Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 11 Issue 11, November-2022

AuFED - Automated Front End Development

Dr. Bhakti Palkar
Professor, Computer,
K. J. Somaiya College of Engg
Vidyavihar, Mumbai, India

Mr. Abdeali Arsiwala
Student, Computer
K. J. Somaiya College of Engg
Vidyavihar, Mumbai, India

Mr. Deep Doshi
Student, Computer
K. J. Somaiya College of Engg
Vidyavihar, Mumbai, India

Raj Nathwani
Student, Computer
K. J. Somaiya College of Engg
Vidyavihar, Mumbai, India

***Abstract*— Transforming a user interface screenshot developed by a designer into computer code is a typical task conducted by a developer in order to build customized software, websites.**
Large corporations have the workforce to dedicate separate teams for the design and development process, which can take several weeks and involve multiple stakeholders to back them up financially.
Small businesses and startups may lack such big resources. As a result their user interfaces and ultimately their product may suffer, causing them monetary losses.
Goal of this project is to use modern Deep Learning algorithms such as Convolutional Neural Networks [CNNs], Recurrent Neural Networks [RNNs] etc to significantly streamline and automate the design workflow by converting hand-drawn web page designs into HTML code and empower any business to quickly create and test web pages.

Keywords*— *GUI, automated HTML code generation, CNN, RNN, GRU, frontend

I. INTRODUCTION

Creating user-friendly and engaging experiences is the basic objective for companies of all sizes and a process driven by rapid prototyping, design, and user testing cycles. They need to go through an assortment of stages, including drawing concept sketches, designing prototypes, and testing the website prior to running it live. These procedures are not going to be completed right away. Truly, engineers spend weeks and months developing a beautiful, responsive website. Enormous associations such as Facebook have the resources to commit entire teams to the design process, which may take several weeks and involve several stakeholders; small organizations may not have these resources and can endure as a result of their user interfaces.

But technological advances are making it simpler for them. Current innovations, for example, Artificial Intelligence (AI) and Machine Learning are driving front-end improvement and making the coding and testing of site formats less complex, faster, and more powerful. Deep Learning, a component of Machine Learning, in particular, plays a crucial role in front-end development.

Taking motivation from this, we have implemented 3 components for automating the front-end development.

- Deep Learning models [CNNs, GRUs].
- Compiler.
- Interface.

The Deep Learning models are used to identify the elements of the web page and generate the Domain Specific Language [DSL]. The Compiler is used to convert the DSL to form the HTML code. The Interface allows the developer to interact with the models, specify the color schemes and acquire the output code. This entire flow of HTML Code generation from a hand drawn web sketch is depicted by figure 1

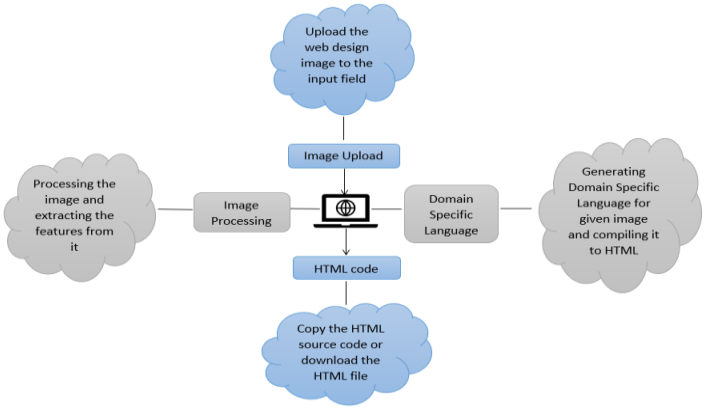

Fig. 1. Product Perspective

II. RELATED WORK

Youssef et. al [1] designed a system to develop HTML code using W3C XML Schema and Style Sheet.
Matija et. al. [2] developed a system which was a desktop-to-web converter. The application converts layout data and form event data to generate JAVA web applications.
Dakhore et. al. [3] designed a system which takes flowchart as an input and uses XML parser to generate C code. The XML Code is generated using Tree Traversal APIs of CDATA. It is a very primitive approach and cannot be implemented for complicated applications.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 11 Issue 11, November-2022

Aparna et. al. [4] developed an approach to design an HTML page from a hand drawn GUI. To identify the elements of the GUI, they have used height, width and diameter as comparing parameters. Back Propagation Neural Network (BPNN) and Learning Vector Quantization Neural Network (LVQNN) are used for character recognition. Discrete Cosine Transform (DCT) [5][6] is used to extract the feature for training neural network. The system is restricted to the identification of only uppercase labels being identified.

Tony Beltramelli et. al. [7] have proposed a method to generate HTML code based on the web page images and a Domain Specific Language. The proposed system makes use of the Convolutional Neural Networks model to extract information from the images and a Recurrent Neural Networks to convert the extracted information into a Domain Specific Language. Which is then used for the generation of the HTML code.

The Tech Giant Microsoft is deeply involved with AI to solve many problems. Microsoft AI Labs has developed a product called Sketch2Code et. al. [8] which implements Deep Learning techniques to extract information from the design made by the designer on the whiteboard. The information is then converted to the HTML code. The product architecture is made of Deep Learning models and has been trained on a dataset of millions of images, neither of which is open-source

TABLE 1 SUMMARY OF AUTOMATIC CODE GENERATION TECHNIQUE

Author	Input	Output
Youssef et. al. [1]	XML Schema and Style Sheet	HTML web interface
Matija [2]	GUI/TUI of desktop application	Java web application
Dakhore et. al. [3]	Flow chart	C program code
Aparna et. al.[4]	Scanned image of GUI drawn on paper	HTML page
Tony Beltramelli et. al. [7]	Coloured GUI images	HTML code
Akash Bharat Wadje, Prof. Rohit Bagh et. al. [8]	Hand Drawn Web Sketches	HTML code

III. IMPLEMENTATION OVERVIEW

Dataset Generation

We start with an open-source dataset used by Tony Beltramelli in pix2code [7]. It consists of 1,700 PNG screenshots of synthetically generated websites along with their corresponding GUI file which consists of tokens of Domain Specific Language. Since the main aim of this project is to work on hand-drawn images, the acquired images are processed and converted to hand-drawn like images using OpenCV and the PIL library in python. The images are resized to a specific aspect ratio. Then the borders of the elements are skewed and the border radius of the elements on the page are changed to curve the corners of the buttons and divs. The thickness of the borders are also adjusted to mimic drawn sketches. The font is changed to one that looks like handwriting. Finally we augment these images by adding skews and shifts.

The dataset used by Tony Beltramelli in pix2code [7] also includes a GUI file (corresponding to each PNG Web design image) which consists of DSL tokens for the HTML file generation. These tokens are used as input to a compiler for generating required HTML tags as detected in the image by the Deep Learning models. These files are built using a vocabulary which is made up using a small set of elements such as:-

,	{ }	btn-active	btn-inactive
small-title	text	<START>	btn-orange
quadruple	double	<c>	btn-green
row	header	btn-red	single

The purpose of this DSL is to make the code generation more efficient. The simplicity of these DSL tokens reduces the size of the total vocabulary of the language decoder thus reducing the size of the search space.

The 1,700 dataset items are then split into training (1360) and validation (340) sets.

Deep Learning Models

We then develop 3 models using Tensorflow and Keras as shown in figure 2 to achieve the desired results

First, A Convolutional Neural Network (CNN) model is used to extract image features and data from the source images. This data includes the information of the elements present in the image.

Second, A language model built using Gated Recurrent Unit (GRU) is used to encode sequences of source code tokens.

Third, A decoder model (also a GRU), which takes in the output from the models in the previous two steps as its input, and predicts the next DSL token in the sequence.

Once the set of predicted tokens is generated from the model, we design a compiler which converts the DSL tokens into HTML code that can be rendered in any browser.

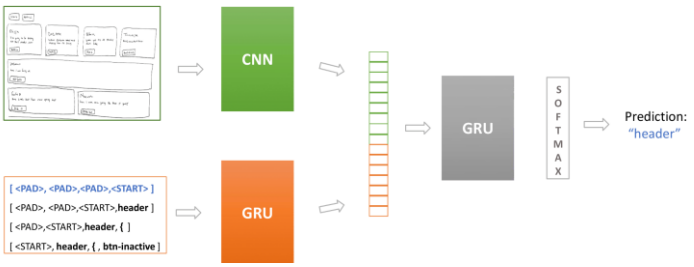

Fig. 2. Deep Learning Models

CNN - Image Encoder

This model was trained to detect the features of the images. In this case the features are the elements present in the web page images. To encode each input image to a fixed-size output

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 11 Issue 11, November-2022

vector, we exclusively use small 3 × 3 receptive fields which are convolved with stride 2. The width of the first two convolutional layers is 16, followed by 2 more layers of size 32, followed by two layers of width 64, and finally width 128. All the layers have a "relu - rectified linear" activation unit. Two fully connected layers of size 1024 applying the "rectified linear" unit activation complete the vision model.

GRU - Language Encoder

We make use of a simple lightweight DSL to describe GUI elements. This model is designed to only work with the GUI layout of the web page, the different graphical components, and their relationships; thus the actual textual value of the labels is ignored. Using the DSL also reduces the size of the search space for the language generation, the DSL simplicity also reduces the size of the vocabulary. We implement the language model as a stack of two GRU layers with 128 cells each. This results in our language model performing token-level language modeling with a discrete input by using one-hot encoded vectors; eliminating the need for word embedding techniques such as word2vec that can result in costly computations.

GRU - Language Decoder

We train this model in a supervised learning manner by feeding an image and a contextual sequence of DSL tokens as inputs; and the token as the target label.

A CNN-based vision model encodes the input image into a vectorial representation. The input token is encoded by an GRU-based language model into an intermediary representation allowing the model to focus more on certain tokens and less on others. Both the vectors are concatenated into a single feature vector which is then fed into a second GRU-based model decoding the representations learned by both the vision model and the language model. The decoder thus learns to model the relationship between objects present in the input GUI image and the associated tokens present in the DSL code. Our decoder is implemented as a stack of two GRU layers with 512 cells each.

Compiler - HTML File Generation

This compiler converts the generated DSL tokens into an HTML file, which can then be rendered into any browser. The input to this compiler is the GUI layout / DSL tokens generated by the GRU-decoder model. Colours are hard coded in the compiler based on the styling option provided at the start of processing with the Default scheme being Black & White and other schemes being Blue, Pink, Green, Purple, Red, Brown, Yellow, Orange, Grey.

Training of models

An important factor that has to be taken into account while language encoding and decoding is the size or length *T* of the sequences used for training to train on long term dependencies.

After the empirical experiments conducted by Tony Beltramelli [7], the sliding window of size 48 was selected; in other words. Taking that into consideration AuFED's language model also makes use of sequences of length 48. The trade-off between the computational costs and the long-term dependencies learning was acceptable. In other words for every new token generation the model will consider the image features and contextual sequence of previous 48 tokens. The special tokens <START> and <END> are used to respectively prefix and suffix the DSL files signifying the start and end of DSL files respectively. Training is performed by computing the partial derivatives of the loss with respect to the network weights calculated with backpropagation to minimize the multiclass log loss:

$$L(I,X) = -\sum_{t=1}^{T} x_{t+1} \log(y_t)$$

With x_{t+1} the expected token, and y_t the predicted token. The model is optimized and the loss L is minimized w.r.t all the parameters of all layers in the CNN model and both GRU models. Models were trained with the RMSProp algorithm which gave the best results with a Learning Rate of $1e-4$. To prevent overfitting, the dropout regularization was set to 25% for the vision model after each max-pooling operation and at 30% after each fully-connected layer. In the GRU-based models, dropout is set to 10% and only applied to the non-recurrent connections. Our model was trained with mini-batches of 64 images.

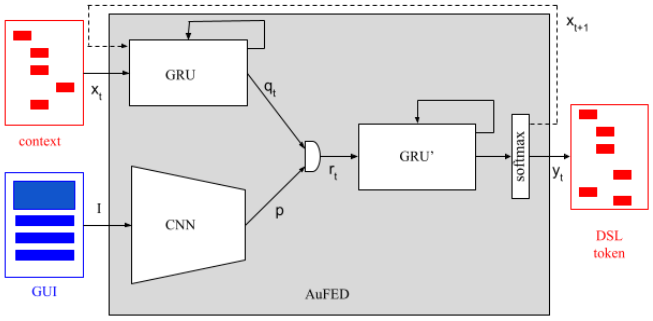

Fig. 3. Model Training

As we can see in Figure 4 which denotes the training and validation loss at the end of each epoch. Both the training and validation losses decrease with each epoch denoting that the models are becoming more and more accurate in learning about the features of the input image and predicting the next token for the GUI File. Also the validation loss is decreasing which suggests that the models are performing properly on unseen data as well.

Fig.4. Training and Validation Loss

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 11 Issue 11, November-2022

Sampling

To generate DSL code while sampling, we feed the GUI image I and a contextual sequence X of T = 48 tokens where tokens xt . . . xT −1 are initially set empty and the last token of the sequence xT is set to the special < START > token. The predicted token yt is then used to update the next sequence of contextual tokens. That is, xt . . . xT −1 are set to xt+1 . . . xT (xt is thus discarded), with xT set to yt. The process is repeated until the token < END > is generated by the model. The generated DSL token sequence is then fed to the compiler to convert it to HTML code.

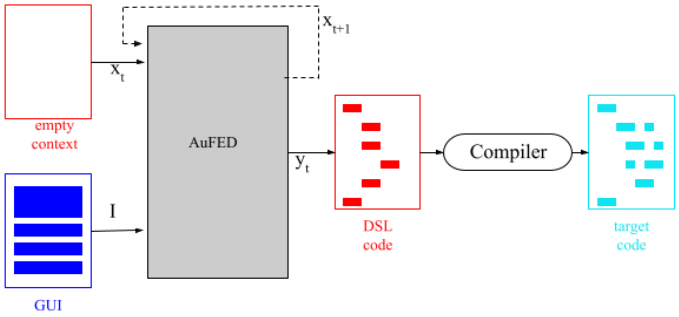

Fig. 5. Model Sampling

User Interface

In order to make this project accessible to all and easy to use, we develop a Web Interface which doesn't require any external dependency as such. Using simple HTML, CSS, and Javascript, we design a UI for the AUFED web application. We use the Django framework (Python-based free and open-source web framework that follows the model-template-views architectural pattern.) as a backend service and for integrating the trained deep learning models with the frontend UI. The UI consists of 6 elements in total which are - Upload Image Button, Colour Scheme Dropdown, Submit Button, Preview button, Download button, Code Editor

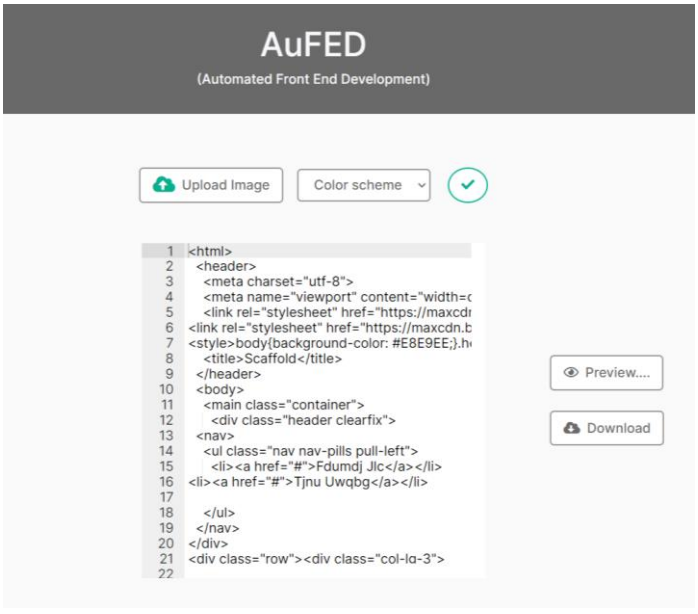

Fig. 6. AuFED User Interface

IV. Results & Discussion

In order to measure the accuracy of the models and the entire architecture we use the accuracy metric of BLEU or the Bilingual Evaluation Understudy Score. In simple terms the BLEU Score is a quality measure which calculates the difference between the machine translation and the human translation. It can be used to evaluate text generated for a suite of natural language processing tasks. BLEU Score compares the n-grams of the machine translation with the n-grams of the human translation. The greater the number of matches between both the translations the better the translation is. For every training epoch the validation loss of the models was decreasing which suggests that the BLEU Score is increasing. For calculation of the BLEU Score we use the NLTK Library's *sentence_bleu* function. Finally the trained models were evaluated on a few hand-drawn web page images and the results are as shown below.

Fig. 7. The hand-drawn test image.

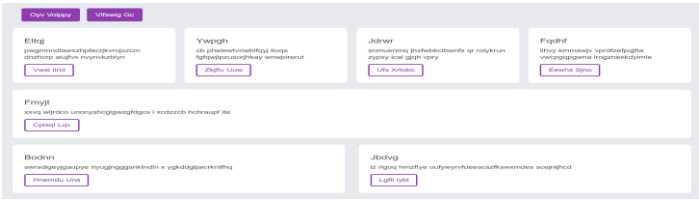

Fig. 8. The output generated by the model.

The figure 7 is the IMG-3 mentioned in TABLE-2 provided as an input to our model; and the figure 8 is the result generated by the models which gives an acceptable BLEU Score of *0.919*.

The BLEU Score was calculated and the result data is as shown in the Table 2

TABLE 2 BLEU SCORE FOR TEST IMAGES

Image File	BLEU Score	Time required for Output Generation (in sec)
IMG-1	0.77	26
IMG-2	0.66	28
IMG-3	0.919	30

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 11 Issue 11, November-2022

The model performed satisfactorily on unseen data as well. Though the model is not fit for real world applications, it still did a good job in its predefined scope and limited number of HTML elements..

V. Conclusion & Future Scope

In this paper, we have proposed a novel way of automatically generating the HTML code from hand-drawn website wireframes in a few seconds. We leveraged Deep Learning techniques such as CNN and GRU to convert hand drawn images into DSL tokens and finally used a compiler to convert those tokens to corresponding HTML code with the user-inputted color style. Our application also allowed the user to copy as well as render and view the HTML code in the browser.

Since the model was trained on a vocabulary of just 16 elements, which include only divs and buttons, it can't predict tokens outside of what it's seen in the data. Our model can be further improvised by generating additional website examples using more elements, such as dropdown menus, checkboxes, radio buttons etc and then training the model using a similar approach. Moreover, one can also implement Generative Adversarial Networks [GANs] for the same task. They are also capable of tackling the exposure bias problem of MLE and can generate better quality tokens for a DSL having larger vocabulary. GAN architecture consists of a Discriminator model which differentiates between the generated samples and the original samples. Thus the Generator model will create a DSL token file, from which the compiler can generate the HTML file and the discriminator can compare the final web page with the image provided in the input stage. GAN can handle many more parameters and generate better results as compared to other Deep Learning models.

References

[1] Youssef Bassil ,Mohammad Alwani, "Autonomic HTML InterfaceGenerator for Web Applications" International Journal of Web & Semantic Technology (IJWesT) Vol.3, No.1, January 2012,PP 33-47

[2] Matija Tomaškovic, Ruben Picek, "Automatic Conversion of Desktop Applications to Java Web Technology", Central European Conference on Information and Intelligent Systems, Varaždin, Croatia, September 19-21, 2012, Page 473 of 493

[3] Dakhore,Mahajan,"Generation of C-Code Using XML Parser"Available at http://www.rimtengg.com/iscet/proceedings/pdfs/advcom p/149.pdf

[4] Aparna Halbe, Dr. Abhijit R. Joshi, "A Novel Approach to HTML Page Creation Using Neural Network" International Conference on Advanced Computing Technologies and Applications (ICACTA2015), Procedia Computer Science 45 (2015) 197 – 204

[5] Anuja Nagare, "Licence Plate Character Recognition System UsingNeural Network", International journal of Computer Application, Volume 25– No.10, July 2011

[6] Dipti Pawade, Pranchal Chaudhari, Harshada Sonkamble, "Comparative Study of Different Paper currency and coin currency Recognition Method", International Journal of Computer Application (IJCA) , vol. 66, no. 23, Mar 2013, ISSN 0975-8887, pp. 26-31..

[7] Tony Beltramelli, "pix2code: Generating Code from a Graphical User Interface Screenshot", preprint, 2017

[8] Akash Wadje, Prof. Rohit Bagh, "Sketch2Code: From Sketch Design on Paper to Website Interface", IJIRT, 2020