EasyChair Preprint

№ 9410

Coding Sketch: Turn Your Sketch Designs into Code

Ayush Gour, Sankalp Sudarsan Rajguru, V Shivanshu Yadav and Shilpa Sharma

December 3, 2022

Coding Sketch: Turn Your Sketch Designs into Code

Ayush Gour, Sankalp Sudarsan Rajguru, V Shivanshu Yadav, Shilpa Sharma

Lovely Professional University
E-mail:ayushgour232@gmail.com, sankalprajgurur1221@gmail.com,
shivanshu.yadav04@gmail.com, shilpa.sharma@lpu.co.in

Abstract

It is an online application that essentially allows you to convert a hand-drawn drawing into HTML code. Any hand-drawn plan may be instantly transformed into HTML code using artificial intelligence. This strategy is unusual and unique. This study shows how several machine-learning approaches may be crucial in building a model from scratch that can generate code from a user-supplied picture. We demonstrate two ways to automate this process: conventional computer vision approaches and state-of-the-art deep semantic segmentation networks. Finally, we release a dataset for systems analysis and training.

Keywords: HTML, Deep Learning, Sketch-to-Code, Artificial Intelligence

Introduction

Web development is a multi-phase process. This also includes wireframe-based website design. The front and back codes come next. Their demands are met by the developer's completely working and enjoyable website. Every website begins with a fundamental concept and basic structure that may be outlined in a wireframe. It supplies the necessary components and gives developers suggestions for building a site structure. The responsibility of the developers is to create the boilerplate code that sets everything up correctly. Like turning wireframes into HTML code, it takes a lot of time and effort. Most boilerplate codes are now created by hand. Users now must write HTML code to organize items on a web page. This forces users to spend precious resources and time on redundant tasks. When the structure of web pages is the same, users are more likely to save a copy of the well-before source code for reuse. Even though the code for the Html elements is the same, the boilerplate script for web pages frequently differs. Such circumstances render the task unnecessary. However, you may automate this procedure to help the world's web developers.

A machine learning model is just what we recommend using to make this process simpler. As with components that identify text from the wireframe, it will be trained to recognize certain symbols and shapes. The model is an outline picture—a web application's input. Processing the information is the goal. Utilize the open-source and free Computer Vision Library to find every piece in a wireframe (Open CV). The backend will have the relevant code for each recognized element. Once the items on the sketch have been identified, the appropriate code is subsequently added to an HTML file. An HTML file serves as the user's product.

Coding Sketch uses machine learning and artificial intelligence to convert a scribbled interface design from an image to a valid HTML layout code. Learn more about how Coding Sketch converts a handwritten graphic to HTML. Much imagination goes into the user interface design process, which begins on a whiteboard where designers discuss ideas. After one design is created, it is often photographed and manually converted into a functional HTML wireframe that can be seen in a web browser. This requires work and slows down the design process. Instead, it uses artificial intelligence, a web-based program that converts a handwritten user interface sketch from an image to a valid HTML markup code.

Making a wireframe on paper to outline the interface's structure is a preliminary stage in the development of an application (Pedro Campos and Nuno Nunes, 2007) and (James A. Landay, 1995). The problem for designers is turning their wireframe into code, which frequently entails handing the design off to a developer who will then create the boilerplate graphical user interface (GUI) script. This effort takes the developer a long time and is consequently expensive (T. Silva da Silva et al, 2011). Prior research has been done on the following issues related to translating designs into code: Digital drawings are transformed into application code using movements by SILK (J. A. Landay, 2001). Many of these applications do detection and classification using traditional computer vision algorithms. However, we have found a gap in the body of knowledge that addresses the issue of excessive archiving. A program that converts wireframe designs into code. This program has several advantages: Faster iteration: A wireframe may be transformed into a website prototype with the help of simply the designer; accessibility: non-developers can construct apps. Allows developers to concentrate on the application code rather than boilerplate GUI programming by removing the demand for prototypes from the development process. We believe a unique deep-learning approach to this problem might improve performance compared to standard computer vision approaches.

Literature review

To convert the wireframe setup into code, intelligent systems and the suggested model go through the process of picture analysis and pattern recognition built on an ML model (Pedro Campos and Nuno Nunes, 2007), (James A. Landay, 1995) and (T. Silva da Silva et al, 2011). It depicts the tedious yet time-consuming job a UI designer performs while turning a Graphical User Interface (UI) design into a programmed UI application. This process will be significantly sped up by an automated system that can substitute human efforts for the simple implementation of UI ideas. The publications that advocate for such a system emphasize using UI wireframes (Pedro Campos and Nuno Nunes, 2007) rather than hand-drawn drawings as input. A platform-independent UI representation object is the network's output (T. Silva da Silva et al, 2011). A dictionary of key-value pairs is used to represent user interface components and their associated attributes. Our UI parser uses this as input and generates code for many platforms. Because of its inherent platform neutrality, the model can train once and provide UI prototypes for several platforms (J. A. Landay, 2001) and (Pedro Campos and Nuno Nunes, 2007). The design phase of a website takes much time, but the systems do not always function as planned. For this reason, only a few libraries employ specific libraries like OpenCV (Andrej Karpathy, 2014), which analyze images and other contours to decrease noise and provide a foundation for precise picture analysis. Making mockups of individual web pages, either by hand or with graphic design and specialist mockup production tools, is the first step in designing and developing a website. Software programmers then transform the mockup into structured HTML or another type of markup code. (James A. Landay, 1995) and (Pedro Campos and Nuno Nunes, 2007). An industry partner undertook a user-centered concept creation process for a Machine Learning (ML) based design tool. The final proposal uses ML to build consistent wireframes by matching graphical user interface elements in paper sketches to their digital equivalents (J. A. Landay, 2001) and (T. Silva da Silva et al, 2011). As soon as the photos and patterns are identified, we can use the text detection method and built-in ML Model library (James A. Landay, 1995) and (Andrej Karpathy, 2014) to separate the pictures from the text and then develop the conversion model to provide the output (J. A. Landay, 2001). It might be difficult to extract text from intricate photos or have more colour. Textual information found in photographs may be used to structure, index, and consistently explain images. The text in each image is extracted via detecting, localizing, tracking, removing, improving, and recognizing it (Oriol Vinyals et al, 2014). This document presents a system and technique for dynamically generating source code for a software application from a collection of wireframe pictures. A series of wireframe pictures are transmitted from an end user's device via a network to a wireframe recognition and analysis engine (James A. Landay, 1995). The attributes that make up each wireframe in the collection of wireframe

pictures are determined by comparison with a model library and then recorded to a data storage. The data store's contents are processed by an inference engine, which is guided by a base of knowledge of wireframe design principles to create a set of wireframe components. A template engine dynamically generates source code for the software program using a collection of graphical elements and a bunch of Source templates. The whole output of source code is reduced into a single archive folder and made available for download to the end user's device (James Lin et al, 2000). Once the product has been formed, the HTML code must be converted. This must be done by creating the HTML code from scratch using sophisticated approaches, which calls for a skilled developer. To generate a personalized user interface, the program was constructed to transform user-generated images into HTML code (James Lin et al, 2000). The ability to train pre-defined models using machine learning is virtually shown in this notion. The user-provided data is reverse-engineered using these models. A code that is generated using this idea will be more accurate. Additional platforms can be added to the compatibility. This analysis shows that the deep learning technique outperforms our traditional computer vision approach and concludes that deep learning is the most effective strategy for future study (James Lin et al, 2000) and (T. A. Nguyen, 2015). The present scenario addresses this reality and provides information on the automatic code-generation approaches for using various inputs to generate code in different programming languages (Chao Dong et al, 2015). These define the breadth of the available technology.

COMPUTER VISION AND TECHNIQUES

Image processing or technique accepts input from web cameras or real-world photos. Due to changes in current across the camera sensor, these pictures frequently have Gaussian noise. Edge detection (S. Singh and B. Singh, 2015), which we use for element detection, can perform poorly on noisy images. Therefore, it is crucial to lessen this noise. Although denoising auto-encoders, a kind of deep learning approach, are particularly good at removing noise, they are slower than kernel filtering methods.

1. **Colour Detection:** in our technique, element detection is aided by colour detection. This section explains many methods for identifying colours in visual media. However, we concentrate on threshold-based detection since it is a valuable method for handling big, homogeneous colour blobs. Red, Green, and Blue (RGB), often known as a colour space, are the three colour channels most frequently used to describe digital pictures.

2. **Edge Detection:** we are interested in identifying components in wireframe drawings. The icons for the wireframe elements mostly have straight edges. Therefore, we employ edge detection as a crucial method for element discovery.

3. **Segmentation:** wireframe elements must first be discovered before they can be categorized. Since a wireframe sketch will likely include several components, a technique for identifying element boundaries is necessary. Numerous possible segmentation algorithms are available. Since our first approach uses traditional computer vision methods, trainable segmentation will not be considered a segmentation object for this approach.

4. **Text Detection:** we employ the stroke width transform in our algorithm to identify text from drawings. It should be noted that SWT is not a text recognition program; instead, it is a quick, lightweight, and language-independent scene text detector. SWT's quickness and linguistic neutrality make it extremely useful for our technique.

Machine Learning Techniques

1. **Deep Learning:** this area of machine learning, also known as "deep learning", uses deep neural networks with several hidden layers. Deep understanding has demonstrated phenomenal performance in several sectors, frequently exceeding conventional approaches [7, 9]. Since this is essentially a visual problem and deep learning has established itself as the gold standard for superior performance in many vision problems.

2. **TensorFlow:** a free software library called TensorFlow exists. TensorFlow was initially created by engineers and researchers working on the Brain Team of Google within Google's Machine Intelligence research organization to conduct deep learning and machine learning research. Still, the system is versatile enough to be used in a variety of other domains as well.

3. **Keras:** an open-source, Python-based high-level neural network framework called Keras is powerful enough to operate with TensorFlow. It is designed to be user-friendly, expandable, and modular, enabling quicker exploration with deep neural networks. It uses the Middleware library to resolve low-level calculations because it cannot manage them.

4. **K Nearest Neighbor:** one of the most fundamental but crucial categorization methods in machine learning is K-Nearest Neighbors. It falls under supervised learning and is heavily utilized in pattern recognition, data mining, and intrusion detection.

5. **OpenCV Python:** openCV is a sizable open-source library for image processing, machine learning, and computer vision. OpenCV supports Python, C++, Java, and many other programming languages. It can analyze pictures and movies to find faces, objects, and even human handwriting. All operations that can be performed with NumPy may be coupled with OpenCV.

Methodology and Model Specifications

The two objectives of this work are to a) develop a program that converts a wireframe into code; b) assess the effectiveness of deep learning, and traditional computer vision approaches. Our program was only allowed to operate on wireframes created using a black marker over a white backdrop. This is reasonable because wireframes are frequently built on paper or whiteboards. Our software was supposed to generate and display code instantly. We developed two strategies using traditional computer vision and deep learning methods to accomplish the aim. This section explains our dataset first, followed by our generic framework, which can produce the website using either way after accepting a picture of the wireframe and pre- and post-processing. Then, we go over each step of the approach. Traditional computer vision in this part, we go over our standard procedure for transforming an image of a drawing into code, which mainly relies on computer vision. This strategy includes four essential phases: Computer vision may be used for element detection to identify and categorize the locations, dimensions, and kinds of every element in the drawing. Necessary for generating identical HTML components. Create a hierarchical tree using the list of all items using structural detection. Necessary for accurately replicating the HTML element tree. Classify the various sorts of container structures, such as headers and footers.

Dataset Training

It would be best to have a big dataset with plenty of examples to use deep learning techniques. The dataset includes code and trained drawings. Human blunders and divergent viewpoints on the proper drawing method might help determine the dataset's quality. We consider three methods for generating the dataset: finding websites and manually drawing them, manually drawing websites, and manually creating matching websites, and finding websites and automatically drawing them (James A. Landay, 1995). To maximize accuracy on our test set, we tuned our hyper-parameters to our dataset using standard methodologies and trial and error. Two MLPs coupled together make up

our model. One MLP is trained and taught to categorize using the container's x, y, width, and height. The other is taught to categorize using the sub-element element's kinds. The categorical element types are binarized using one hot encoding. The final classification is created by combining both results into a final MLP (Figure 1). This model was created to give the network the best possible chance of success. We considered the Tanh (Nasser M Nasrabadi, 2007) and ReLU (Vinod Nair and Geoffrey E Hinton, 2010) activation functions for hidden layers.

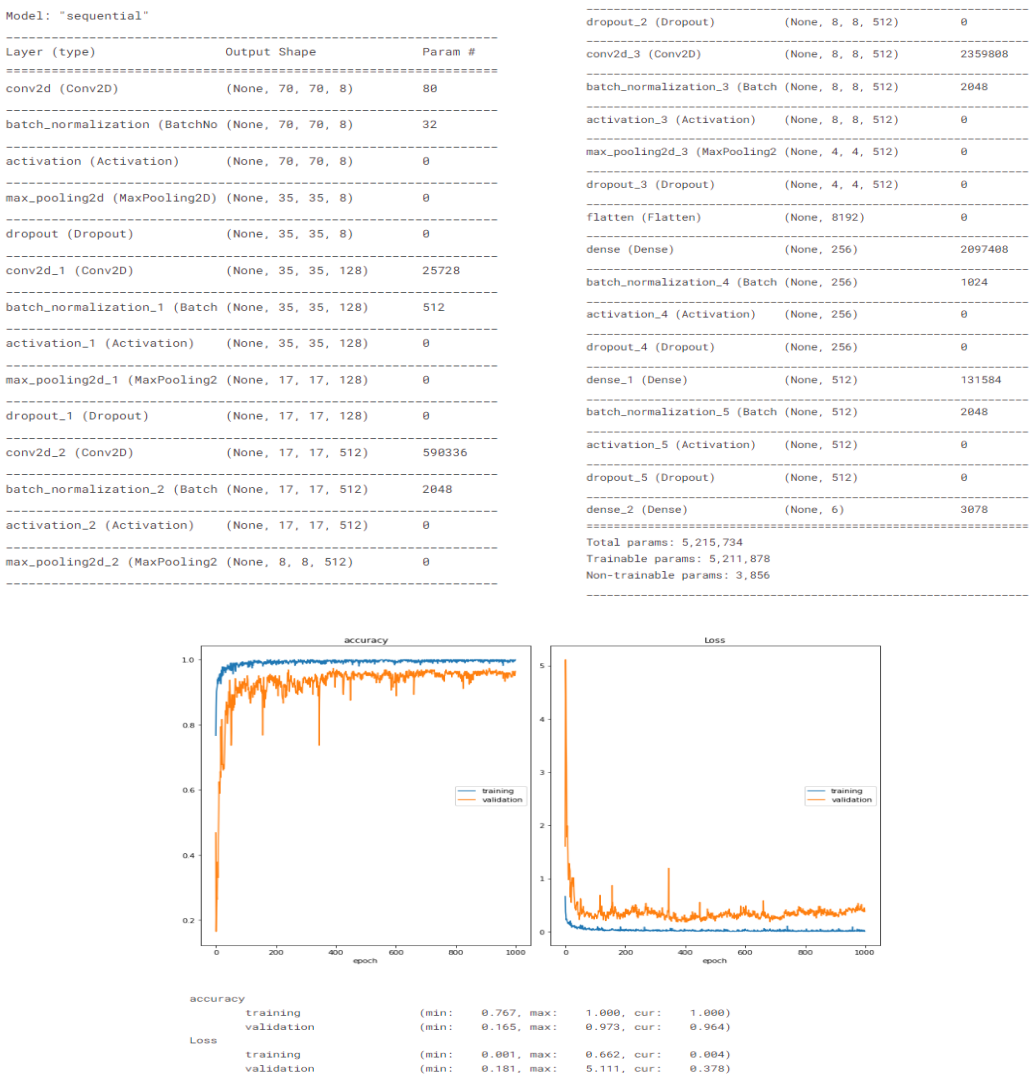

Figure 1: (*Dataset Training*)

Conclusion

To create a tool that converts a wireframe design into a website and to compare deep learning to traditional computer vision techniques for this purpose. The study that has already been done has been expanded in this work to include the innovative field of wireframe-to-code conversion.

Wireframes are converted into websites using an end-to-end framework that generates outcomes instantly. We describe how our framework was created to be simple to use: by enabling the use of photographs taken using webcams or mobile devices, hosting the generated webpage for collaboration, and employing widely used wireframe symbols to reduce the need for special training. A dataset and tools for recreating or expanding the dataset have been available. Two strategies have been devised for us: a traditional computer vision strategy and a 49-strategy utilizing deep feature extraction networks. Our deep learning method employs an innovative approach by training on fabricated wireframe designs for websites. To assess how successfully a system converts a wireframe that has been sketched into a website, we lastly developed repeatable empirical approaches. As a result, we believe we have accomplished our two objectives. According to our evaluation, both techniques we created did not perform well enough to be employed in production contexts. However, we contend that our dataset, infrastructure, and assessment methods will significantly impact the area of design-to-code methodologies. Our dataset sketching method makes it possible to do empirical evaluations on converting drawings into code, which was impossible before our study. Additionally, we were not knowledgeable of any deep learning applications in this issue area. We anticipate that our publication and the availability of our dataset and methodology will encourage more research in this area.

References

Pedro Campos and Nuno Nunes. "Practitioner Tools and Workstyles for User-Interface Design". In: 24 (Feb. 2007), pp. 73–80.

James A. Landay and Brad A. Myers. "Interactive Sketching for the Early Stages of User Interface Design". In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. CHI '95. Denver, Colorado, USA: ACM Press/Addison-Wesley Publishing Co., 1995, pp. 43–50. ISBN: 0-201-84705-1. DOI: 10 . 1145 / 223904 . 223910. URL: HTTP : //dx.doi.org/10.1145/223904.223910.

T. Silva da Silva et al. "User-Centered Design and Agile Methods: A Systematic Review". In: 2011 Agile Conference. Aug. 2011, pp. 77–86. DOI: 10.1109/AGILE.2011.24.

J. A. Landay and B. A. Myers. "Sketching interfaces: toward more human interface design". In: Computer 34.3 (Mar. 2001), pp. 56–64. ISSN: 0018-9162. DOI: 10.1109/2.910894.

James Lin et al. "DENIM: Finding a Tighter Fit Between Tools and Practice for Web Site Design". In: (Apr. 2000), pp. 510–517.

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). Nov. 2015, pp. 248–259. doi: 10.1109/ASE.2015.32.

Chao Dong et al. "Image Super-Resolution Using Deep Convolutional Networks". In: CoRR abs/1501.00092 (2015). arXiv: 1501.00092. url: http://arxiv.org/abs/1501.00092. .

Oriol Vinyals et al. "Show and Tell: A Neural Image Caption Generator". In: CoRR abs/1411.4555 (2014). arXiv: 1411.4555. url: http://arxiv.org/abs/1411.4555.

Andrej Karpathy and Fei-Fei Li. “Deep Visual-Semantic Alignments for Generating Image Descriptions”. In: CoRR abs/1412.2306 (2014). arXiv: 1412.2306. url: http://arxiv. org/abs/1412.2306.

S. Singh and B. Singh. "Effects of noise on various edge detection techniques". In: 2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom). Mar. 2015, pp. 827–830.

Nasser M Nasrabadi. "Pattern recognition and machine learning". In: Journal of electronic imaging 16.4 (2007), p. 049901

Vinod Nair and Geoffrey E Hinton. "Rectified linear units improve restricted Boltzmann machines". In: Proceedings of the 27th international conference on machine learning (ICML10). 2010, pp. 807–814.