Doodle2Code: Generating Code from a Hand Drawn Sketch

Ziyuan Zhu
zyzhu@mit.edu

Ous Abou Ras
ous@mit.edu

Jaeyoon Song
jaeyoons@mit.edu

1 Introduction

There has been a growing interest in automating the conversion of a graphical user interface screenshot created by a designer into code (12; 9; 10; 18; 20). It is typical for designers and people who want to make a website to find some reference website designs to define the style of the website, draw a simple sketch, and mock up it in low and high resolution. However, due to the time and skill limitation, it is hard for those who design and mock up to do the developing work at the same time. Thus, this paper aims to take the existing literature to the next step by exploring transforming a sketch and style reference to the HTML code. A sketch is restricted to a portrait image of a black-and-white, rough drawing without a lot of details. An existing website screenshot or picture works as supplementary material for stylizing the website with CSS code.

In this milestone presentation, our team will mainly focus on the first part — transfer hand drawing sketch to HTML code. The related work is as follows:

Pix2Code. Many research projects focus on transforming graphical user interface screenshots created by a designer into code. Pix2Code uses deep learning methods to generate code from a single input image and obtain over 77% accuracy for iOS, Andriod, and web-based platforms. Inspired by this project and doing initial research with the project, our work uses the pix2code dataset as the basis and generates new data as a supplement. To see more details about pix2code.(`https://github.com/tonybeltramelli/pix2code`)

Sketch-Code. Sketch-Code is a deep learning model which takes hand-drawn web mockups. Based on the synthetically generated dataset and model architecture from pix2code, it uses the drawn framework to retrain the model. However, due to the limited stylish elements in bootstrap, the project's results are in mono-tune. In our projects, we refer to the methods of creating the sketch-HTML dataset and improve the sketch-code idea by improving website style with CSS code. More details about Sketch-Code.(`https://github.com/ashnkumar/sketch-code`)

2 Method

2.1 Data Generation

An ideal dataset would consist of a set of a hand-drawn wireframe sketch and the corresponding HTML/CSS code. To obtain this dataset, we referred to the pix2code dataset that contains 1,743 sets of screenshot, Domain Specific Language (DSL) code, and HTML code (12) (See Appendix A).

DSL generation. Our dataset has three improvements compared to the pix2code dataset. First of all, we defined more DSL tokens that correspond to a Bootstrap element, in comparison to existing approaches (17; 16; 21; 12). In particular, we added tokens for images, text with different sizes, different types of headers, and empty space (See Appendix A). Second, we updated the version of Bootstrap library to the most recent one (24). Lastly, we increased the volatility of the position of each element in our dataset in order to improve the performance on a wider variety of sketches (See Figure 1).

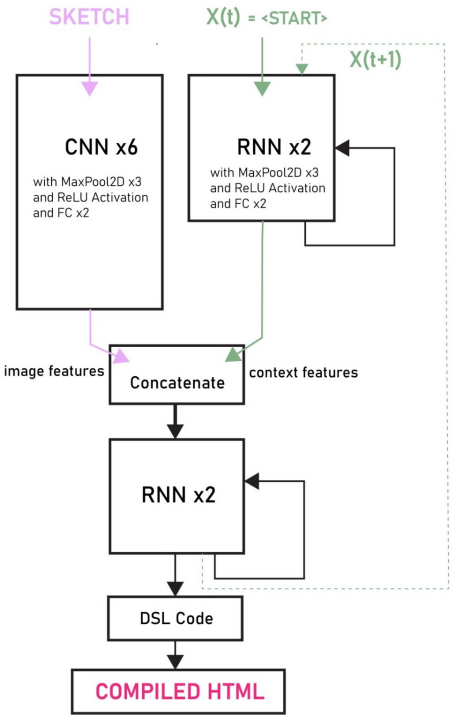
*The three authors contributed equally to the paper

Figure 1: The comparison of the existing dataset (A) and our new dataset (B). The sketches from our dataset has higher variability.

We generated DSL codes via three steps. First, we defined the tokens and mapped them to the corresponding HTML. We also modified the previous DSL tokens written by pix2code in order to fit the recent version of Bootstrap. Second, we defined six types of grid using single, double, and quadruple blocks. Finally, we loop through each case of grid types and randomly generate contents to be placed inside each block. The codes used for the process above can be found at the following link: `https://github.com/sketch2code-mit/data-generation`.

Sketch synthesis. Then we compiled the generated DSL codes into HTML, styled the compiled HTML codes so that they look more like sketches, and then took screenshots of the HTML pages using a python library (`https://github.com/SeleniumHQ`). Alternatively, we tried pre-processing the screenshots with OpenCV and PIL library to detect the contour and acquire synthetic sketches (16; 21). The codes used for taking screenshots are at `https://github.com/sketch2code-mit/html-to-image`. As a result, we gained 2,160 pairs of sketches and the corresponding DSL codes in addition to the existing 1,743 pairs in the pix2code dataset.

Figure 2: The Network Architecture



2.2 Network Architecture & Training

Our method uses a Pytorch version of Tony Beltramelli's Pix2Code network (12), implemented by Vaibhav Yadav (`https://github.com/VaibhavYadav/pytorch_pix2code`). We changed the code to train it on a dataset (generated by Vincent Kieuvongngam (22)) of screenshots of HTMLs that were adjusted with CSS to look like sketches. We also train the model on our own dataset of automatically generated HTML files where we increase the number of tokens.

The network is a Recurrent Neural Network that reads an image using Convolutional Neural Networks and outputs a DSL file (which is later compiled into HTML) divided into two parts, the Encoder and the Decoder:

(1) Encoder: The encoder has two parts, the Image Encoder, and the Context Encoder

Image Encoder: A CNN Network that reads the screenshot/sketch and outputs a flattened vector of features.

Context Encoder: An RNN Network that inputs context of the last bit of generated code and also takes in as input the starting context " <START>". The combination of both of these Encoders help define the order for the Decoder to know what to generate given the image and last generated context.

(2) Decoder: An RNN network that takes in the concatenates the output of both encoders and generates the next DSL Token in the code.

The model is trained for just 10 epochs, using the Adam Optimizer and a learning rate of 0.0001. The loss fn used is the Cross Entropy Loss Fn between expected output of tokens and actual output of these tokens.

Among the total 2,160 pairs of sketches and DSL codes, we used 1,800 for the training process. The code used for the training process is at the following url: `https://github.com/sketch2code-mit/doodle2code`.

3 Results

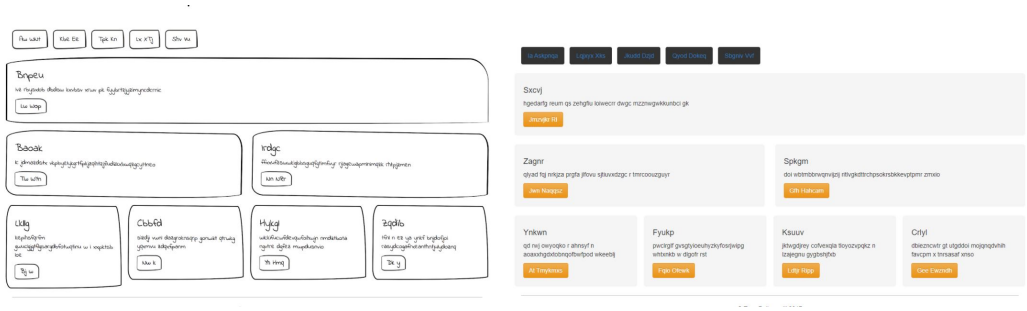


Figure 3: The input sketch (left) and the output web page (right) for pix2code Dataset.

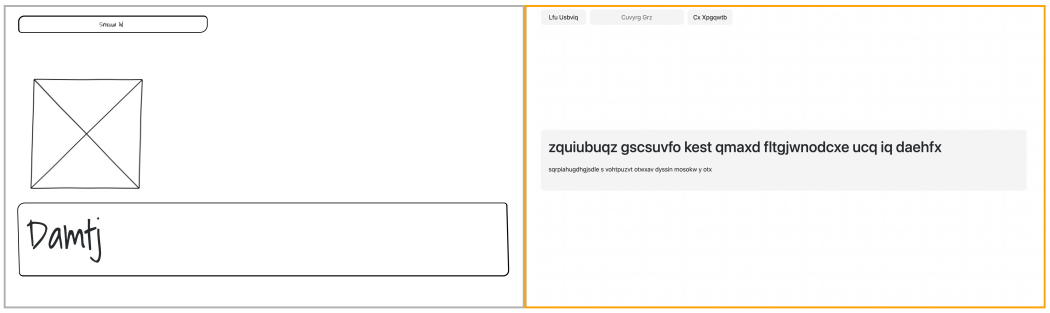


Figure 4: The input sketch (left) and the output web page (right) for our Dataset.

The intermediate results are not as good as we expected. The outputs for various sketches are all similar to each other, even if the input sketch is very different. We believe there are two causes for this issue. First of all, it might be because we did not randomize the order of training sets, while the order has a certain meaning: the closer the order is, the more similar the structures of the sketches are. We plan to shuffle the dataset in the future training. Second, the same model worked great on the pix2code dataset that was adjusted to look like a sketch using a similar method as ours. It had a loss of 0.02. The loss on our generated dataset was 0.2. Note that the pix2code dataset is more structural and has less tokens (19 compared to 28 for ours) which could be one of the reason our model found a harder time getting a better result.

There are three directions for future improvements of out existing model. First, we could generate more data. Particularly, our current dataset almost always contains one image in the first row and boxes in the second row. We can remove this pattern from the dataset. Second, generating stylesheet in a similar way might help complete the web page generation process. Lastly, we can improve the network itself by using Long short-term memory (LSTM) or better recurrent neural network (RNN)

models, and explore such architectures in more depth. Another possibility is to add dropout and mini-batches.

References

[1] Levenshtein, Vladimir I(1966): *Binary codes capable of correcting deletions, insertions, and reversals*, 8: 707–710.

[2] Chen, Tao / Cheng, Ming Ming / Tan, Ping / Shamir, Ariel / Hu, Shi Min(2009): *Sketch2photo: Internet image montage*, 5: 1–10.

[3] Hu, Rui / Collomosse, John(2013): *A performance evaluation of gradient field hog descriptor for sketch based image retrieval*, 7: 790–806.

[4] Wang, Lijun / Ouyang, Wanli / Wang, Xiaogang / Lu, Huchuan(2015): *Visual Tracking with Fully Convolutional Networks*In: 2015 IEEE International Conference on Computer Vision (ICCV)3119–3127.

[5] Wigington, Curtis / Stewart, Seth / Davis, Brian / Barrett, Bill / Price, Brian / Cohen, Scott(2017): *Data augmentation for recognition of handwritten words and lines using a CNN-LSTM network*639–645.

[6] Ellis, Kevin / Ritchie, Daniel / Solar Lezama, Armando / Tenenbaum, Joshua B(2017): *Learning to infer graphics programs from hand-drawn images*.

[7] Deka, Biplab / Huang, Zifeng / Franzen, Chad / Hibschman, Joshua / Afergan, Daniel / Li, Yang / Nichols, Jeffrey / Kumar, Ranjitha(2017): *Rico: A Mobile App Dataset for Building Data-Driven Design Applications*In: Proceedings of the 30th Annual Symposium on User Interface Software and Technology.

[8] Yun, Young Sun / Jung, Jinman / Eun, Seongbae / So, Sun Sup / Heo, Junyoung(2018): *Detection of gui elements on sketch images using object detector based on deep neural networks*In: International Conference on Green and Human Information Technology86–90.

[9] Chen, Chunyang / Su, Ting / Meng, Guozhu / Xing, Zhenchang / Liu, Yang(2018): *From UI design image to GUI skeleton: a neural machine translator to bootstrap mobile GUI implementation*In: Proceedings of the 40th International Conference on Software Engineering665–676.

[10] Moran, Kevin / Bernal Cárdenas, Carlos / Curcio, Michael / Bonett, Richard / Poshyvanyk, Denys(2018): *Machine learning-based prototyping of graphical user interfaces for mobile apps*, 2: 196–221.

[11] Baltrušaitis, Tadas / Ahuja, Chaitanya / Morency, Louis Philippe(2018): *Multimodal machine learning: A survey and taxonomy*, 2: 423–443.

[12] Beltramelli, Tony(2018): *pix2code: Generating code from a graphical user interface screenshot*In: Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems1–6.

[13] Aşıroğlu, Batuhan / Mete, Büşta Rümeysa / Yıldız, Eyyüp / Nalçakan, Yağız / Sezen, Alper / Dağtekin, Mustafa / Ensari, Tolga(2019): *Automatic HTML code generation from mock-up images using machine learning techniques*In: 2019 Scientific Meeting on Electrical-Electronics & Biomedical Engineering and Computer Science (EBBT)1–4.

[14] Audebert, Nicolas / Herold, Catherine / Slimani, Kuider / Vidal, Cédric(2019): *Multimodal deep networks for text and image-based document classification*427–443.

[15] Li, Mengtian / Lin, Zhe / Mech, Radomir / Yumer, Ersin / Ramanan, Deva(2019): *Photosketching: Inferring contour drawings from images*In: 2019 IEEE Winter Conference on Applications of Computer Vision (WACV)1403–1412.

[16] Robinson, Alex(2019): *Sketch2code: Generating a website from a paper mockup*.

[17] Jain, Vanita / Agrawal, Piyush / Banga, Subham / Kapoor, Rishabh / Gulyani, Shashwat(2019): *Sketch2Code: Transformation of Sketches to UI in Real-time Using Deep Neural Network*.

[18] Chen, Sen / Fan, Lingling / Chen, Chunyang / Su, Ting / Li, Wenhe / Liu, Yang / Xu, Lihua(2019): *Storydroid: Automated generation of storyboard for Android apps*In: 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)596–607.

[19] Brown, Tom B u.a.(2020): *Language models are few-shot learners*.

[20] Chen, Jieshan / Xie, Mulong / Xing, Zhenchang / Chen, Chunyang / Xu, Xiwei / Zhu, Liming / Li, Guoqiang(2020): *Object detection for graphical user interface: old fashioned or deep learning or a combination?*In: Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering1202–1214.

[21] Zita, Aleš / Picek, Lukáš / Říha, Antonín(2020): *Sketch2Code: Automatic hand-drawn UI elements detection with Faster R-CNN*.

[22] Kieuvongngam, Vincent(2020): *Sketch2code using Visual Attention LSTM Decoder*.

[23] Ferreira, J / Restivo, André / Ferreira, Hugo Sereno(2021): *Automatically Generating Websites from Hand-drawn Mockups*.

[24] Twitter(2021): *Bootstrap · The most popular HTML, CSS, and JS library in the world.*

[25] Themesberg(2021): *gpt-3-tailwindcss*.

Appendix

A. Domain Specific Language (DSL) designed by Pix2code

Pix2code(12) designed a lightweight domain specific language (DSL) to describe HTML components. The simplicity of DSL reduces not only the size of the search space, but also the size of the vocabulary (i.e. the total number of tokens supported by the DSL). The following are the list of components defined in the DSL of the pix2code project.

- Header
- Row
- Single
- Double
- Quadruple
- Buttons
- Text
- Big title
- Small title

In addition to using these components, we modified the pix2code DSL to define more variety of components to improve the performance as follows.

- Header that is right-aligned
- Container
- Image
- Big text
- Small text
- Empty space