International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 05 | May 2020 www.irjet.net p-ISSN: 2395-0072

Automatic HTML Code

Neenu James

Student, Dept. of Computer Applications, Christ Knowledge City College, Kerala, India

---***---

Abstract – *The first phase of designing a web site starts with creating mock-ups for individual web pages of the website. The mockups can be either by hand or using graphic design and specialized mock-up creation tools. In the next phase the mock up is then converted into structured HTML code by software engineers. Usually this process is repeated many more times until the desired template is created. In this paper, we have to generate the HTML code automatically from the mockups. This study aim is to automate the code generation process from hand-drawn mock-ups. These Hand drawn mock-ups are processed using computer techniques and deep learning techniques are used to implement the proposed system. This system achieves 96% method accuracy and 73% validation accuracy.*

Key Words: **HTML, Convolutional neural network, Deep learning, Object Recognition, Object Detection, Code Generation.**

1. INTRODUCTION

Due to the progress made in today's technology the importance of the Internet website is increased rapidly. No matter what our profession or business, a website can create goodwill among the customer and prospects. The user can easily access the website and speak their needs, and clear action step to take next.

A website is important, that can establish credibility of business. Nowadays websites are available for any stream of work i.e., in almost every field from social work to game training. Many websites is created for advertising purpose and product marketing. A website is mainly said as a collection web page that is connected and individually identified by a common domain name. A website must be always mobile compatible, accessible to all users, well planed information architecture, fast load time, effective navigation and good error handling. So, one of the most important parts of creating a website is to design the most interactive web pages. Web pages are the front-end of every website were the user interacts. So, it is important that the web pages should be simple and attractive so the user can easily interact.

However, developing a web page with those efficient features is a burdensome process. To develop a web page, graphic designers, software engineers, end-users and authorities and employed people in many areas required to work together. Usually the process of creating a web page begins with the mock up design by the graphic designers or mock up artists either on paper or a graphic editing software. And after creating, the software experts write the code for the web pages on these mock-ups. The resulting web pages may change based on the end-user feedback received. So the changing process may involve a lot of repetitive tasks. Like rewriting the code for the components in web page and page structure changing over time, make the process tedious. So in this case, we need a better solution for designing the web pages.

The best solution for this problem is to design a web page by generating automatic code. If web pages are created automatically then we can reduce programming time, process cost, and resource consumption. Also, the final website production can be done with shorter time period.

In this paper, we are designing an algorithm to generate an automatic HTML code for hand-drawn mock-ups. It is aimed to recognize the component in mock-up and will generate its HTML code automatically. To do this research a public dataset of hand-drawn images of websites obtained from Microsoft AI Lab's GitHub Page[1] is used to train and verify proposed scheme. Our model achieves about 96% method accuracy and 76% validation accuracy.

2. RELATED WORK

In [2] an algorithm REMAUI is used to find the components of the user interfaces a mobile application such as buttons, textboxes, and pictures and creates the code for them from the screenshots of an application window or conceptual drawings. In their study, which was the first regarding providing conversion to the code from the screen images or drawings for mobile platforms, computer vision, and optical character recognition methods are used. A prototype tool was implemented to evaluate REMAUI that generates the UI portion of Android applications. This tool is freely available in REMAUI. A big website exists between graphic artists' conceptual drawings and working UI code while developing the UI code of a mobile application. By re implementing the conceptual drawings in code, which is cumbersome and expensive the programmers manually bridge this gap. To bridge this gap, we introduced the first technique to automatically Reverse Engineer Mobile Application User Interfaces (REMAUI). On a given input bitmap REMAUI identifies UI elements via computer vision and OCR techniques. In our experiments on 488 screenshots of over 100 popular third-party applications, REMAUI-generated UIs were similar to the originals, both pixel-by-pixel and regarding their runtime UI hierarchies.

Although the REMAUI method works successfully, it does not support cross-page transition and animations within the page.

In [3] the authors developed the P2A algorithm to remedy the deficiencies of the REMAUI algorithm. The computer vision techniques for developing animated mobile applications are adopted using P2A tool. The average of 26seconds to infer in-screen animations is taken by P2A.

In[4] the graphical interface for a web page to structured code using deep learning with convolutional and recurrent neural networks was converted by pix2code algorithm. The Android, IOS and mobile platforms has been tested by these method and successful results have been obtained.

In [5], the algorithm ReDraw takes mock-ups of mobile application screens and creates a structured XML code for it. Computer vision techniques are used to detect individual GUI components in the first stage of their implementation. Classification of the detected components according to their function, e.g. toggle-button, text-area, etc are involved in second stage. Deep convolutional neural networks are also used in this stage. The XML code is generated by combining with the kNN algorithm according to web programming hierarchy is the last stage.

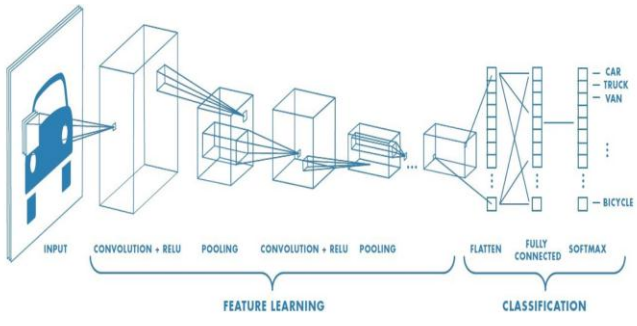


Fig -1: Typical Components of CNN Architecture

Nowadays open source code libraries such as Github [6] are used very frequently to share code and applications. It is a common practice to start or improving software projects to investigate this repository and reuse code. By using the shared codes in these libraries reduce the coding time by writing same code over and over by different people. In [7], users define a graphical interface with simple drawings and keywords by using a search program called SUISE. This interface is then searched in existing libraries to obtain similar interfaces. To select the most suitable interface the obtained interfaces are turned into operable codes and returned to the end user.

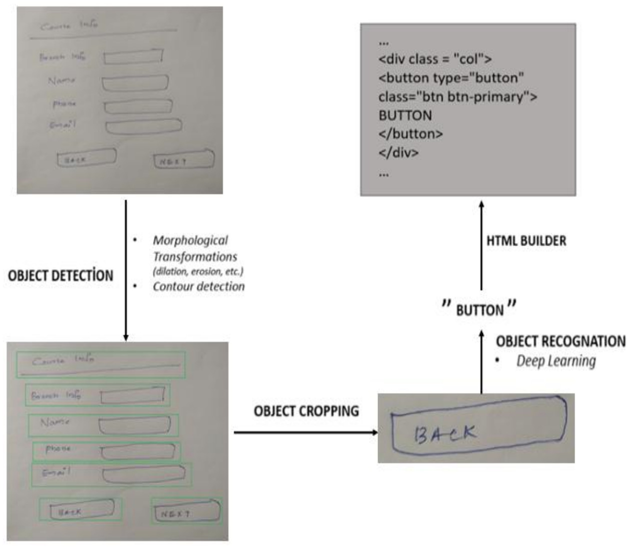
Microsoft has recently developed a system that takes hand-drawn mock-ups of simple web pages and creates the structure HTML code [1]. There are no literature that explains their work, however they have published their code and dataset online. In this project we use some of the images from this dataset.

3. DATASETS

In this study, we used some of the images provided from Microsoft AI Lab for their Sketch2Code application [1] in order to create our dataset. As we create it for the experiment, we selected the images that contains four type of components: textbox, dropdown, button, and checkbox. Then we cropped each component from these images and collected them to train our CNN model.

4. METHODS

This research was carried out in four basic steps. Object detection is the first step that was applied on the input image with image processing techniques such as erosion, dilation and contour detection. After object detection, the identified objects were cropped and then the components obtained were labeled with the trained CNN model which is the second step. The output of this model has been converted to HTML code through the HTML Builder script is the final step. The algorithm proposed is visualized in Fig. 2.



```
...
<div class = "col">
<button type="button"
class="btn btn-primary">
BUTTON
</button>
</div>
...
```

Fig -2: Proposed Algorithm.

4.1. Object Detection and Cropping

After getting the input file, it is converted to gray scale format. Then, 2 times Gaussian Blur was applied to them with 3x3 rectangle kernel. The components in the input image have been detected after the threshold process was carried out, rectangle were drawn by applying the contour detection algorithm to determine the objects by applying morphological transformations. After detecting components were cropped and transferred to the CNN model. The output of this stage is shown in Fig. 3 and Fig. 4.

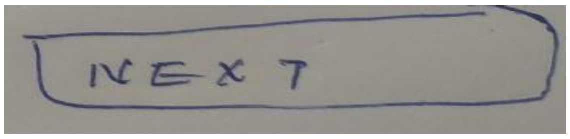


Fig -3: The output of cropping

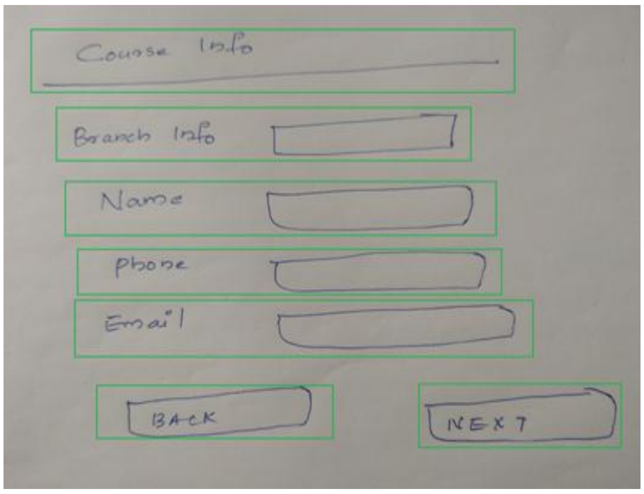


Fig-4: The output of the object detection process

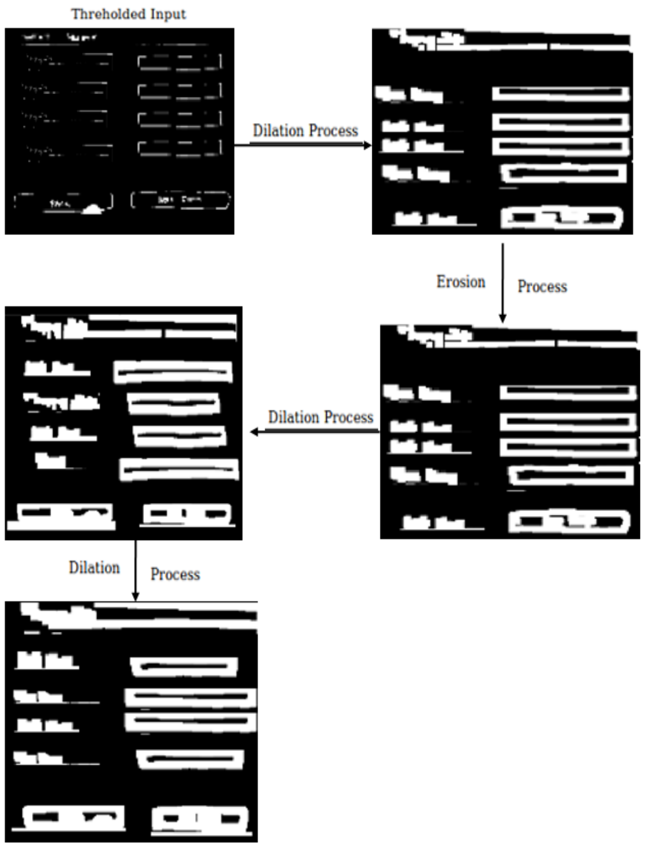


Fig-5: Morphological transformation processes

Usually, 8 iteration dilation was performed with 4x4 rectangle kernel in the stage of morphological transformations. And 3x3 ellipse kernel for 4 iterations was performed for erosion process and 1x10 rectangle shaped kernel for 2 iterations for dilation process. Finally, a 10x1 rectangle shaped kernel was used for dilation process. These operations are shown in Fig. 5

4.2. Object Recognition

The model shown in Fig. 6 was trained with the elements in our component dataset. The loss function was trained for 200 epochs using Binary Crossentropy and RMSProp algorithms by setting the batch size to 64 after the stage of training the model. Then, Component recognition process was carried out by giving the cropped components that came from the previous stage as input.

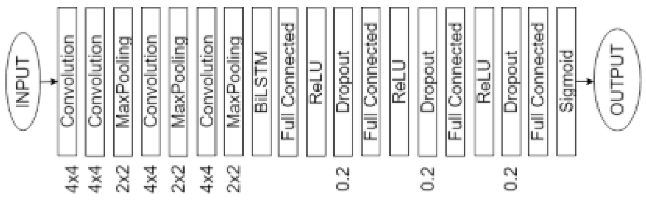


Fig-6: CNN Model

4.3 HTML Builder

After the second stage, recognized components were successfully translated into HTML code via the bootstrap framework. It was performed with the help of the coordinates from the result of the contour finding algorithms. The latest output that obtained from a browser when the input image is the first one of the images in Fig. 1 is demonstrated in Fig.7

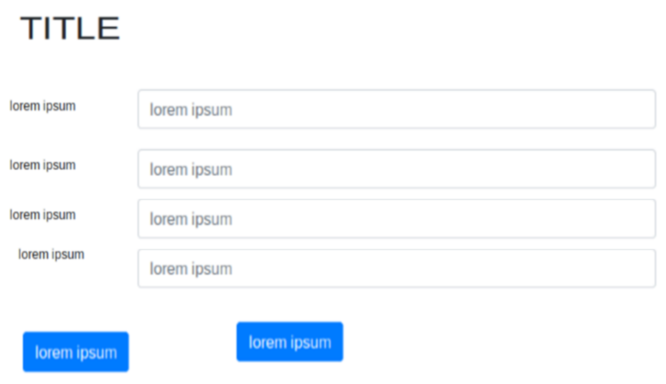


Fig-7: Browser render of the design code produced by HTML Builder algorithm from sketch input image

In HTML builder algorithm , first we created the templates for a header and a footer. Then, we detected how many items there are on each of the rows with coordinates of the components. And we mapped the labels of the components to their template codes. At the end of this process the body section of the HTML code was successfully obtained. Finally the header, body and footer sections were combined as well. So, the final HTML code was composed.

5. CONCLUSIONS

In creation of website the cost labor and minimum time can be reduced by converting web page mock ups to their markup code. Also, by using this automatic HTML code generator it is easy to make modification in web page creation with the least cost and minimum time. This can fasten the deployment process of website. The purpose of this study, is to create automatic HTML code from hand-drawn mock-ups was successful. For this study, we have designed four consecutive principles object reorganization, object cropping, object detection and HTML building. A CNN architecture was also used in object reorganization stage. As a result, after the training phase of 200 epoch, accuracy and validation accuracy were obtained as 96% and 73%, respectively.

ACKNOWLEDGEMENT

Interdependence is a higher value than independence. The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of people, who made it possible, whose constant guide and encouragement crowned our efforts of success. First and foremost, I thank almighty for all his blessings he showered on me. Also I like to express my gratitude to all Teachers and Staffs of Christ Knowledge City for their kind co-operation and help. I extend my thanks to all my friends for their moral support and encouragement. I would like to show my sincere gratitude to Ms. Amalumol S for her support.

Last but not least we thank our parents and benefactors who inspired me always to do our best.

REFERENCES

[1] Sketch2code. Microsoft AI Labs. [Online]. Available: https://github.com/

[2] T. A. Nguyen and C. Csallner, "Reverse Engineering Mobile Application User Interfaces with REMAUI (T)," in 2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE). IEEE, nov 2015, pp. 248–259. [Online]. Available: http: //ieeexplore.ieee.org/document/7372013/

[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 interface screenshot," CoRR, vol. abs/1705.07962, 2017. [Online]. Available: http://arxiv.org/abs/1705.07962

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

[6] [Online]. Available: https://github.com/

[7] S. P. Reiss, Y. Miao, and Q. Xin, "Seeking the user interface," Automated Software Engineering, vol. 25, no. 1, pp. 157–193, mar 2018. [Online]. Available: https://doi.org/10.1007/s10515-017-0216-3