International Journal of Grid and Distributed Computing
Vol. 14, No. 1, (2021), pp. 1304 – 1310

Implementation of Machine Learning Approach for Designing an Automated Front-End

Kanchan S. Gawande[1], Prof. K. V. Warkar[2], Prof. A. D. Gotmare[3]
[1]PG Student, Department Of Computer Engineering Department of Computer Engineering, Bapurao Deshmukh College of Engineering, Sevagram, Wardha Sevagram, Wardha.
[23]Assistant Professor, Department Of Computer Engineering Department of Computer Engineering, Bapurao Deshmukh College of Engineering, Sevagram, Wardha Sevagram, Wardha.

Abstract

Starting the design process for a website by constructing detailed mock-ups of various web pages using either visual or manual computerization and then developing the mock-up tools are the first steps. Developers use mock-ups to develop organised HTML or related labelled code, then they move them into HTML. Developers are commonly expected to provide a web page mock-up, which they then translate into actual code. The process typically has to be done several times on numerous additional occasions till the ideal design is produced. Tragically, this operation is laborious and time-consuming. This study introduces a way to automate the code creation from mock-ups, resulting in less time and being significantly cheaper. One or two profound learning techniques are employed to bring the framework to fruition.

Keywords: *HTML, Mock-ups, Machine Learning, Dataset, Convolution Neural Network.*

1. Introduction

It is critical that Internet websites continue to grow and advance, because of how far behind we are with current technology. These days' sites have a clear connection to the states, organisations, networks, persons, and so on. There is no way to stop the constant change of the Information Technology industry. More recent engineering products, such computers, cell phones, laptops, and even vehicles, continue to show significant improvements in the aspects of usefulness, technical support, and outcomes quality. On the other side, the expectation is that the official organisations would provide increasing productive sorts of aid. The present day user-facing software programming apps are all Graphical User Interface (GUI) driven, and they employ UI lures to draw users in (UI). The front-end of each site is a “web page”, which is the section of the site that engages and holds the attention of end users. It is absolutely necessary to present a page that appeals to the end user, however, this is quite tough to do and there is significant value in the pages that do this. In any event, working on pages that are flexible in their reactions to these demands is a challenging task. Web page creation necessitates the cooperation of a wide range of specialists, ranging from graphic designers, software specialists, end-users, and company staff to employees who work in a variety of locations. The approach often begins with the visual designers mimicking the structure of the UI by upstaging their counterparts, the structure of the UI then fits the business's requirements.

Because these proposals rely on existing draughts, programming professionals write code for the internet pages that depend on these proposals. Web sites created by companies advance for monetary justifications, with the primary objective of getting and presenting products in the public eye. In order to meet the needs of our clients, we are always looking to update the content of our website. This is a very labor-intensive

ISSN: 2005-4262 IJGDC

technique that involves tonnes of mundane tasks. Segments retain their similar functions, and as long as there is a continuing change in page structure, we find the process of rewriting code to be quite tedious. Many startups build software prototypes to exhibit their ideas and convince investors that the concept would also greatly benefit by demonstrating an early iteration of the application. I feel that instead of wasting time and money on over and over designing and developing the UI, a more precise automated approach would be received with much more positivity. By allowing small firms to prioritise component development and greater value and reducing emphasis on interpreting structures into practical application code, this will allow them to devote more resources to these areas. Ensure to avoid the pitfalls that might affect front-end engineers and front-end originators by constructing intricate GUI's, since this shows that there is still a high degree of requirement for improved UI resolution in web page design.

The designing of web pages by means of automated code generation is gaining increased interest as a research area. The decrease in programming time, process cost, and source utilisation due to automatic web page production is referred to as "automated web page production." Since the dynamic site-redesign plan incorporates the rapid development, the finished project is completed in a reduced amount of time. The HTML code for the mockup of a website page has been automatically generated by creating a methodology for doing so. To encode these components into the mock-up photos, it is expected that the product designer will see them as well as comprehend the hierarchy on the website. Using the dataset from the Pix2Code study, we can bootstrap websites using our newly generated dataset. Bootstrap has been an invaluable tool for reducing the amount of CSS and HTML needed when working with our clients, and has contributed greatly to the expansion of our professional vocabulary. To achieve better picture results, a deep neural network including Convolutional Neural Networks (CNN) is used to train the pictures used in the training data sets.

2. Literature Review

The design cycle for a web site often begins with the production of preliminary mock-ups for individual web pages, whether those mock-ups are done by hand or through specialist mock-up production tools. Once the mockup has been developed, it is rendered into a structured code or comparable markup by software programmers. Once the necessary template has been produced, the procedure is repeated several times until it reaches the required consistency. In this work [1], the goal of the author is to automate the code creation process from hand-drawn mock-ups. Computer vision techniques are utilised to process mock-ups hand-drawn in the design phase, and afterwards deep learning methods are implemented in order to build the suggested system. The system can attain a high level of method accuracy and a moderate level of validation accuracy.

It is typical practice for software engineers who work on the user-facing side of the product to translate a mock-up of a graphical user interface (GUI) into source code. It is in effect a life cycle for an application, including stages that begin before the programme is created and go until features are made that affect the interface. Unfortunately, it is time-consuming and difficult to follow this practice. The authors in this work offer a system that automates this process by making GUI prototyping a seamless job. In order to do this, the technique relies on three individual tasks: detecting, classifying, and assembling. After this first stage, it is next determined if the logical GUI components can be discovered from a mock-up artefact using computer vision methods or mock-up metadata. This procedure, which involves a software repository mining operation, automated dynamic analysis, and deep convolutional neural networks, is then used to properly categorise GUI components into specified domain categories (e.g., toggle-button). The approach finally used was one which was data-driven and which used K-nearest-neighbors to produce a GUI prototype from which an application could be automatically constructed. As an

ISSN: 2005-4262 IJGDC

example, the author built this methodology for Android in a system called ReDraw. The information provided in our assessment is that ReDraw is capable of classifying each GUI component with an accuracy of 91%, as well as assembling prototype programmes that closely mimic the desired mock-ups in terms of visual affinity, while demonstrating adequate code structure. While conducting interviews with industry practitioners, it was discovered that ReDraw can help enhance the processes of companies developing real-world solutions.

REMAUI, an algorithm designated for reverse engineering the user interfaces in mobile applications, is used to obtain the elements of a UI, for example, buttons, text-boxes, and photos. By reverse engineering the UI, REMAUI produces the code for the elements from the screenshots of the UI window. As PC vision and optical character acknowledgment techniques are used, it converts to the code for screen images or drawings for mobile platforms, mobile platforms screen pictures and drawings, and screen captures for PC vision. Despite the fact that the REMAUI technique works efficiently, it does not aid with cross-page changes and animations within the page. The P2A algorithm was developed by creators in order to meet the inadequacies of the REMAUI computation.

Although they established the pix2code method, which aims to shift the graphical interface of a website page to structured code using deep learning with convolution and repeated neural networks, the creators enhanced it to operate better with the structure of a webpage [4].

Redraw's method (which includes drawing new screen mock-ups) creates XML files that provide standardised screen definitions for all app screens. The primary objective of PC vision technology in the early stages of installation is to differentiate GUI components on the screen. The second step involves the sequence of the identified components, such as a toggle button, textbox, and so on, in accordance with their functions. Convolutional neural networks are in use at the moment. In the penultimate phase, the XML code is created by combining the K-Nearest Neighbour's (KNN) method, as listed in the programming hierarchy on the web, to generate the final result. For instance, the code libraries that are prominent at the moment, such as GitHub, are commonly used to exchange code and apps these days. Many developers consider this repository to be a good place to look for code to reuse when beginning a new programming effort. The numerous common codes in these libraries decreases the likelihood of an identical code being created by distinct persons over and over. The authors use an inquiry tool called SUISE in which the customers provide basic illustrations and phrases to describe a simple interface with only a few controls [6]. Then the interface attempts to search for already existing libraries in order to obtain further interfaces. These interfaces are built into operable programmers and are provided back to the end client, where the client has the ability to pick the most reasonable interface.

3. Proposed System

The research was carried out in four phases. Object detection was applied to the input picture in the first stage, using image processing techniques including erosion, dilation, and contour detection. The recognised objects were subsequently cropped, and the resultant components were labelled with the trained CNN model. Finally, the HTML Builder script was used to transform the model's output to HTML code. Figure 1 depicts the suggested method.

ISSN: 2005-4262 IJGDC

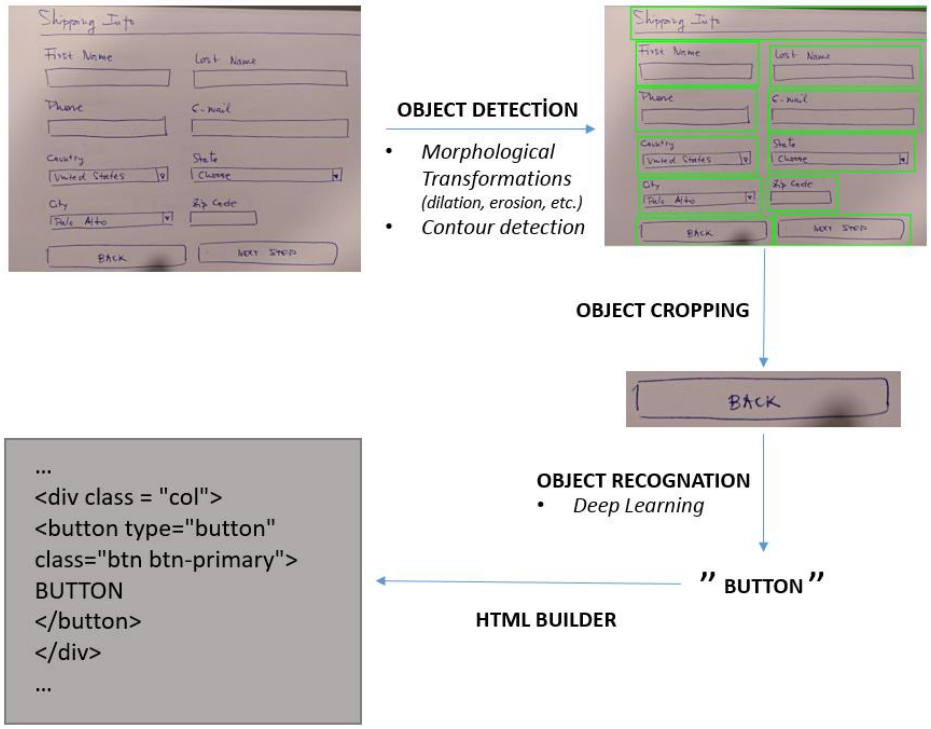

Figure 1. Proposed Algorithm

3.1 Detection and Cropping of Objects

The input file is converted to grey scale format once it has been read. Then, using a 3x3 rectangle kernel, Gaussian Blur was applied to them twice. Following the thresholding procedure, rectangles were formed using the contour detection technique to determine the objects using morphological alterations. The components of the supplied image have been recognised in this way. To transfer the discovered components to the CNN model, they were cropped. Figures 2 and 3 depict the output of this stage.

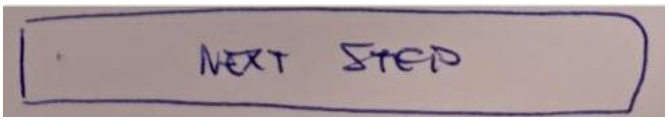

Figure 2. Cropping of Object

The 8 iteration dilation with a 4x4 rectangle kernel was used in the phases of morphological alterations. Then, for four iterations, a 3x3 ellipse kernel was used for erosion, and for two iterations, a 1x10 rectangle shaped kernel was used for dilation. Finally, for the dilation procedure, a 10x1 rectangle shaped kernel was employed.

ISSN: 2005-4262 IJGDC

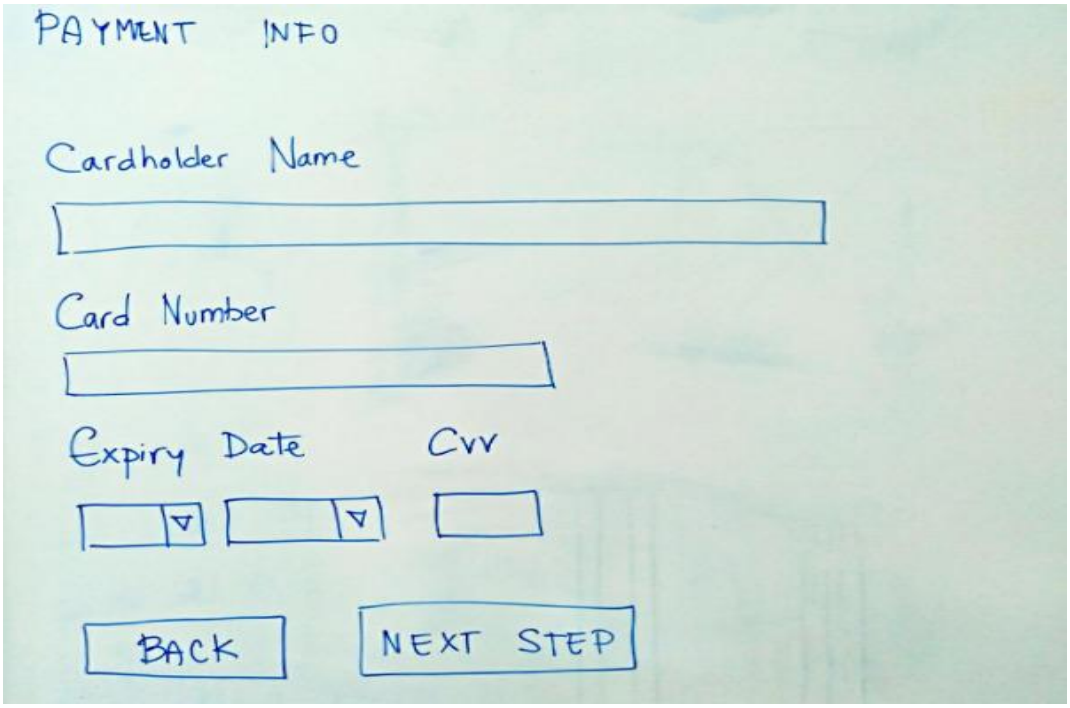

Figure 3. Object Detection Process

3.2 Recognizing Objects

The components in our component dataset were used to train the model displayed in Fig. 5. It comprises of four various sorts of components, including textboxes, dropdowns, buttons, and checkboxes, as previously mentioned. The loss function was trained for 200 epochs using Binary Crossentropy and RMSProp methods with a batch size of 64 after the model was learned. After that, the cropped components from the previous step were used as input for the component recognition method.

We used many convolution layers with 4x4 kernels and subsequently conducted max pooling procedures with 2x2 kernels for feature extraction, as shown in our CNN Model in Fig. 4. Following the feature vectorization procedure, the BiLSTM layer is used to catch correlation of the retrieved features. After all, in order to reach the categorization goal, we used a 20 percent ratio of Full Connected Layers to Dropout Layers.

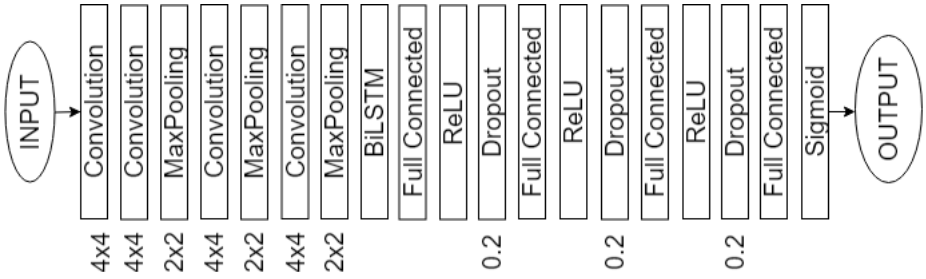

Figure 4. Model C of CNN. HTML Editor

The bootstrap framework was used to successfully transform recognised components into HTML code. It was carried out using the coordinates obtained from the contour finding algorithms' output. Figure 5 shows the most recent output from a browser when the input picture is the first of the pictures in Figure 1.

ISSN: 2005-4262 IJGDC

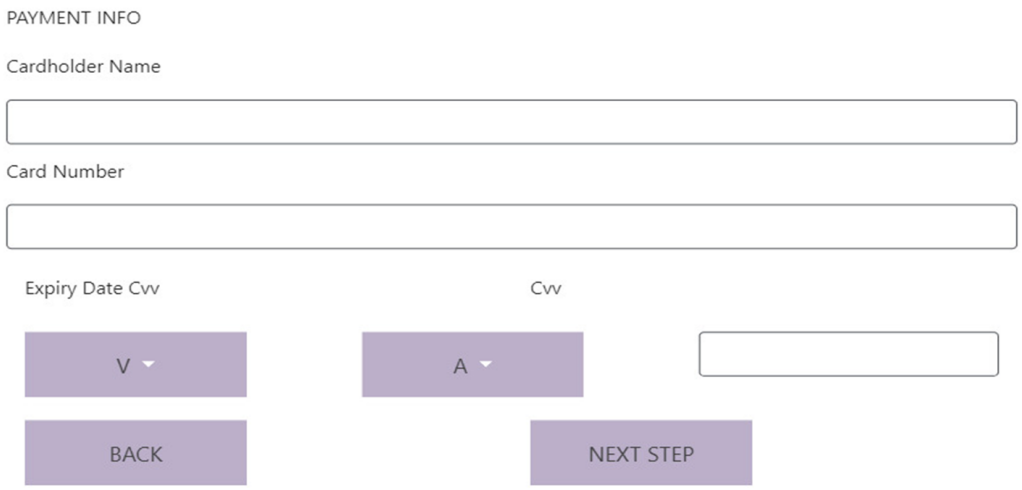

Figure 5. Shows a browser rendering of the HTML Builder algorithm's design code generated from the sketch input picture

First, we constructed the templates for a header and a footer, as indicated in the HTML builder algorithm in Fig. 6. Second, using component coordinates, we determined how many objects are on each of the rows. The component labels were then mapped to their template codes. The body part of the HTML code was successfully acquired at the end of this operation. Finally, the components of the header, body, and footer were integrated. As a result, the final HTML code was created. Figure 6 shows the method that was created for the purpose of completing HTML modification.

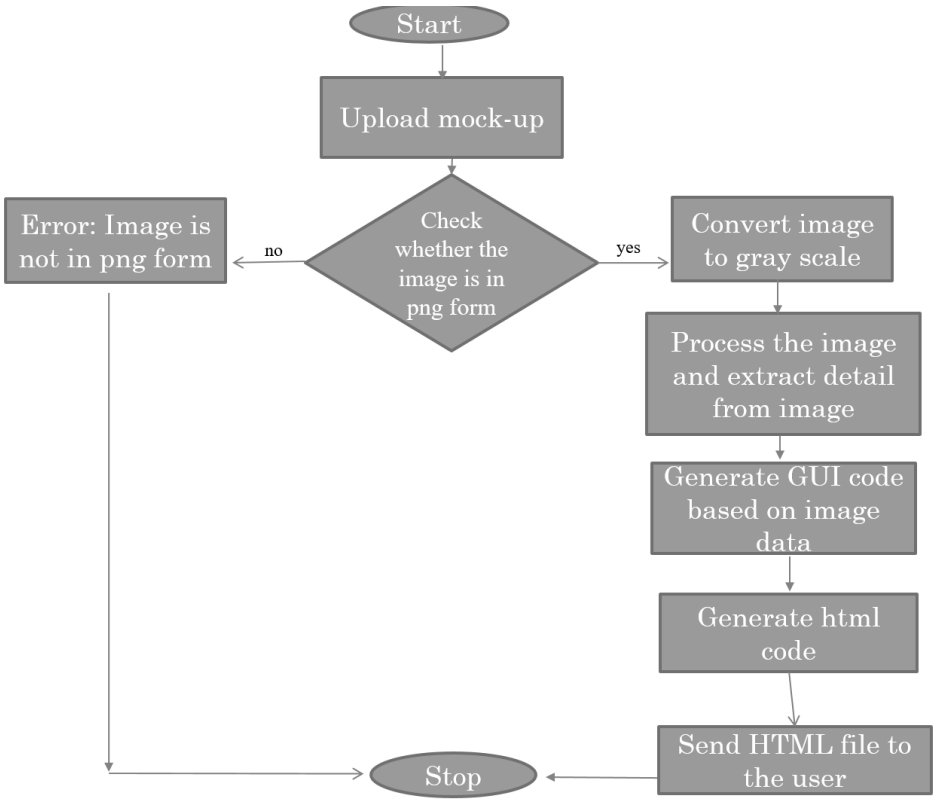

Figure 6. HTML Builder Algorithm

ISSN: 2005-4262 IJGDC

4. Conclusion

A critical aspect has been converting website mock-ups into mark-up code with less time and development expense. We built a technique in this study that takes web page mock-ups, processes them, and generates structured HTML code. A picture-based data collection was used, which included several prototypes of web page architectures. The CNN model is trained using this dataset. Although my work demonstrates the capabilities of such a framework to automate the process of executing GUIs, we have just begun to scratch the surface of what is possible. The model has few restrictions in general and is built on a similarly small dataset. The quality of the developed code might be significantly improved by developing a larger model based on a greater amount of data over a longer period of time.

References

[1] B. Aşıroğlu et al., "Automatic HTML Code Generation from Mock-Up Images Using Machine Learning Techniques," 2019 Scientific Meeting on Electrical-Electronics & Biomedical Engineering and Computer Science (EBBT), 2019, pp. 1-4, doi: 10.1109/EBBT.2019.8741736.

[2] K. Moran, C. Bernal-Cárdenas, M. Curcio, R. Bonett and D. Poshyvanyk, "Machine Learning-Based Prototyping of Graphical User Interfaces for Mobile Apps," in IEEE Transactions on Software Engineering, vol. 46, no. 2, pp. 196-221, 1 Feb. 2020, doi: 10.1109/TSE.2018.2844788.

[3] S. Natarajan and C. Csallner, “P2A: A Tool for Converting Pixels to Animated Mobile Application User Interfaces,” Proceedings of the 5th International Conference on Mobile Software Engineering and Systems -MOBILESoft ’18, pp. 224–235, 2018. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3197231.3197249

[4] T. Beltramelli, “pix2code: Generating code from a graphical user inter- face screenshot,” CoRR, vol.abs/1705.07962,2017. [Online]. Available: http://arxiv.org/abs/1705.07962

[5] K. P. Moran, C. Bernal-Cardenas, M. Curcio, R. Bonett, and D. Poshy- ´ vanyk, “Machine learning-based prototyping of graphical user interfaces for mobile apps,” IEEE Transactions on Software Engineering, pp. 1–1, 2018.

[6] S. P. Reiss, Y. Miao, and Q. Xin, “Seeking the user interface,” Automated Software Engineer- ing, vol. 25, no. 1, pp. 157–193, mar 2018. [Online].

ISSN: 2005-4262 IJGDC