Classification and Detection with CNN Report

Shu liao
sliao32@gatech.edu

Abstract

The goal of this project is to use Convolutional Neural Network (CNN) models to detect and predict the numbers in a picture. This project use the Street View Hose Number (SVHN) data set[1] as the training data for three different CNN models: the author designed model, VGG-16, pre-trained VGG-16. The train and test accuracies are 94.00% and 93.52% for author designed model, 94.12% and 90.77% for VGG-16, 94.70% and 93.52% for pre-trained VGG-16. The project use sliding window method for detecting and classifying digits in real life images.

I. Training Method

A. Data Processing

The SVHN dataset has two format of data: original images with character level bounding boxes and MNIST-like 32-by-32 images centered around a single character. This project uses the latter one for training plus the non-number image cropped from the original images. As a result, there are 98,472 training images, 11 labels, of which numbers are labeled as their values and non-numbers are labeled as 10.

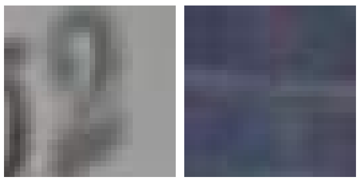

(a) number (b) non-number

Fig. 1: Examples of the training data.

B. CNN Architecture

To create a CNN model, I created a model similar to the description in a SVHN paper[2]. It contains 7 convolutional hidden layer, and 2 densely connected layer. The hidden layer contains a 2d convolution, a batch normalization, a relu activation, a 2d maxpooling and a dropout, see Fig. 2 for illustration. The first 3 hidden layer has a filter size 48 and a 3-by-3 kernel size for the 2d convolution; the maxpooling has 2-by-2 pool size and 1-by-1 strides, ecept the first layer has 2-by-2 strides; the dropout rate for each hidden layer is 0.2. The 4th to 6th hidden layer has filer size 64, and every other parameters are the same except the 4th hidden layer has a 2-by-2 strides. The 7th hidden layer has filter size 128, and 2-by-2 strides. After the 7 hidden layer 2 dense layer of size 64 is added, and finally an output of 11 features using softmax as activation function.

For VGG-16 model, two size 256 dense layers are add after VGG, and then a dropout at 0.2 followed by the output of 11 features using softmax.

The pre-trained VGG-16 is followed by two size 128 dense layers and an output of 11 features using softmax.

II. Detection Method

In order to detect the digits in the photo, I'm using different size of sliding window to identify if there is a number in the window. The detection contains 3 major stages: slide window at large step to find the location of the sequence of digits, crop the image around the area; slide window at smaller step to find the potential digits; merging the boxes that contains the same digit. Fig. 3 shows an example of these steps.

At first stage, the window size starts at half of the original image size, it slides over the image with a step of half of the window size. If there is a window has confidence larger than 99% by the trained classifier, then crop the image round this area by extending hight twice and width five times, enter into second stage; otherwise shrink the window size to $2/3$ of its size, repeat until window size is less than 32 pixels.

At second stage, the window size starts at the size of the cropped image, it slides over the cropped image with a step of $1/4$ of the window size. If there is a window has confidence larger than 85% by the trained classifier, then record this window. Shrink window size by $3/4$. Repeat four times.

At third stage, iterate through the recorded windows. If two window has the same number, and are overlapping each other at lease half of their area, then merge the two window to their average. Repeat until there is no overlapping windows.

After the above three stages, the numbers will be marked on the image.

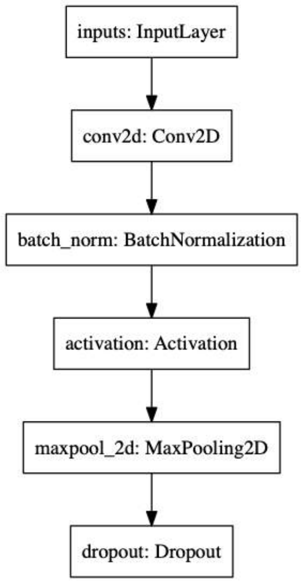
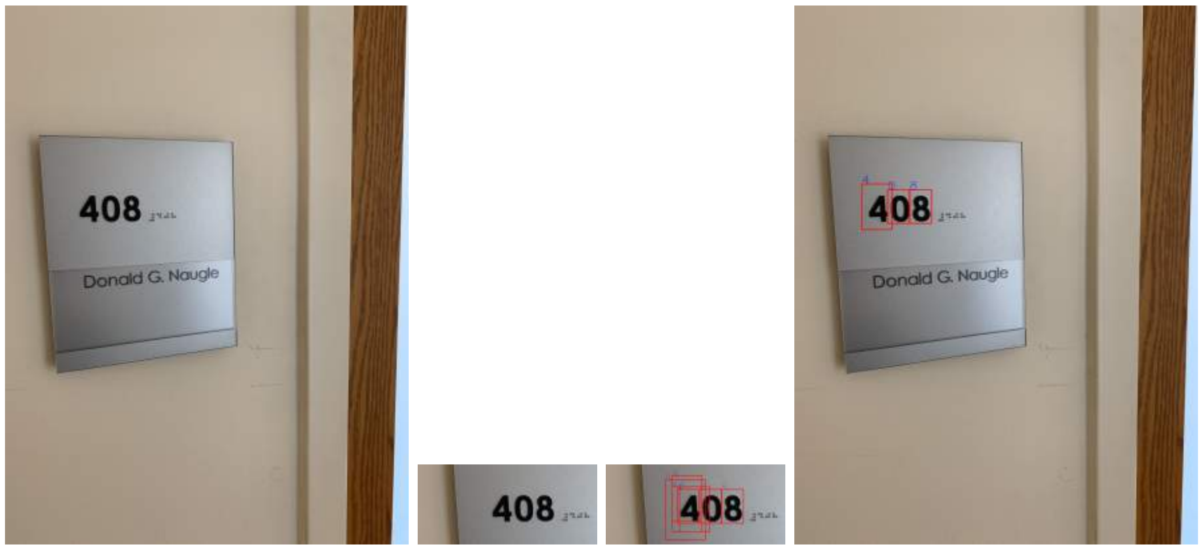


Fig. 2: The structure of the hidden layer used in this project



(a) origin image (b) cropped image (c) potential candidates (d) final result

Fig. 3: Different stages of the image.

III. Result

The detection and classification works and fails in different scenarios. It works as in Fig. 4 and it fails as in Fig. 5. There are many false positives during detection, especially then the window happen to slide between two digits. The problem may be solved by adding more non-number images to the data set.

IV. Performance analysis

I plot the loss and accuracy vs epochs for each model in Fig. 6 and Fig. 7. The designed model and pre-trained VGG-16 have similar performance while the VGG-16 has lower accuracy.

V. Discussion

The accuracy of the trained model looks to be promising ($\sim 94\%$). However the actual result of the detection suffers from severe false positive. For example, the letter ’M’ is often classified as ’4’. I have tried different threshold in the detection

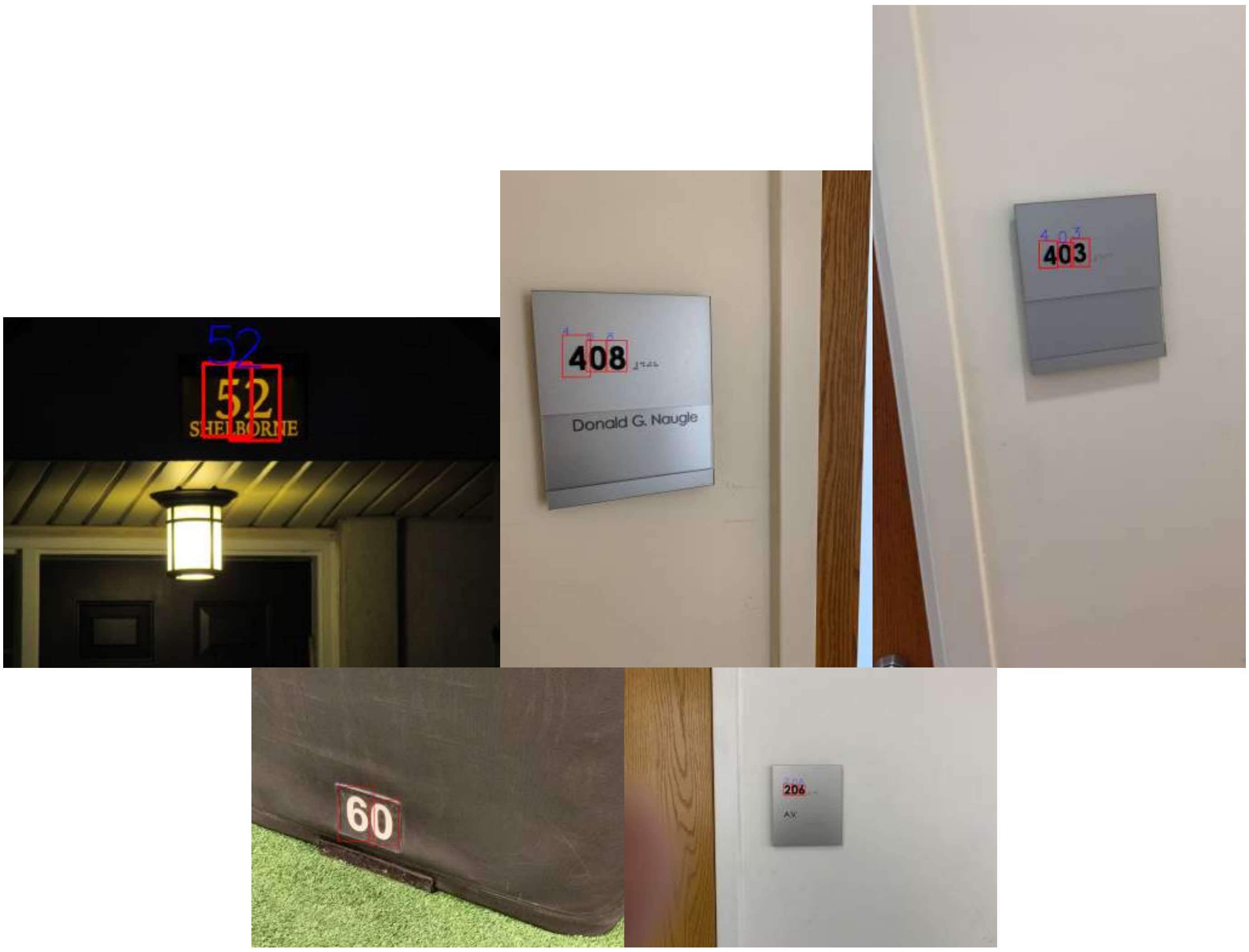


Fig. 4: Example images where the detection and classification works. Please refer to the submitted files for clearer images.

algorithms, it turns out that if I want less false positives then there are many digits which will not be detected (true negatives). The current parameters are the best in terms of balancing the false positives and true negatives.

Sliding window method is working slow due to the algorithm needs to slide the window over an image multiple times. Even after the aggressive stop and cut method as described in method section, the algorithm still take several seconds or even a minute to run on an image.

The project may be improve in both classification and detection side. At classification, a larger amount of data set, especially increasing the data for non-number image may significantly reduce the number of false positives. At detection, using segmentation instead of sliding window may help to reduce the running time. Another possible choice is to train a YOLO algorithm which is known to be fast at detecting objects in the image[3].

The model in the paper[2] for detecting a sequence of number has an accuracy of $\sim 96\%$. By trying to mockup their CNN model, my designed by achieved $\sim 94\%$ on the SVHN data set. However, the accuracy on real life image is much lower. An unsupervised way of learning on the same task is done by paper[1], their best accuracy is 90.6% on the data set.

VI. Links

Video demonstration: https://youtu.be/0Mb17M_dZFw   Presentation: https://youtu.be/fiI7e1mbme8

References

[1] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, Andrew Y. Ng. Reading Digits in Natural Images with Unsupervised Feature Learning. NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011

[2] Goodfellow I J, Bulatov Y, Ibarz J, et al. Multi-digit number recognition from street view imagery using deep convolutional neural networks[J]. arXiv preprint arXiv:1312.6082, 2013.

[3] Redmon J, Divvala S, Girshick R, et al. You only look once: Unified, real-time object detection[C]. Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 779-788.

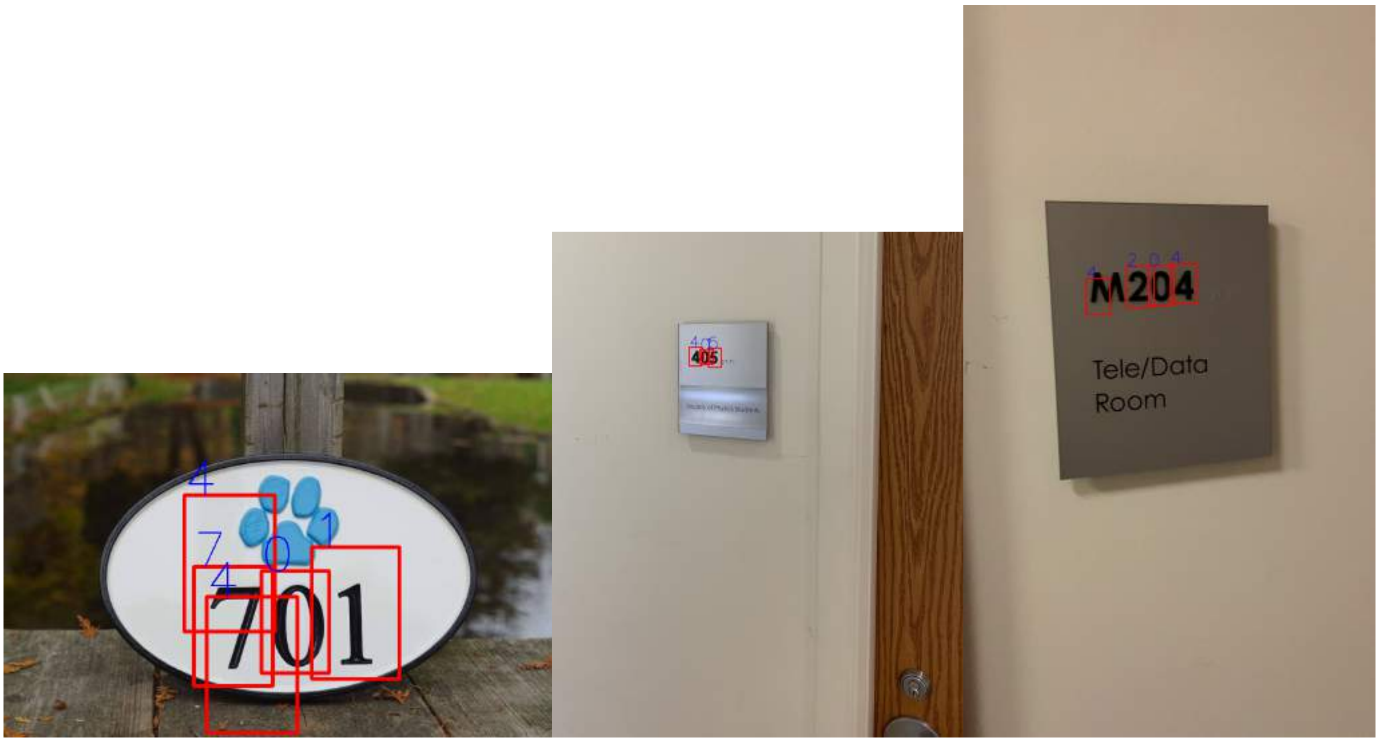
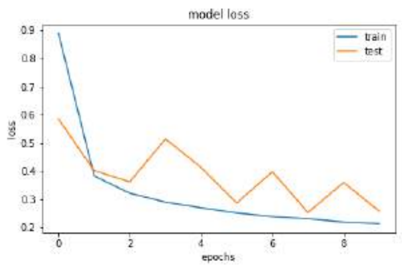
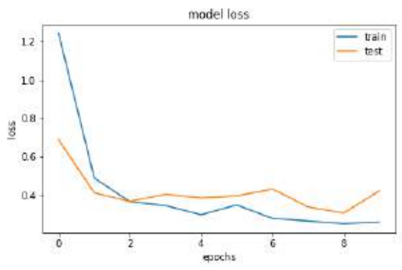


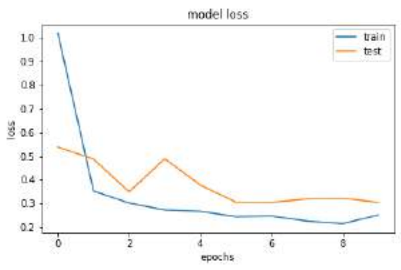
Fig. 5: Example images where the detection and classification fails. Please refer to the submitted files for clearer images.



(a) designed model

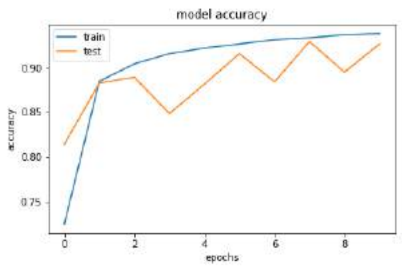


(b) vgg model

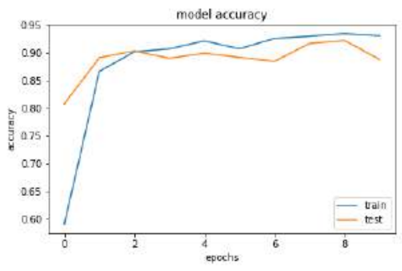


(c) vgg pre-trained model

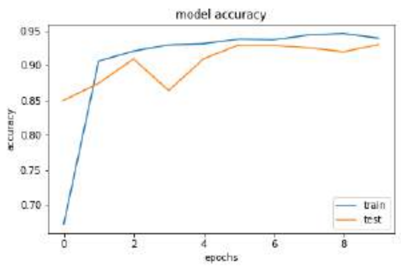
Fig. 6: The model loss vs epoch.



(a) designed model



(b) vgg model



(c) vgg pre-trained model

Fig. 7: The model accuracy vs epoch.