

Web Components Template Generation from Web Screenshot

Pattana Anunphop

Chula University, Thailand, pattana.anun@gmail.com

Prabhas Chongstitvatana

Chula University, Thailand, Prabhas.c@clula.ac.th

AI-driven automation is the game-changer in this decade. The one concept that belongs to this domain is to simulate human working processes by using machine learning. An adaptation of this knowledge in web development is popularized topic in the web developer society. Moreover, Web Components, the new paradigm in software engineering practices in web development, becomes the new standard defined by World Wide Web Consortium (W3C). It is an essential building block for modularizing large and complex web applications into smaller pieces and then presenting them via the web browser on the user's computer or mobile. We combine knowledge between Computer Vision (CV) with deep learning and Web Components developer framework together to train the machine to recognize bounding boxes and category labels for each object of interest in an image. This paper introduces the methodology to automatically generate a website by neuron network model composite with many small web components. Our work's best result has a validation loss of 1.873, which can recognize the web object and transform it into the Web Components Template by React web framework.

CCS CONCEPTS • Computing methodologies~Artificial intelligence~Computer vision~Computer vision problems~Object detection • Software and its engineering~Software creation and management~Software development techniques~Automatic programming~Genetic programming

Additional Keywords and Phrases: Web components, Deep Learning, RetinaNet, Computer Vision

ACM Reference Format:

NOTE: This block will be automatically generated when manuscripts are processed after acceptance.

1 INTRODUCTION

Web Development was previously only known for the development of web pages and websites. However, nowadays, it is more like creating web applications. The complexity of development always increases and hard to maintain. W3C is an international community to develop web standards released the new approach to new standards to handle the complexity called "Web Components." Web Components proved themselves from many studies that can improve web development quality [10, 11, 18, 19]. It will separate each element that composite the web pages, websites, or web applications into small pieces and easy to maintain. Web components consist of three core technologies 1. Custom Elements, 2. Shadow DOM, 3. HTML Template works together.

Each piece of the web element can transform into only one HTML tag call the Custom Elements. Shadow DOM technology will separate configuration or CSS from each piece of web elements object. The last HTML template concept will encapsulate the web elements object into only an HTML tag. Figure 1 shows how to write the code for generating the card element between generic HTML and Web Components concept. The card element will become only a tiny tag name. In this example, the name is app-card.

Object Recognition with deep learning can be used in many ways. For example, autonomous vehicles such as self-driving cars, face detection, security systems, cancer detection, and more. Combine two knowledge between Web Components and Object Recognition will lead to the AI-drive approach to automatically simulate web development working processes to create web mockups from the reference website.



Figure 1: Compare to create card elements between generic HTML and Web Components concepts.

2 RELATED WORK

Using machine learning techniques to an automatic generating computer program has been around for a long time. Furthermore, this research area still active and have many studies that propose different technologies or apply to a different domain. Deep learning had been used for this purpose. An example is DeepCoder [5] which integrates neural network architectures with search-based techniques to write the computer program.

Recently, the challenge to transform graphical user interface (GUI) mockups into computer programs has become a command practice for current software development. The pic2code model [4] transforms GUI screenshots into computer code. Transform GUI mockup into mobile apps [7]. Alternatively, from hand-drawn mockups into android code using the YOLOv5 Model [6]. Nevertheless, in this paper, we will be using web screens as input and transform them into computer code.

Web screenshot represents the website's final result on a computer web browser or mobile web browser. The big difference between web screenshots and web mockups or hand-drawn mockups is that we can control the object's shape, the number of the objects, the number of classes of the objects, and the size of the digital input image to train into the model in web mockups or hand-drawn mockups, but for web screenshots, we are using the actual object that presents on the screen to train the model. We cannot control anything, as we can see in Figure 2. So, each web component's classes will be varied, multi-scale, and in entirely different colors. The digital input image will be multi-size. Then the challenge is to teach the machine to recognize each class of web components object. To achieve our purpose, we will control the scope of the type of web we want to recognize as just only business information website in Thailand (the purpose of the website focus to present the company information) to control the number of classes present on the website. This concept will increase the possibility of successfully train our model.

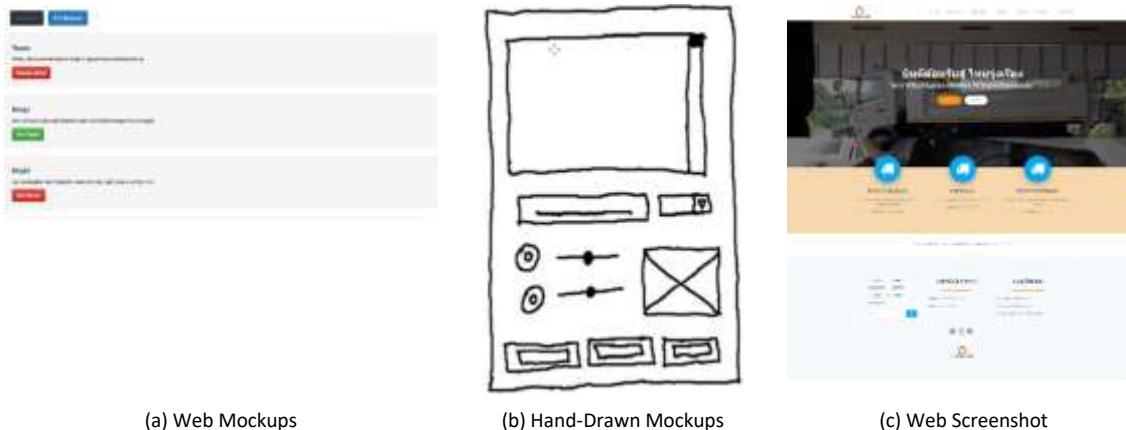


Figure 2: Difference between (a) Web Mockups, (b) Hand-Drawn Mockups, and (c) Web Screenshot.

3 METHODOLOGY

This paper uses the object detection model to predict each web component object's class and position in the web screenshot. Then transform predicted class and location to generate web components template. We separate methodology into three parts. The first part is to generate a dataset of web screenshots used to input our mode. The second part is the Deep Learning model for prediction. The last part is the process to generate a Web Components Template from the predicted result. The process is shown as the workflow like the following figure 3.

3.1 Datasets

For the object detection problem, the first and essential step is datasets in the format of digital images in the format of RGB color for three dimensions tensor. We need it for the train, validate and test process. However, there is no public web object dataset that is already labeled. So, we must generate datasets by ourselves. We are using the Google Chrome extension name FireShot to capture the entire page of the website. We focus only on the business information website for scope the number of components class. We have 34 classes and separate the class into three categories. The first group is the HTML5 Tag or standard element support by all web browsers, not needing to

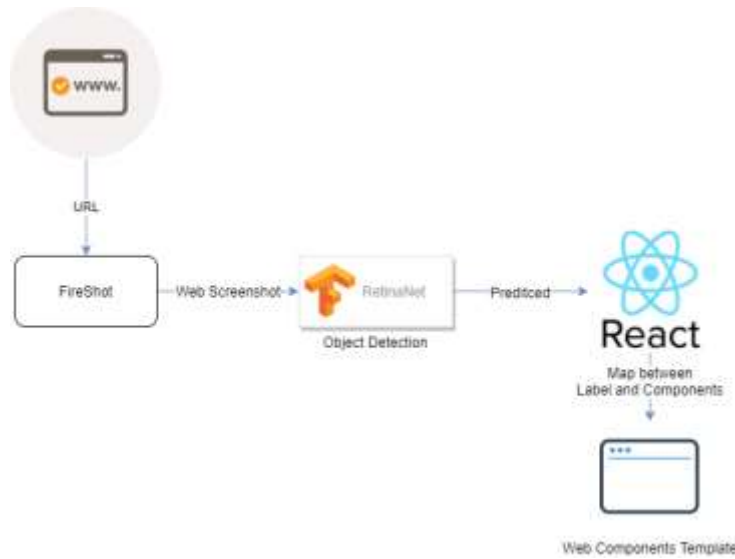


Figure 3: Web Components Template Generation Workflow.

handle anything on the object in this category. The second group is Bootstrap5 Tag, a custom-made readied from the popular web object library. The Complex Tag is the last category. It is the customized object that we must craft for support to generate web component templates. Bootstrap5 object and Complex object also count as the custom element. We label each component and show it in the following figure. Their mapping with the custom components is shown in table 1. Moreover, There is class0 for Not Found classes (the constrain from the deep learning model)

We are using the labeling tools name labelImg [8] to label our dataset. It generates in PASCAL XML format and then transforms into a tfrecord format using the tensorflow_datasets library. It is helpful to improve performance when train data with the TensorFlow machine learning framework.

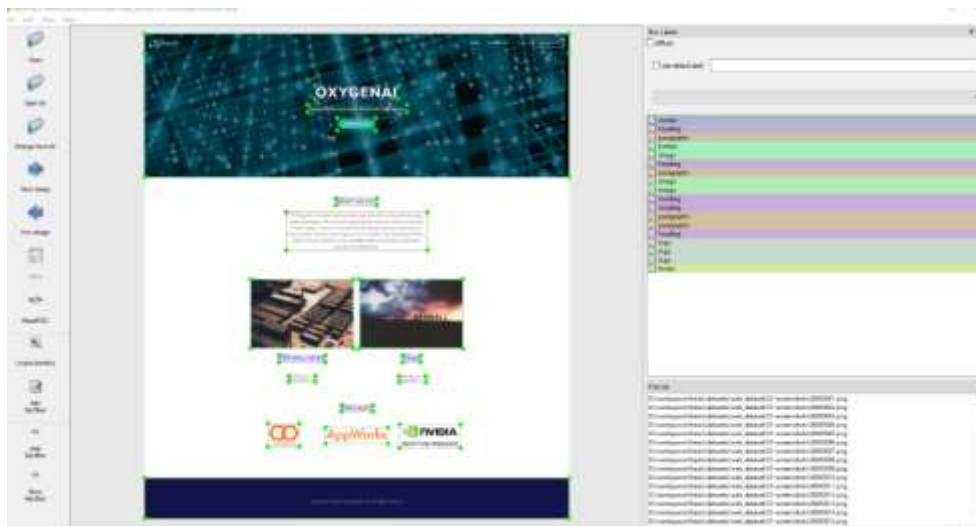


Figure 4: LabelImg tool for label our dataset.

Table 1: Class Table

Class ID	Class Name	Object Type	React Component Name
0	non-object	-	-
1	button	HTML 5	Button
2	footer	HTML 5	Footer
3	heading	HTML 5	Heading
4	image	HTML 5	Image
5	list	HTML 5	List
6	nav	HTML 5	Nav
7	nav_vertical	HTML 5	NavVertical
8	paragraphs	HTML 5	Paragraphs
9	button_with_icon	Bootstrap 5	ButtonWithIcon
10	button_with_icon_right	Bootstrap 5	ButtonWithIconRight
11	button_with_icon_top	Bootstrap 5	ButtonWithIconTop
12	card	Bootstrap 5	Card
13	carousel	Bootstrap 5	Carousel
14	dropdown	Bootstrap 5	Dropdown
15	form	Bootstrap 5	Form
16	icon	Bootstrap 5	Icon
17	list_group	Bootstrap 5	ListGroup
18	navbar	Bootstrap 5	Navbar
19	pagination	Bootstrap 5	Pagination
20	stat	Bootstrap 5	Stat
21	video	Bootstrap 5	Video
22	banner	Custom	Banner
23	calendar	Custom	Calendar
24	card_icon	Custom	CardIcon
25	copyright	Custom	Copyright
26	facebook_card	Custom	FacebookCard
27	facebook_chat	Custom	FacebookChat
28	google_map	Custom	GoogleMap
29	logo	Custom	Logo
30	profile_image	Custom	ProfileImage
31	qr	Custom	Qr
32	search	Custom	Search
33	star_rate	Custom	StarRate

3.2 Deep Learning Model

For generating a web components template, we want three pieces of information for each web element's components name, size, and location. The model will provide the predicted class map to the web components name and predicted bounding box transform to width, height, and position of the web components object. Since R-CNN is a popularized two-stage object detection framework, the first stage will generate a sparse set of candidate object locations, and the second stage will classify each candidate location. This technique is further developed to Fast-RCNN and Faster-RCNN

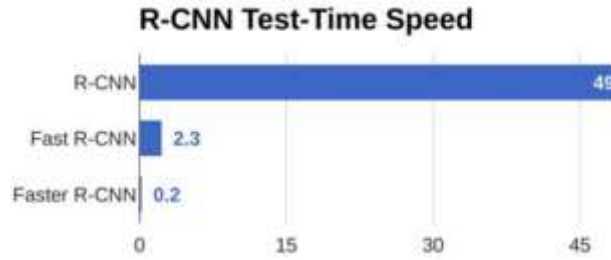


Figure 5: Compare test-time speed between R-CNN, Fast R-CNN, and Faster R-CNN.

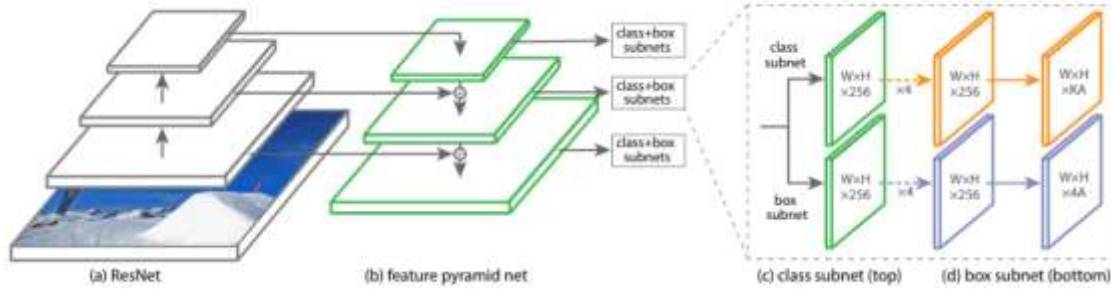


Figure 6: RetinaNet model architecture.

RetinaNet

RetinaNet is a one-stage object detection model that has high accuracy in the COCO challenge. Figure 6 show the model archirecter of RetinaNet. Moreover, better accuracy when compared with R-CNN, Fast R-CNN, and Faster R-CNN. Figure 7 shows the accuracy and speed of the object detection model with the COCO challenge. Moreover, this model has the crucial module that necessary for our paper. The first is Focal Loss [1] and the second Feature Pyramid Network [2].

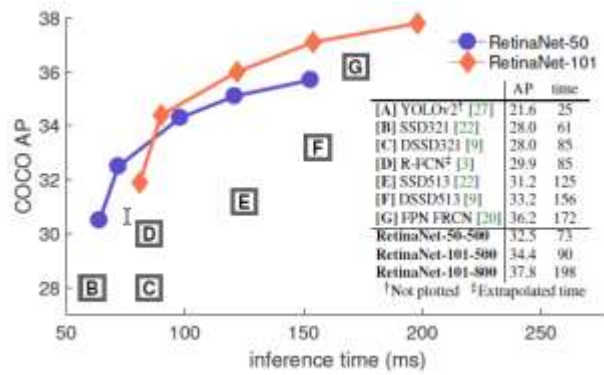


Figure 7: Speed (ms) versus accuracy (AP) on COCO test-dev.

Class imbalance is an obstacle to our work. It is impossible to have an equal number of web elements of each class. The Focal Loss functions more effectively for dealing with the extreme class imbalance. This function is a dynamically scaled cross-entropy loss.

Each web element in the website can be a variety scale. Therefore, Feature Pyramid Network (Fig. 8(b)) is the crucial layer. Feature Pyramids build upon image pyramid [3] (Fig. 8(a)) and be the fundamental component in recognition systems for detecting objects at different scales.

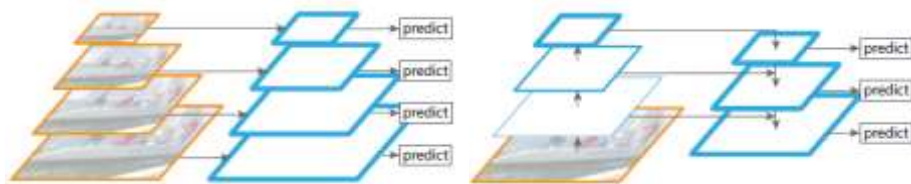


Figure 8: (a) Featured image pyramid. (b) Feature Pyramid Network

3.3 Web Components Template Generator Module

The results from object detection are used to generate Web Components templates. The result of the generator module is a class of the web components and the object's location that belongs to the website. React is the web development framework that supports building web components. It has many features that make it easy to prepare the web components for mapping with the result. In the mobile-first approach, web responsive must be the core and must be considered when generating Web Components templates. Bootstrap 5's grid system is the core concept of web layout, and this library also supports web responsive by itself. The row and column class concept set each component's layout that must generate in the template. Figure 9 shows the example of the Bootstrap 5' grid system. Some components will be one row as default like Carousal or Video components, and some components can be only the components in the column like Card or ButtonWithIcon components. Figure 10 shows the example of the one-row component, and Figure 11 shows the example multi-column components. We can also mix the various components to the same row, as shown in Figure 12. Our preliminary work has only one component per object class.



Figure 9: Example of Bootstrap 5's Grid System.



Figure 10: One-Row Components: Carousal.



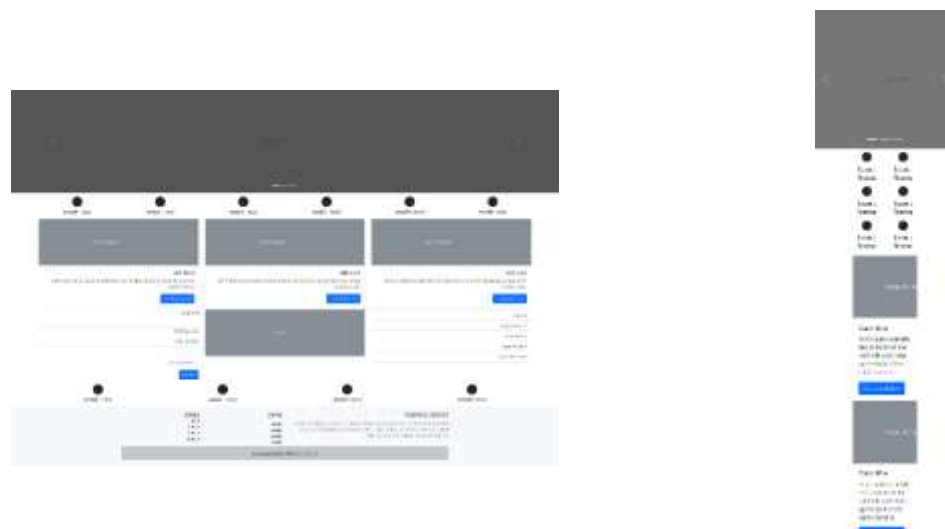
Figure 11: Example of multi-column components: a) ButtonWithIconTop and b) Card components.



Figure 12: Mix component in the same row with ListGroup, Image, and Form components.

4 RESULTS

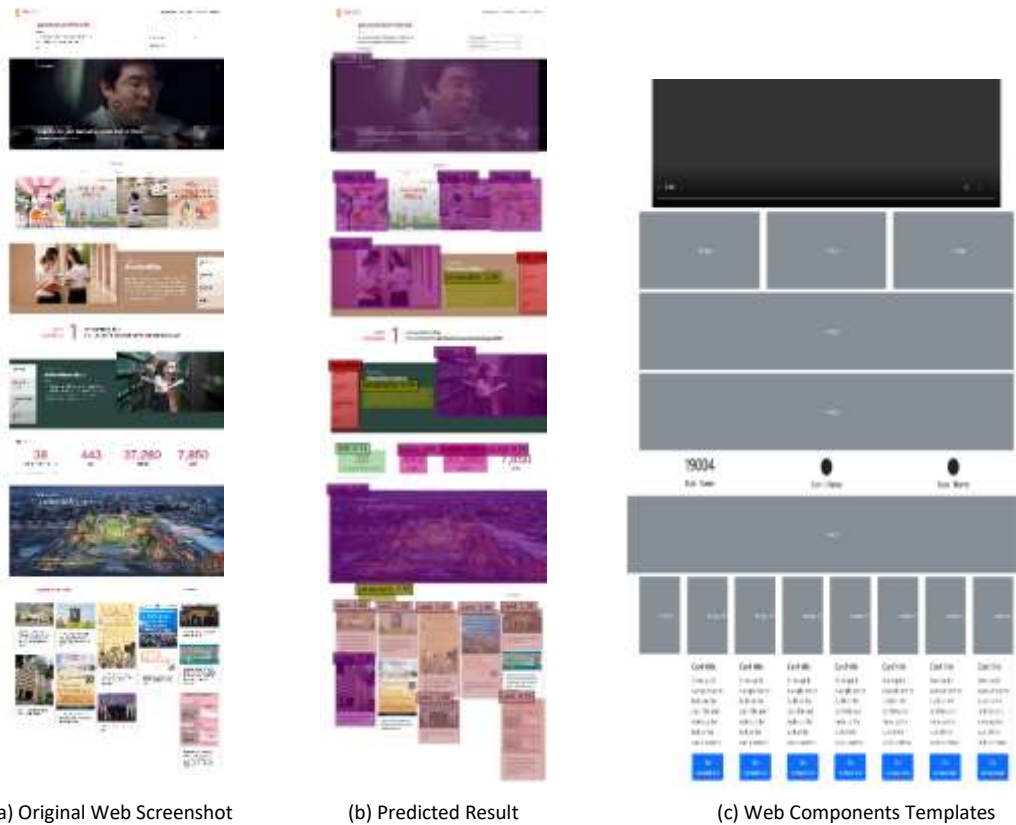
Our model can recognize the object of interest in the web screenshot, and our generator module can also transform the predicted result into a web components template. However, it not perfect. Some web elements still cannot recognize—for example, Navbar. Our best result from test datasets shows in Figure 14 below. Each component will have the virtual function interactive with the user as it used to be in a fundamental component, not just an image. Because of Bootstrap 5's grid system for the mobile-first approach, our web template can transform between the desktop and mobile versions shown in Figure 13.



(a) Desktop Version

(b) Mobile Version

Figure 13: Web Components Template in (a) desktop version and (b) mobile version.



(a) Original Web Screenshot (b) Predicted Result (c) Web Components Templates
 Figure 14: Example of compare between original web screenshot, predicted result, and web components templates.

We divided our dataset from 140 screenshots into train/validate/test 120/10/10 as mentioned in section 3 and train the model in more than one thousand epochs shown in figure 15. It has 3,433 objects in the dataset. From Table 2, show the imbalance of number of the objects that belong to classes. We are using the weight every time we have an epoch_loss reverse spike to predict the result, and we found epoch number 762 with validate loss 1.873 has the better result compared with the other, as shown in Table 3. We transform the result from the model into JSON format, including the list of the web element with class_id and bounding box of the object, and then create the module to read the JSON file to map with the React's Component that we prepare for generate the template. The critical algorithm is when we define the object in the same row by using the bounding box. If that object is not a one-row component, we will be using the bounding box's ymin and ymax to group the object to the same row. One interesting observation is that an image can be more than one class simultaneously, and we have the algorithm to calculate IOU between each predicted object and using the higher confidence score to choose an object and discard the lower score. We are using 0.5 confidence_threshold to filter out the object that not good enough to be in the result.

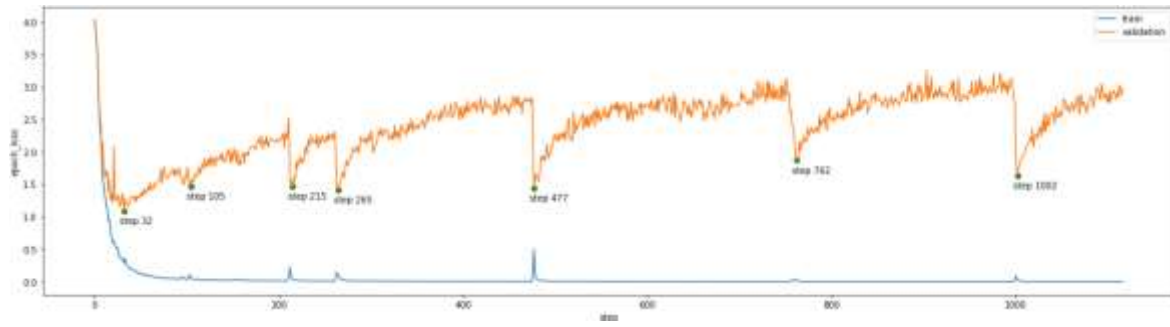









Figure 15: epoch_loss graph of train and validate the dataset.

Table 2: Number of the objects in Class

Class	Number of objects
paragraphs	1118
heading	1018
image	757
card	626
logo	281
button	272
button_with_icon_top	238
icon	237
navbar	121
nav	105
button_with_icon	95
footer	95
carousel	78
list	61
stat	58
banner	50
list_group	47
video	33
form	32
copyright	24
star_rate	19
google_map	15
facebook_card	10
button_with_icon_right	8
search	7
qr	6
profile_image	5
nav_vertical	4
pagination	3
card_icon	3
calendar	3
facebook_chat	2
dropdown	2

Table 3: Result of interested epoch

Epoch Step	Predicted Result	Loss	Number of Predicted Object
32		1.09	13
105		1.475	19
215		1.471	18
265		1.424	19
477		1.442	22
762		1.873	21
1002		1.633	22

REFERENCES

- [1] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Doll'ar. Focal Loss for Dense Object Detection. 2018.
- [2] T.-Y. Lin, P. Doll'ar, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection, in CVPR, 2017.
- [3] E. H. Adelson, C. H. Anderson, J. R. Bergen, P. J. Burt, and J. M. Ogden. Pyramid methods in image processing. RCA engineer, 1984.
- [4] Tony Beltramelli. pix2code: Generating Code from a Graphical User Interface Screenshot. arXiv:1705.07962v2, 2017.
- [5] M. Balog, A. L. Gaunt, M. Brockschmidt, S. Nowozin, and D. Tarlow. Deepcoder: Learning to write programs. 2016.
- [6] Abdelaziz A. Abdelhamid, Sultan R. Alotaibi, Abdelaziz Mousa. Deep learning-based prototyping of android GUI from hand-drawn mockups. IET Softw Vol. 14 Iss. 7, pp. 816-824. 2020.
- [7] Kevin Moran, Carlos Bernal-Cardenas, Michael Curcio, Richard Bonett, Denys Poshyvanyk. Machine Learning-Based Prototyping of Graphical User Interfaces for Mobile Apps. arXiv:1802.02312v2, 2018.
- [8] Tzu-Ta Lin, labelImg, (2021), GitHub repository, <https://github.com/tzutalin/labelImg>
- [9] Mohammad Farhad Aryan, Worarat Krathu, Chonlameth Arpnikanonndt, Boonrat Tassaneetrithep. Image Recognition for Detecting Hand Foot and Mouth Disease. 2020.
- [10] Andres-Leonardo Martinez-Ortiz, David Lizcano. A quality model for web components. 2016.
- [11] Rui Guo, Bin B. Zhu, Min Feng, Aimin Pan, Bosheng Zhou. CompoWeb: A Component-Oriented Web Architecture. 2008.
- [12] Rohan Mukherjee, Dipak Chaudhari, Matthew Amodio, Thomas Reps, Swarat Chaudhuri, Chris Jermaine. Neural Attribute Grammars for Semantics-Guided Program Generation. 2021.
- [13] Yuan Zhifeng. Human Body Tracking Method Based on Deep Learning Object. 2019.
- [14] Panagiotis Rizos, Vana Kalogeraki. Deep Learning for Underwater Object Detection. 2020.
- [15] Anandhavalli Muniasamy, Areej Alasiry. Deep Learning: The Impact on Future eLearning. 2020.
- [16] Niall O' Mahony, Sean Campbell, Anderson Carvalho, Suman Harapanahalli, Gustavo Velasco Hernandez, Lenka Krpalkova, Daniel Riordan, Joseph Walsh. Deep Learning vs. Traditional Computer Vision. 2019.
- [17] Xiaoyi Zhang, Lilian de Greef, Amanda Swearngin, Samuel White, Kyle Murray, Lisa Yu, Qi Shan, Jeffrey Nichols, Jason Wu, Chris Fleizach, Aaron Everitt, Jeffrey P. Bigham. Screen Recognition: Creating Accessibility Metadata for Mobile Applications from Pixels. 2021.
- [18] Markus Ast, Martin Gaedke. Self-contained Web Components through Serverless Computing. 2017.
- [19] Michael Krug, Martin Gaedke. SmartComposition: Enhanced Web Components for a Better Future of Web Development. 2015.