PROTOCODE

R L Rahul Krishnan
Department of Computer Science And Engineering
SRM Institute of Science and Technology
Chennai,
India

Surya Bharathi
Department of Computer Science And Engineering
SRM Institute of Science and Technology
Chennai,
India

Shashank Chakrawarty
Department of Computer Science And Engineering
SRM Institute of Science and Technology
Chennai,
India

M Muthurasu
Department of Computer Science And Engineering
SRM Institute of Science and Technology
Chennai,
India

***Abstract* -The cycle for designing a web application starts with creating mock-ups for individual web applications either by using graphic design or hand and specialized mock-up creation tools. The mock-up is then converted into structured HTML code or similar markup code by web designers. This process is usually repeated many more times until the desired template is created. In this study, we aim to automate the code generation process from hand-drawn mock-ups. Also, separate teams are required to generate mock-up and to develop HTML code. By using protocode the numbers of teams involved can be minimised and the 6 by decreasing the phases involved in creating the UI**.

Index terms: Deep Learning, Convolutional Neural Networks, Gated Recursive Unit, ResNet-152, HTML

I. INTRODUCTION

The significance of Internet web applications has increased considerably due to the progress made in today's technology. Nowadays web applications reflect the confrontations of the states, institutions, communities, individuals, etc. Web applications are accessible in almost every field, from information to social work, from diversions to training and numerous more. Web locales made by companies come to the fore for budgetary reasons for item marketing or promoting purposes. On the other hand, official institutions aim to supply more proficient administrations.

At the front-end of each web, the location could be a "web page" which is the portion of the internet location that interacts with the client. It is very important to serve a page that draws in the consideration of the end-user, is simple to utilize and has enough functional features. However, creating web pages that react proficiently to these needs includes a burdensome handle. Within the [1] preparation of web pages, graphic designers, program specialists, end-users, corporate specialists and individuals utilized in many different ranges are required to work together. More often than not, the process begins with the mock-up plan of the client interface by the realistic originators or mock-up craftsmen, either on paper or a realistic altering computer program, in line with desires of the institution. Software experts compose code for the net pages based on these drafts. The coming about web pages may alter based on input gotten by the conclusion clients. Their handle involves a part of dreary errands. Revamping the code for components with comparative capacities and page structures changing over time makes the method repetitive. These uncovers we have to explore more productive arrangements in web page plans.

The thought of planning a web page by producing automatic code is picking up significance as a research subject. [5] Automatic production of web pages diminishes programming time, process cost and asset utilization. Much obliged to the speedier progressive plan stages, the ultimate web-site is delivered in a shorter time.

In this study, an algorithm has been created to automatically produce the HTML code for hand-drawn mock-up of a web page [1]. It is pointed to recognize the components created in the mock-up and to encode them agreeing to the web page chain of command [2]. An open dataset of hand-drawn pictures of web destinations gotten from Microsoft AI Labs' GitHub page is used to prepare and confirm the proposed plot. The pictures on the dataset are prepared to utilize computer vision strategies and a profound neural arrange show including convolutional neural networks are utilized to prepare the information. A short time later a structured HTML code is obtained.

Our model accomplishes 97% accuracy using blue score.

II. RELATED WORK

In an algorithm named as REMAUI finds the components of the client interface a portable application such as buttons, textboxes, pictures and makes the code for them from the screenshots of an application window or conceptual drawings. In their consideration, which was the primary in terms of providing conversion to the code from the screen pictures or drawings for versatile stages, computer vision and optical character recognition strategies are utilized. In spite of the fact that the REMAUI [2] method works effectively, it does not bolster cross-page transition and activity inside the page. The creators developed the P2A calculation to cure the lacks of the REMAUI [2] algorithm. The pix2code [1] calculation which aims to change over the graphical interface for a web page to structured code utilizing profound learning with convolutional and recurrent neural systems. The strategy has been tried on Android, IOS and portable stages and effective comes about have been gotten.

[3] The algorithm named ReDraw takes mock-ups of mobile application screens and makes an organized Bootstrap and CSS code for it. Within the to begin with organizing their usage computer vision techniques are utilized to identify GUI components. The second stage includes classification of the identified components according to their work, e.g. toggle-button, text-area, etc. In this organization profound convolutional neural networks are utilized. In the final stage, the CSS and HTML code is produced by combining with the RNN calculation concurring to a web programming chain of command.

Pix2code [4] has as of late created a framework that takes hand- drawn mock-ups of basic web pages and makes the structure HTML code. There is no writing that clarifies their work, be that as it may they have distributed their code and dataset online. To extend this we utilize a few of the pictures from this dataset.

In [6], the authors use a search program called SUISE in which the clients characterize a graphical interface with basic drawings and keywords. This interface is then searched in existing libraries and obtains similar interfaces. These interfaces are turned into operable codes and returned to the end user to choose the foremost appropriate interface

III. DATASET

Our perfect training dataset would have been thousands of sets of hand-drawn wireframe portraits and their HTML code equivalents.

Obviously, we couldn’t discover that correct dataset, and we had to form our own dataset for the task.

IV. METHODS

Firstly, the training images are converted into matrices which are distributed into 0-255 characters and divided into three dimensions for red, green and blue. In spite of the fact that much of program synthesis deals with code produced from natural language specification or execution traces, in our case we may use the reality that we had a source picture (the hand-drawn wireframe) to begin with. There’s a well-studied space in machine learning called "image captioning" that looks for to memorize models that tie together images and text, particularly to produce depictions of the substance of a source image. Given the image captioning approach, our perfect training dataset would have been thousands of sets of hand-drawn wireframe portrays and their HTML code equivalents. Obviously, we couldn’t discover that correct dataset, and we had to form our own dataset for the task. Each created site within the dataset comprises combinations of a number of straightforward Bootstrap components such as buttons, text boxes, and div. In spite of the fact that this implies our model would be constrained to these few components as its ‘vocabulary’. The approach should easily generalize to a larger vocabulary of elements. The source code for each test comprises tokens from a domain-specific-language that we created for their task. Each token corresponds to a snippet of HTML and CSS, and a compiler is utilized to translate from the DSL to working HTML code. We utilized a model design used in image captioning that comprises three major parts: A computer vision model that employs a “Convolutional Neural Network (CNN)” to extricate picture highlights from the source images. A dialect show comprising a “Gated Recursive Unit (GRU)” that encodes groupings of source code tokens. A decoder show (moreover a GRU), which takes within the output from the past two steps as its input, and predicts the following token within the arrangement. The CNN is encoded by Loading the pretrained ResNet-152 and replacing the top fc layer also weights are set followed by image feature vectors.

To train the model, we split the source code into sequences of tokens. A single input for the model is one of these sequences along with its source image, and its label is the next token in the document. The loss-function utilized in the model is cross-entropy cost which helped us in comparing the model’s next token prediction with the actual next token. The CNN is encoded by Loading the pretrained ResNet-152 and replacing the top fc layer also weights are set followed by image feature vectors. At inference time when the model is tasked with generating code from scratch, the process is slightly different. The image is still processed through the CNN network, but the text process is seeded with just a starting sequence. At each step, the model's prediction for the next token in the sequence is appended to the current input sequence, and fed into the model as a new input sequence. This is repeated until the model predicts an <END> token or the process reaches a predefined limit to the number of tokens per document. The compiler converts the Domain specific language tokens into HTML code, which can be opened in any browser, when the set of predicted tokens gets generated from the model.

We decided to use the BLEU score to evaluate the model. This is a common metric utilized in machine translation errands, which looks for to degree how closely a machine-generated content takes after what a human would have generated, given the same input. Basically, the BLEU compares n-gram arrangements of both the generated content and reference content to form an altered shape of accuracy. It's exceptionally appropriate for this project since it factors within the real components within the generated HTML, as well as where they are in connection to each other. The below graph indicates how BLEU score improved with more datasets.

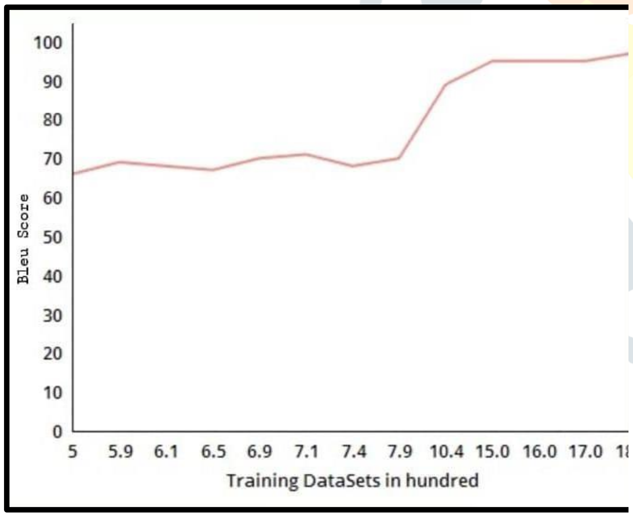

Fig. 1. Evaluation Graph

V. RESULT

In this consideration, the reason for producing programmed HTML code from mock-up pictures was effectively reached. For this purpose, four continuous vital stages were carried out such as question discovery, protest editing, question recognition and HTML building. These progressive steps are the mile- stones of our proposed calculation. CNN algorithm is utilized to train the model with better loss function and resnet better accuracy is made. The generated html code is also feasible to be used in any devices as it involves usage of bootstrap. And we obtained a score of 0.97 accuracy. In the end as a result we get the web page generated from the hand drawn mockup image which any browser can display.

VI. CONCLUSION

The HTML code which was generated has the high scope of agility, hence protocode is not restricted only to modelling and POC but also can be used in the production environment. The model can be more improved by increasing epoch, more training data set and intuiting more components. The code generation model can be made into a docker and deployed anywhere hence giving huge contribution in cost, time, resource saving by the organization.

VII - REFERENCES

[1] Tony Beltramelli. "pix2code: Generating Code from a Graphical User Interface Screenshot". In: CoRR abs/1705.07962 (2017). arXiv: 1705.07962.url:://arxiv.org/abs/1705. 07962.

[2] T. A. Nguyen and C. Csallner, "Reverse Engineering MobileApplication User Interfaces with REMAUI (T)," in 2015 30thIEEE/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 Pixelsto Animated Mobile Application User Interfaces," Proceedings of the5th 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-Cardenas, M. Curcio, R. Bonett, and D. Poshy- ´vanyk, "Machine learning-based prototyping of graphical user interfaces for mobile apps," IEEE Transactions on SoftwareEngineering, pp. 1–1,2018.

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