ML Final Project

Avi Garg (2017223)
Divyam Agrawal (2017230)
Yash Kalyani (2017273)

Abstract

An engaging and intuitive experience for customers is a desirable objective for many front-end developers. Large companies can afford to dedicate teams and resources for the design process but this is not the same for smaller companies. Hence, to save time and resources, we have worked to transform a graphical user interface sketch to a working bootstrap code for websites.

1. Introduction

A lot of designers suffer from the fact that they don't know how to code their designs, and thus, have to hire developers at high rates to test their initial designs. But the problems are not limited to only designers. As a developer, it is required to implement a working GUI code based on mockups provided by designers. Implementing GUI code is, however, time-consuming. There's a well-studied domain in machine learning called image captioning that seeks to learn models that tie together images and text, specifically to generate descriptions of the contents of a source image. Our ML project aims to prevent the above problems by using image captioning models and creating a working bootstrap code, based on an input of hand-drawn GUI sketch.

2. Related Work

Work on this field started with Tony Beltramelli's paper pix2code, which had similar motivation but worked on screenshots of websites, not hand-drawn mockups.

A similar venture, although vaster and more accurate, is Sketch2code, which was acquired from Airbnb by Microsoft recently. This program transforms any Hand-drawn mockups to a working HTML code.

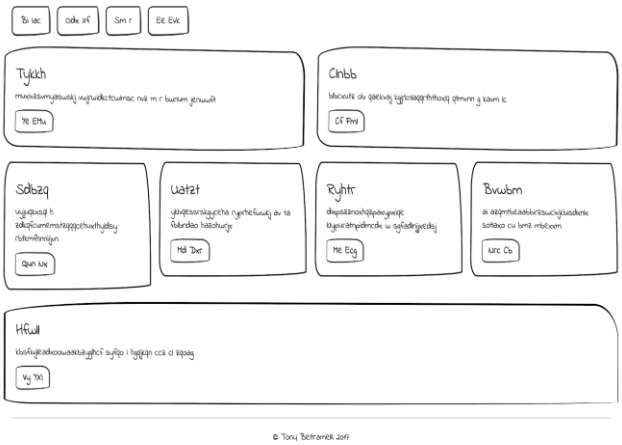
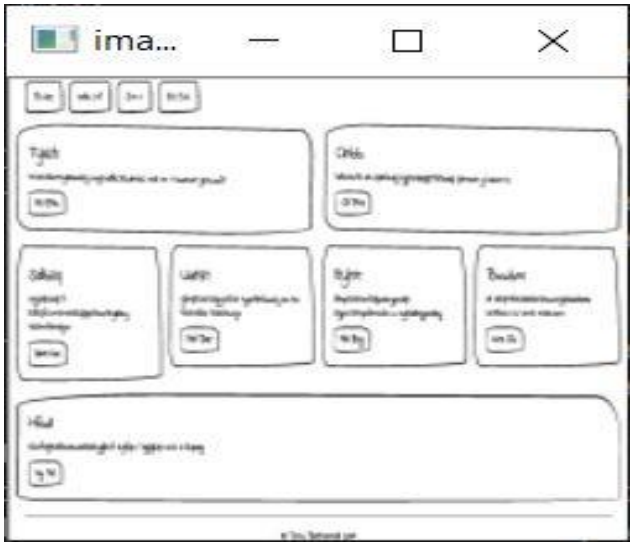


Figure 1a(above) and 1b(below): Original image above. Image after preprocessing.



3. Methodology

3.1. Dataset

We have used the dataset provided by Tony Beltramelli for his own project pix2code, and dataset by Ashwin Kumar, which created hand-drawn look-alike mockups of the website screenshots. The dataset we have consists of 1700 labelled sketches. We are using 20% of the dataset as Test data and 80% as Training data.

3.2. Feature Extraction

The training set is augmented using Keras ImagePreprocessing library, adding variations in the given dataset in terms of rotations, shifting and zoom. The corresponding data is then converted to grayscale and adapted for our custom model which is then resized to a 256 x 256 array for a single-color channel.

Our vocabulary, i.e., labels comprise of the given below classes, which are then tokenized using Keras Text Preprocessing library to give them their corresponding ids.

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

3.3. Evaluation Metrics

Sentence BLEU score was used for calculating the accuracy of our model, a method provided by nltk library as sentence_bleu() which compares combination of our model's hypothesis with the provided tokens i.e. different buttons, etc. with the actual value.

To evaluate the generated website, we need to rely on manual user evaluation.

4. Model Analysis

Using models based on Image Captioning, we designed our model architecture in three parts.

For the computer vision model, we used Convolutional Neural Network (CNN) to extract image features from our datasets. We have used Keras cross-entropy loss function in our model to calculate the overall train/val losses.

We simultaneously worked on a pre-existing CNN model AlexNet, designed by Alex Krizhevsky and also created a CNN model from scratch to extract image features from the dataset.

We compared the performance of both the models and found the accuracy obtained from our model to be far superior.

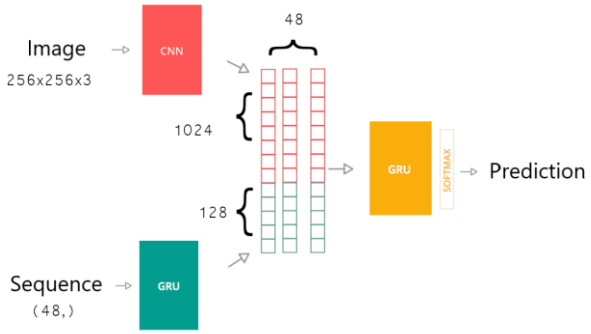


Figure 2: Model Architecture.

Since we are processing sequences, we will employ a Recurrent Neural Network as they can remember things learnt from prior inputs while generating the output.

Vanilla-RNN have a short term memory and thus to meet our demands of long sequences we decided to use the extended version of RNNs, Long Short Term Memory(LSTM) and Gated Recurrent Units (GRU).

For the language model, we tried both the GRU and LSTM models. After training both these models we found the GRU worked slightly better than the LSTM model.

Thus, we finally used GRU to encode sequence of source tokens.

For the predictor model, we used GRU again to predict the next token in the sequence based on the previous outputs.

The model was then trained for 10 epochs with the initial learning rate of 0.0001 and 64 pictures per batch (batch size).

We used RMSprop as our optimizer as it restricts the oscillations in the vertical direction which allowed us to increase our learning rate. The optimal hyperparameter was decided after analyzing multiple models.

5. Results

	Custom Model	AlexNet
GRU	93.34%	79.88%
LSTM	92.71%	79.32%

Figure 3: Accuracy report for all models on Testing set.

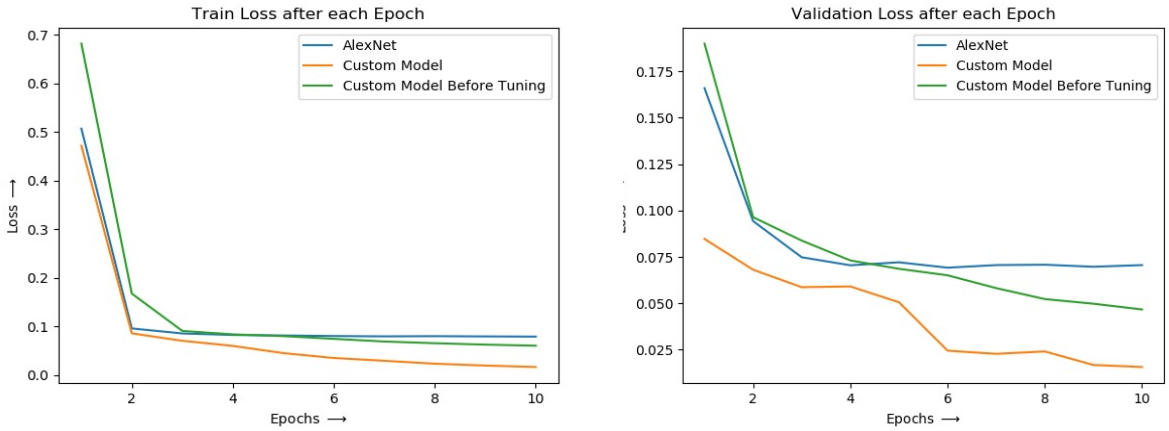


Figure 4a(left) and 4b(right): Loss after each epoch for the 3 models trained.

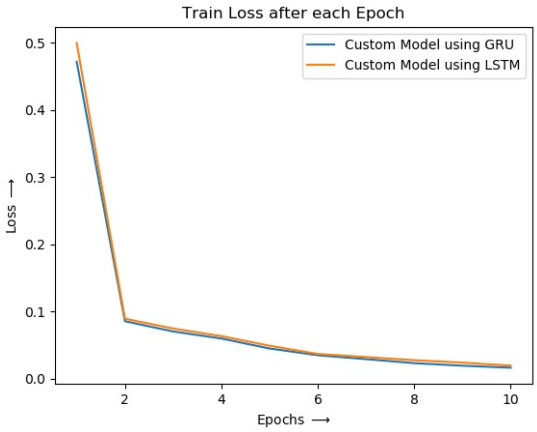


Figure 5: Loss after each epoch for the GRU and LSTM models.

6. Conclusion

We found that our model extracts image features with higher accuracy than previously. Since the number of classes (features) we have included is smaller than the commercially available products, our model doesn't perform well on any hand-drawn sketch.

Finally, we are successfully able to classify test image objects into their classes, and generate a working Bootstrap website.

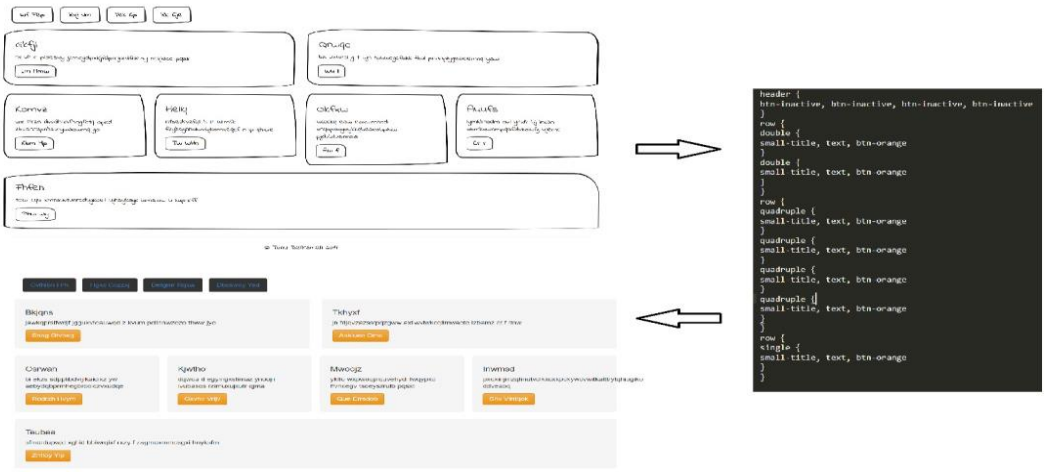


Figure 6: Final output of the model.