iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 8 **Issue:** III **Month of publication:** March 2020

DOI:

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 8 Issue III Mar 2020- Available at www.ijraset.com

DrawAI: Conversion of Sketch to Code using Faster R-CNN

Jyothirmai Kottu
Department of Computer Science and Engineering, Sridevi Women's Engineering College (India)

Abstract*: DrawAI is an efficient machine learning model developed using TensorFlow Object Detection API. A trained and experimented model which is developed to recognize the User Interface elements usually enclosed in HTML tags. The objective of this model is to convert mobile hand-drawn wire-frames from a picture into digital layouts and front-end code that can be used in starting stages of mobile application development. Faster R-CNN is used for object detection as selective search process is done with Regional Proposal Network, which makes it efficient than Fast R-CNN and R-CNN. Using this network entire training and testing of HTML elements are done using Transfer Learning.

Keywords: Region-Based Convolutional Neural Networks, TensorFlow, Object Detection API, HTML, Transfer Learning, Regional Proposal Network, User Interface, Machine Learning Model.

I. INTRODUCTION

The average time invested in application development is around 9-10 weeks. The developer usually spends vast amount of your time by writing code for every element which is employed in making the interface . An application developer must ensure reliability and scalability of the appliance . Front-End is that the key aspect of the client side of the online interface. A UI developer is liable for the interface visual, running and its operation.

They also work for application's user-facing code and therefore the structure of its user experiences. so as to execute these objectives, UI developers must be good at a number of the languages like: HTML, CSS, and JavaScript programming. Writing front-end code are often time-consuming, because it may be a similar process for several of the applications which are being developed now-a-days. it's the age of automation and web designing is additionally automating the planning and its development process. Faster R-CNN, uses Selective Search algorithm and trains the model completely.

The details about RPN will not be discussed in this paper, usually it is assigned with the task to output objects based on their "objectness" score.

In this paper, DrawAI is proposed to rework sketches of wire-frames as a sort of picture into layouts with code which are often used for mobile applications development. Users can sketch the front-end layouts on paper and upload images and further, a ML model processes these images to get the desired code of the layout. A model that predicts UI components that are present within the image and their location.

Using this approach, the appliance components are often developed from its sketches. For this, neural networks are employed and therefore the model is trained on a specially prepared dataset of images of UI sketches. The model identifies the appliance components and represents it as a JavaScript Object Notation (JSON).

II. LITERATURE REVIEW

Under literature review, mentioned are the varied research papers on object detection in images, Faster region-based convolutional neural networks are studied to notice the restrictions and advantages of their research.

A Review and an Approach for Object Detection in Images by Kartik Umesh Sharma* and Nileshsingh V. Thakur published in Int. J. Computational Vision and Robotics stated that Object detection (OD) system finds objects within the world by making use of the thing models which is understood a priori. In this paper, the varied techniques and approaches that are wont to detect objects in images and videos are taken in to consideration for present work. Object detector with region proposal networks like Fast/Faster R-CNN has shown the state-of-the art performance on several benchmarks. However, they need limited success for detecting small objects. The limitation is less performance of Fast R-CNN block in Faster R-CNN. Through this paper, the performance improvement of Faster-RCNN and efficiency in the detection of small objects in images helped to analyze the utilization of Faster R-CNN in proposed work.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 8 Issue III Mar 2020- Available at www.ijraset.com

III.PROPOSED SYSTEM

In this paper, a model is designed which is used for detecting the hand-drawn front-end layouts and generating the code for detected elements using JavaScript. An application is developed to detect the layout using camera and display the respective front-end code.

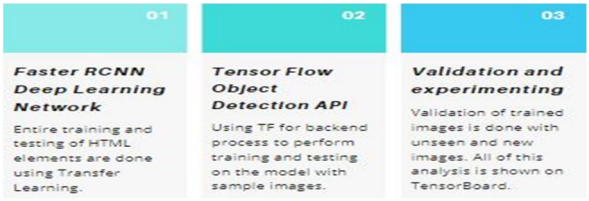


Fig. 1 Highlights of the proposed system

The proposed work is made to generate the code for the User Interface from a Sketch for understanding the objects in the image and applying meaningful explanations to them. The proposed system is divided into three modules. First module highlights the steps to detect objects from images using computer vision models. These objects present in the layout are elements of User Interface. The classifier highlights the bounding boxes with respective classes for regions where it identifies the objects in the sketch i.e. UI elements. The use of Faster R-CNN Deep Learning Network is highlighted in second module. The final module highlights about creating a working prototype application by implementing the UI components identified by computer vision model.

A. Machine Learning Model to Detect Objects from Images

The use of Convolutional Neural Networks (CNNs) has driven substantial growth in deep learning for computer vision, and in areas of object detection. TensorFlow, which is a deep learning framework, is used in this paper, to build Faster R-CNN architecture to automatically recognize objects in images. TensorFlow Object Detection API plays a key role in developing Machine Learning Model used for detecting objects, images and many other components. Training a dataset involves the below steps,

1) *Introducing a Dataset:* The model must contain more than 100 different images of UI sketches. In this paper, 5 types of components are labeled for every image. Components like text-box, image, toggle button etc. To train a model, it is required to create a database of training images. Initialize a database of more than 100 UI images and save them in the respective training images folder. After initializing, separate them into two sections for training and testing/validation. The bounding box labels for each component instance in each image is necessary, for that use a LabelImg to outline a bounding box for each component in each of training and validation images. A directory of .csv files containing the coordinates like xmin, xmax, ymin, ymax for each component instance that was manually annotated for every image.
2) *Preparing Images and Annotations:* After collecting data, immediately if the model is trained that leads to suboptimal results. There can be uncertainty with the data. If there aren't any problems, applying image augmentation stretches the dataset and decreases overfitting. Preparing images for object detection includes, but isn't limited. It is important to check if the annotations are correct and ensuring that the EXIF orientation of the photographs is correct (i.e. the photographs are stored on disk differently than how they're viewed in applications). To match the newly sized image resizing images and updating image annotations is ensured. Finally, it is important to efficiently maintain the health of dataset properties like class balance, size of images, and aspect ratios — and determining the impact on preprocessing and augmentations. Many color corrections which can increase the model performance like grayscale and contrast adjustments.
3) *TFRecords and Label Maps and Training Model:* Creating TFRecords could also be a file format that contains both images and their annotations. It's calibrated at basic dataset grounds, which makes it possible to form one set of records for training set, validation set, and testing set. The image and label training data are processed into TFRecord file format.

Once the train.tfrecords and val.tfrecords files are created for the dataset, the training begins on object detection model. Further, a category labels map text file that forms a link between the integer label to the text class label is documented. The three files created out of the entire dataset are:

a) *Training_Images:* Images that are used to teach the model. Together it forms a group of classes and respective bounding boxes for each class in this folder.
b) *Testing_Images:* Images in this folder are used to validate and evaluate the trained model and to make necessary predictions. This set doesn't have the classes and their bounding boxes.
c) *Training.csv:* This file consists of the name, class and bounding box of each image. There are usually multiple parameters for one image.

B. Faster R-CNN

R-CNN family consists of different algorithms(R-CNN, Fast R-CNN, and Faster R-CNN) listed in table 1, compared to other algorithms, Faster R-CNN has better performance and is fast in deploying a model. It follows a two-steps object detection mechanism : Identification of regions of interest, and passing these identified regions to convolution networks. The resultant features maps are sent for classification. Faster R-CNN, despite its name, has limitations as the model is comparatively slow than any other choices like YOLOv3 or MobileNet.

TABLE I
Convolutional Neural Networks

Algorithm	Features	Limitations
CNN	It divides the image into multiple regions and classifies each region into classes.	CNN requires inadequate amount of regions to predict with accuracy and thus it needs high computation time to do this task.
R-CNN	It performs selective search to detect regions. The algorithm draws out nearly 2000 regions from each image.	Computation time is adequate as each region is passed to the CNN separately. Also, it makes use of different models for predictions.
Fast R-CNN	Images are passed only once to the CNN and feature maps are extracted. Selective search is imposed on these extracted maps to make predictions.	Selective search is still slow and hence computation time remains unaffected (high).
Faster R-CNN	To increase the performance of algorithm, this model replaces the selective search method with region proposal network.	Object proposal is slow and the performance of systems depends on systems used before.

The steps followed by a Faster R-CNN algorithm to detect objects during a picture are:

1) An input image is considered and using it to the ConvNet returns feature maps for the image.
2) Applying Region Proposal Network (RPN) on these feature maps and to urge object proposals.
3) Applying ROI pooling layer to leverage all the proposals to the same size.
4) Processing these proposals to a connected layer for classification and prediction of the bounding boxes for the image.

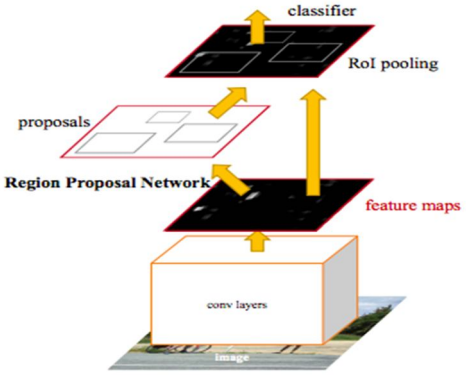


Fig. 2 Faster R-CNN Deep Learning Network

We minimize an objective function following the in Fast RCNN. Loss function for an image is defined as:

$$L(\{pi\}, \{ti\}) = 1 \text{ Ncls X i Lcls } (pi , p* i) + \lambda 1 \text{ Nreg X i } p * i \text{ Lreg } (ti , t* i)$$

C. Connecting ML model and JavaScript

The output model gives representation of the layout from the sketches drawn on the paper. The mobile application is integrated with this model where the user uploads the sketch images with desired layouts. The front-end representation layout is an object containing the sorts of the identified components of UI and their properties. For this work, a mobile application is deployed supported react UI library consuming a NodeJS server, a [9] WebSocket connection between the client and thus the server. .

IV. EVALUATION AND RESULTS

The generated file is an HTML document containing the UI components which can be run on any browser. The dataset we've prepared is aimed toward creating a group of fundamental building boxes of a interface . this is often often done by creating images consisting of 5 different classes of UI elements. These classes are Text, Text Box, Toggle Button, Button, Image. A group of 330 sketches, containing 1650 samples of elements, is used for the training process.

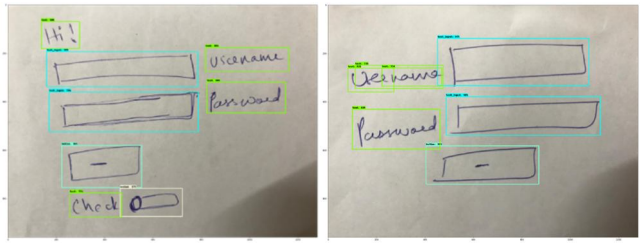


Fig. 3 Detecting UI components

A. Pre-Processing

The CNN model has two comma-separated value (CSV) files: the primary one is given for annotations, and thus the second one is for Integer Identifiers (ID). The annotation file contains six unique values: file-location, x-min, y-min, x-max, y-max, class; where x-min, y-min, x-max, and y-max are dimensions of the bounding box of the annotated UI element during a selected image. This file contains a separate record for every UI element within the image. The class file contains an inventory that maps an integer to classes present in the annotations file. Once the UI components are detected, the appliance renders the respective code for execution.

V. CONCLUSION

The presented work, DrawAI, demonstrates that we'll train a neural network to spot UI components in hand-drawn sketches. Performance, which is suffering from the training set, are often improved by providing more labelled samples of sketches. While analyzing, we evaluated the model on a gaggle of diverse images. The model learns about the structure of the UI components, and after fixing overlapped components, it gives a final UI representation object using the Mobile Application.

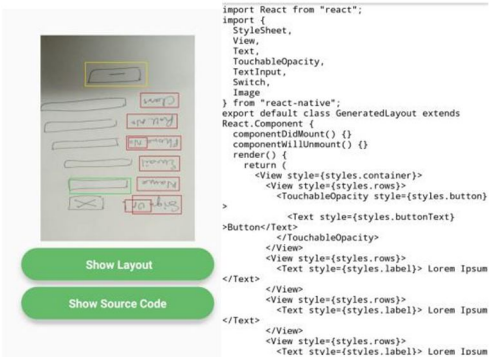


Fig. 4 Executable Code and Layout

A. Future Scope and Recommendations

Further, many more components can be added to the model and neural networks with better performance can be used to train the model.

REFERENCES

[1] Myers, S. E. Hudson, R. Pausch, and R. Pausch, "Past, present, and future of user interface software tools," ACM Trans. Comput.-Hum. Interact., vol. 7, no. 1, pp. 3–28, Mar. 2000. [Online]. Available: http://doi.acm.org/10.1145/344949.344959

[2] C. Szegedy, A. Toshev, and D. Erhan, "Deep neural networks for object detection," in Advances in neural information processing systems, 2013, pp. 2553–2561

[3] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov, "Scalable object detection using deep neural networks," in The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2014.

[4] G. Nagy, T. A. Nartker, and S. V. Rice, "Optical character recognition: An illustrated guide to the frontier," in Document Recognition and Retrieval VII, vol. 3967. International Society for Optics and Photonics, 1999, pp. 58–70.

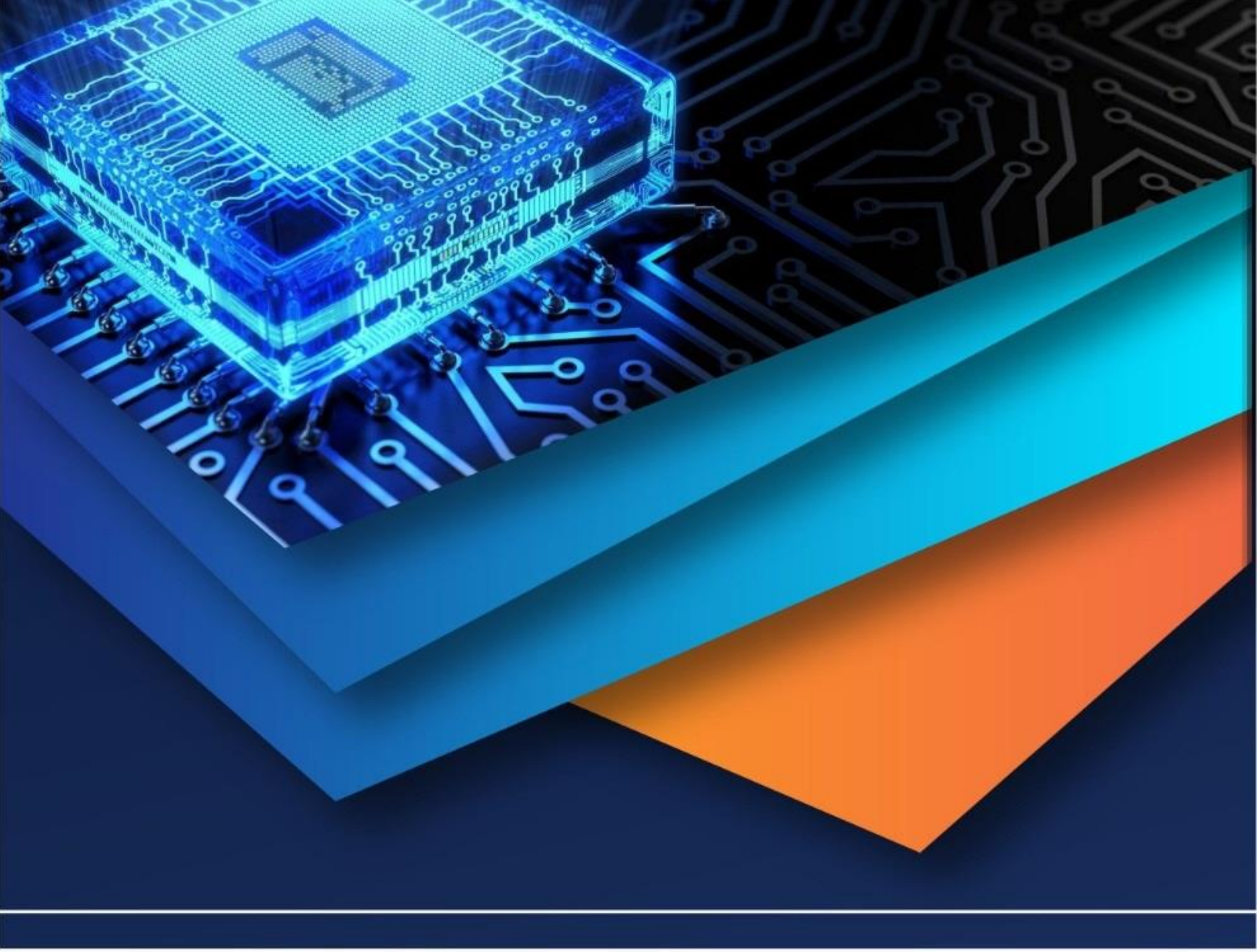
[5] W. Zhiqiang and L. Jun, "A review of object detection based on convolutional neural network," in Control Conference , 2017 36th Chinese. IEEE, 2017, pp. 11 104–11 109.

[6] T. Beltramelli, "pix2code: Generating code from a graphical user interface screenshot," in Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems. ACM, 2018, p. 3.

[7] Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks by Shaoqing Ren ; Kaiming He ; Ross Girshick ; Jian Sun published in IEEE Transactions on Pattern Analysis and Machine Intelligence (Volume: 39 , Issue: 6 , June 1 2017).

[8] Refining Faster-RCNN for accurate object detection by Myung-Cheol Roh and Ju-young Lee published in 2017 Fifteenth IAPR International Conference on Machine Vision Applications (MVA).

[9] I. Fette and A. Melnikov, "The websocket protocol," 2011. [Online]. Available: https://tools.ietf.org/html/rfc6455



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY