]. This system consists of multi-layered structure and is capable of unsupervised learning[4]. Sets of stimulus patterns are repeatedly fed to the input layer, and this module acquires a structure of hierarchical visual nervous system [4]. LeCun et al.(1998) presented a LeNet-5 architecture, used to recognize a pattern in a document[5]. This famous architecture has building blocks like fully connected layers, and sigmoid functions, and introduced two layers Convolution and Pooling [5]. Eskonen et al. (2020) have published an approach to image-based deep reinforcement learning to test GUI applications where a Convolution Neural Network is used to extract features from images[7]. In order to memorize the sequence of inputs, a Recurrent Neural Network is used [7]. Long short-term memory (LSTM) cells made of gates are used to learn, classify and process the input [7]. Elements on the GUI are categorized into two groups text and non-text elements [8]. Text elements on a page are detected by Optical Character Recognition (OCR) while icons and buttons are detected using CNN [8]. Text in a GUI are detected using state-of-the-art deep learning scene [9].

III. Research aims and questions

Due to frequent changes in requirements, updating test cases is laborious. Additionally, due to the dynamic nature of elements in a webpage, properties like ID and Name are dynamically updated each time a page is loaded. As a result, locating elements

in a webpage becomes tedious and consumes more time and resources to update test cases. In this paper the following issues will be explored and solutions for the following problems listed below, found:
i. How can elements be detected using a screenshot of the graphical user interface using CNN?
ii. How dynamic properties of an element can be tackled so that elements are detected without using locators of an element?
iii. How can we manipulate images of the User Interface to detect text in a GUI?
iv. How can we reduce the test maintenance time of test cases?
V. How can elements in a GUI be grouped based on type?

IV. Research Methodology

A. Research Plan

To answer the questions listed above, research will be conducted in the Salesforce, customer relationship management system. Due to multi-tenant architecture, web-page and contents are dynamically loaded, which makes the platform ideal to test our proposed system. Automated test will be conducted on the Sales app provided by Salesforce.

Phase I: Analyse the requirements. Collect, research papers, existing algorithms, and proposed designs.
Phase II: Design the system using design documents.
Phase III: Develop the designed system using proposed algorithms and libraries.
Phase IV: Create a dataset consisting of icons and buttons. Prepare test data.
Phase V: Test the developed application and record accuracy and loss on validation and training dataset.

B. Data Security, and Privacy Plan

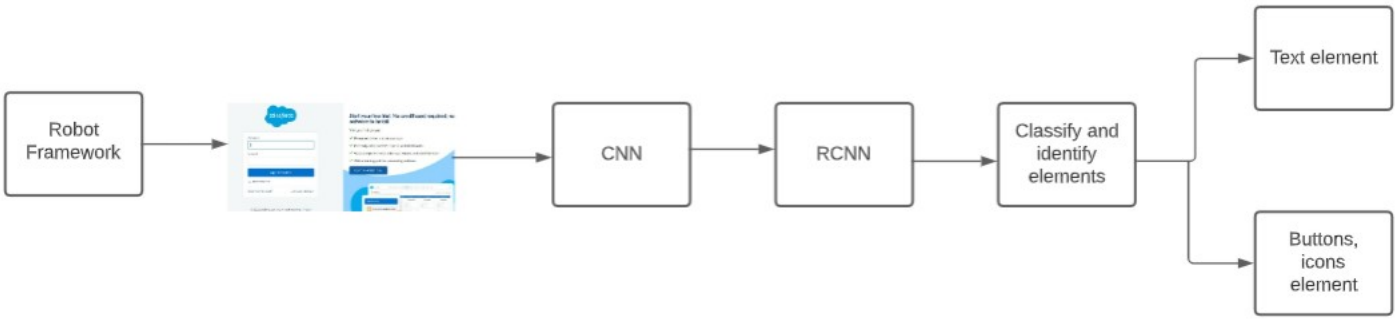
Data generated by performing automated test are not related to any person. Privacy and security policies imposed by Salesforce will be maintained throughout the research period. The dataset consisting of icons, and buttons will be created and will be used by CNN in the later phases.

C. Model Training Plan

The dataset for this module will be created with several classes. Training and test images will be divided so that the test data and training data will cover all classes. In order to avoid overlap, classes are selected in a mutually exclusive way.

D. Proposed Design

The below diagram shows the proposed design of our system. We use the Robot framework, which uses webdriver internally to perform an automated test in browser. Opening the browser, and navigating to the webpage are performed using webdriver.



E. Tools, Libraries and Programming Languages

Tensorflow
Robot Framework
Selenium Webdriver
Python-tesseract
Python

V. Summary of Expected Outcomes

A system to verify elements on a GUI using CNN, RNN and OCR will be developed. CNN is a special kind of neural network having multiple layers which extracts features from images. LSTM network, application of RNN, is used to recognize the patterns. A dataset consisting of buttons, and icons will be created. The proposed system will be trained using CNN to identify whether the element is present on a webpage or not without writing locators for text, buttons, icons and images. Elements on a webpage are grouped into two types: text and non-text. Text on the webpage is identified by OCR. This reduces the dependencies with web element's dynamic properties so that test case maintenance time and resource is reduced.

VI. Conclusion

In this proposal report, system to perform GUI test of web application has been presented. This system works along with Selenium, an open source framework for web application testing. Selenium will take the screenshot of the web-page and the screenshot is then processed by CNN, RNN, and OCR to verify elements are present in the given web-page. This will reduce the cost of test development and maintenance. As a result, time and resources will be reduced. Additionally, it will increase the test coverage and reduces the chance of getting visual bugs in the production environment.

References

[1] D. H. Hubel, *Single Unit Activity in Striate Cortex of Unrestrained Cats, 1957*

[2] D. H. Hubel and T. N. Wiesel, *Receptive Fields of Single Neurones in the Cat's Striate Cortex, 1958*

[3] D. H. Hubel and T. N. Wiesel, *Receptive Fields and Functional Architecture of Monkey Striate Cortex, 1958*

[4] Kunihiko Fukushima, *Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position, 1980*

[5] Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner, *Gradient-Based Learning Applied to Document Recognition, 1998*

[6] Aurélien Géron, *Hands-on Machine Learning with Scikit-Learn,Keras TensorFlow, 2nd ed,* O'Reilly, Canada, 2019

[7] Juha Eskonen, Julen Kahles, and Joel Reijonen, *Automating GUI Testing with Image-Based Deep Reinforcement Learning*

[8] Mulong Xie, Sidong Feng, Zhenchang Xing, Jieshan Chen, and Chunyang Chen, *UIED: A Hybrid Tool for GUI Element Detection*

[9] Xinyu Zhou, Cong Yao, He Wen, Yuzhi Wang, Shuchang Zhou, Weiran He, and Jiajun Liang, *EAST: an efficient and accurate scene text detector,* 2017, IEEE conference on Computer Vision and Pattern Recognition