Multi-Digit Classification and Detection using CNN's

Anupriya
anupriya@umail.ucsb.edu

Spring 2018.

Presentation Outline

- Introduction to the Problem
- Problem Analyses and Approaches
- Approach -1
- Approach -2
- Conclusions and Improvements

Introduction to the Problem

- The goal of the project is to both recognize (classify) and detect multiple digit sequences in real world images
- Different approaches were used to tackle this problem based on Convolutional Neural Network (CNN) to achieve results which were noise, localization and scale invariant.
- Image samples from SVHN (http://ufldl.stanford.edu/housenumbers/) dataset were used to train three different CNN models: self-designed architecture, VGG-16, and Pre-Trained VGG-16
- CNN models (trained on the SVHN dataset) were developed using Tensorflow with Keras backend.

Approaches and Challenges

- **It is important to understand that detection and classification are two separate problems and need different approaches to tackle.**

- **When performing basic image classification, we pass a given input through our neural network, and obtain a single class label with a probability associated with it.**

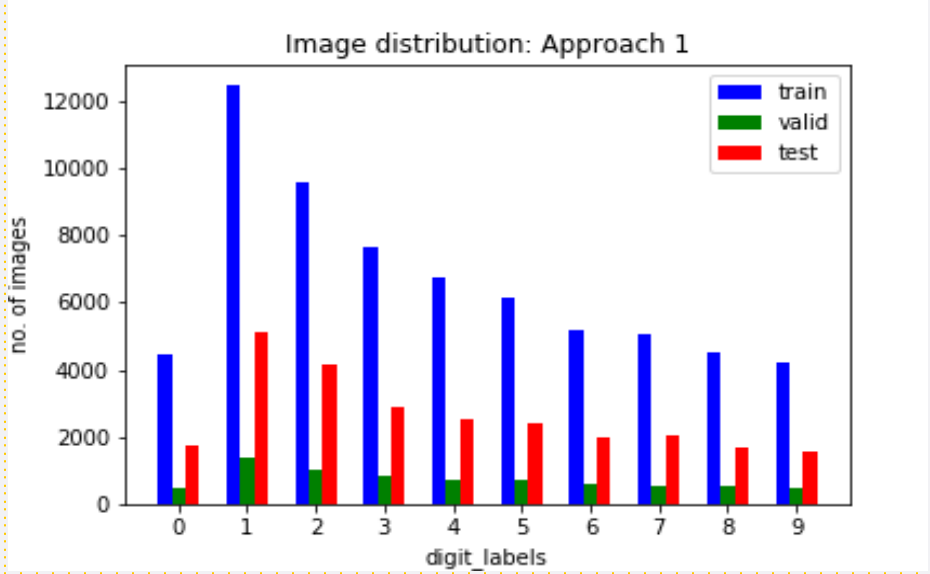
- **Object detection uses either a classifier (Deep learning or Machine learning) or conventional computer vision techniques to localize and identify the exact location of a target object.**

- **The tasks involved in this project include.**
  1. *Preprocessing –Using both* single well cropped and full house numbers
  2. *Classification* - Train three different CNN models: self-designed architecture, VGG-16, and Pre-Trained VGG-16 on the SVHN single cropped house numbers and additional cropped images from the full image dataset
  3. *Detection* – Used the CNN classifier in conjunction with Fixed-width sliding window, image pyramids and non-maximum suppression techniques

Approach 1- Single Digit Classifier

- **Pre-Processing- Only single digit cropped images**

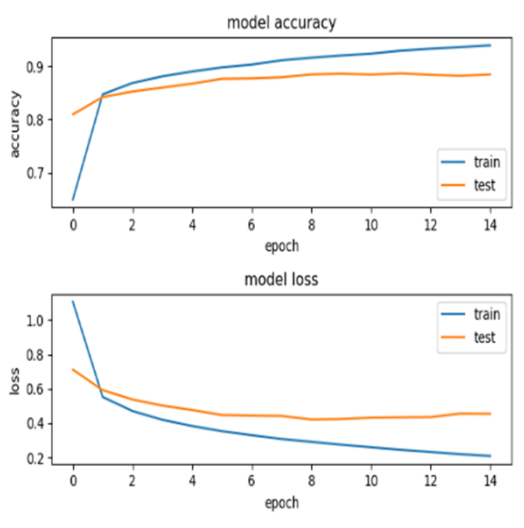
Single-Digit Cropped Dataset



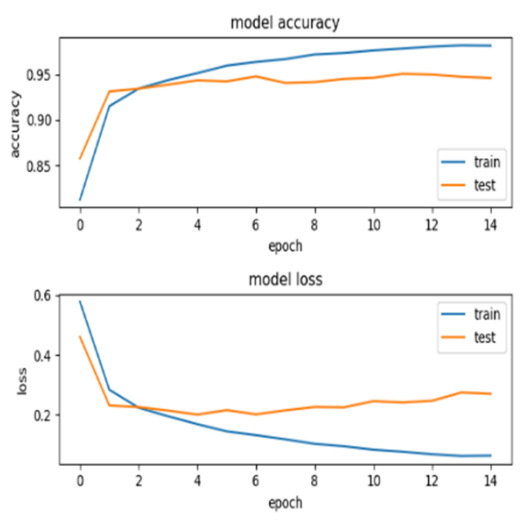
Histogram of Data Points and Labels

Approach 1- Single Digit Classifier

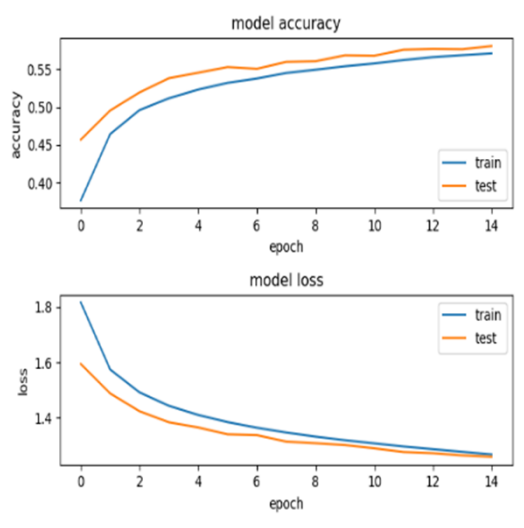
- **Classification- Testing Accuracy close to 90%**



Own Architecture



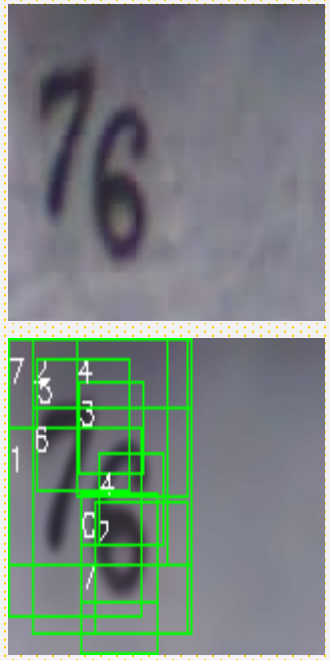
VGG 16



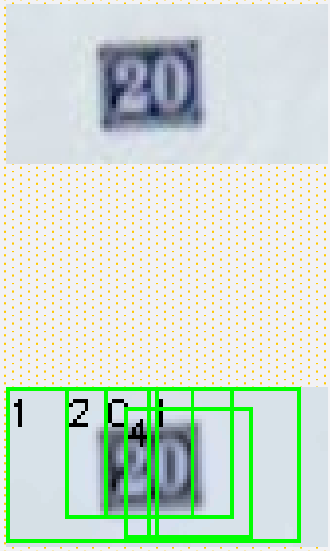
VGG 16-Pretrained

Approach 1- Single Digit Classifier

- Detection-Results – Concerns with erroneous bounding boxes and false positives.



Approach 1-False Positives

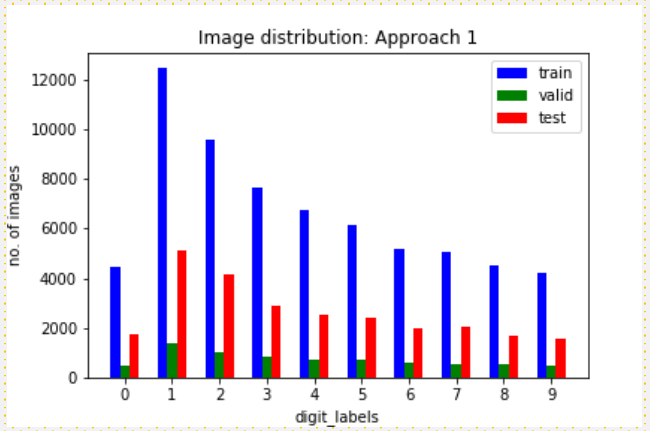


Approach 1-False Positives

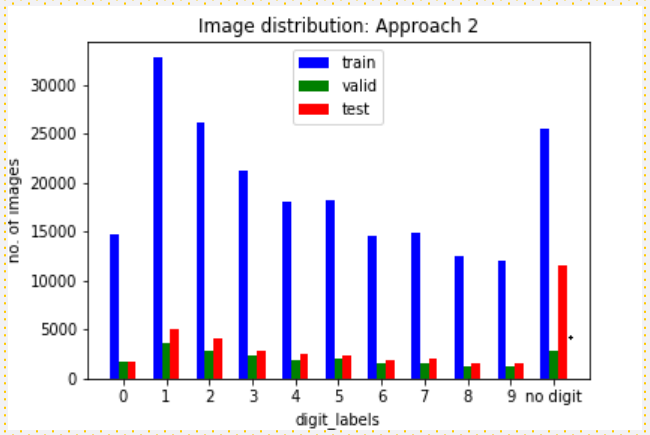
Approach 2- Single Digit Classifier

- **Pre-Processing- Single digit cropped images + Additional Training Data from Extra folder (25%) + Artificially created images gathered by cropping bounding box corners from full size images.**

Approach 2-False Positives



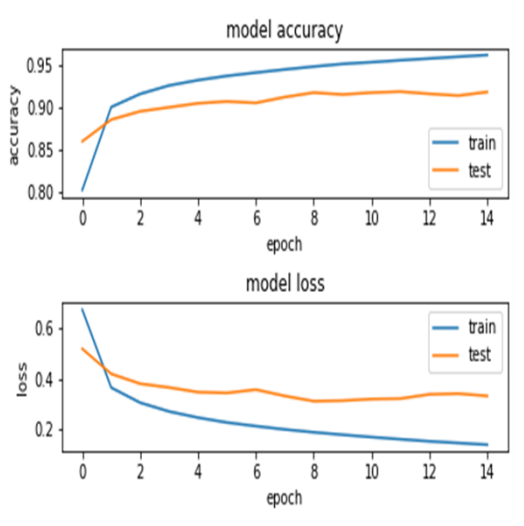
Approach 1-Data Histogram



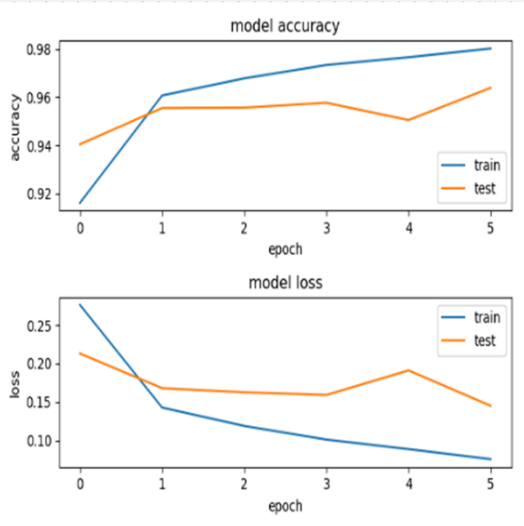
Approach 2-Data Histogram

Approach 2- Single Digit Classifier

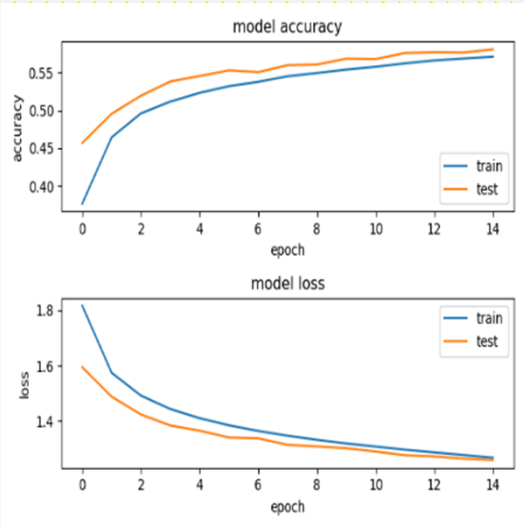
- **Classification - Testing accuracy close to 95%.**



Own Architecture



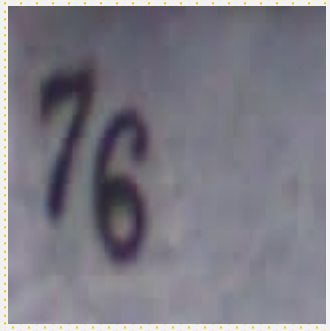
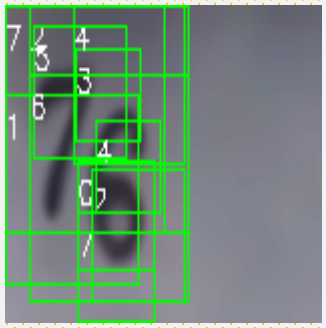
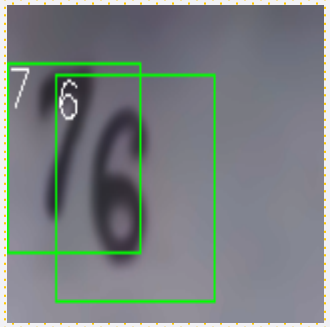
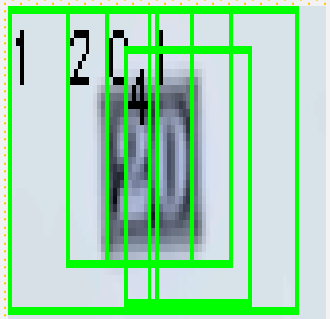
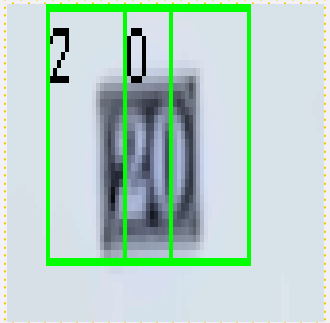
VGG16



VGG16-Pretrained

Approach 2- Single Digit Classifier

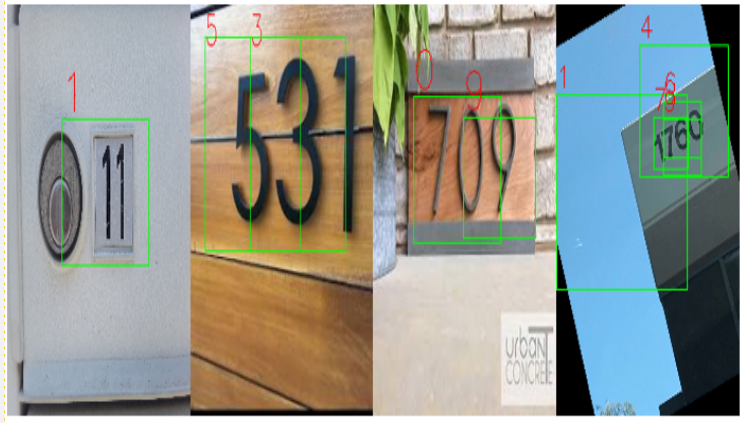
- **Detection-Results**

Original	Approach 1 Results	Approach 2-Results
		
		

Approach 2- More Results

- Detection-Results using own model on Real Images

Positive Results



Negative Results

Improvements

- **Learnings from Single Digit Classifier (Approach 2)**

1. Concerns with noise, lighting and different orientations.
2. False positives due to lack of contextual information. Creating a more robust dataset is very important.
3. Issues with Repeated digits of same digits causes some errors in the my implementation of non-maximum suppression.
4. Detections algorithms using sliding windows are very time consuming. Conventional computer vision techniques have to be applied to get a region of interest before classification can be applied.

- **Possible Solutions and Future Improvements**

1. An option of multi-digit classifier trained on full house images was explored based on work of Goodfellow.
2. Detections process was made faster by using canny filters, thresholding and resizing.

Approach 3- Multi Digit Classifier

Single Digit Classifier

The SVHN cropped dataset is used in addition with cropped images from the full set.

Data is very similar to MNIST data with 32X32 images

Labels are straight forward:

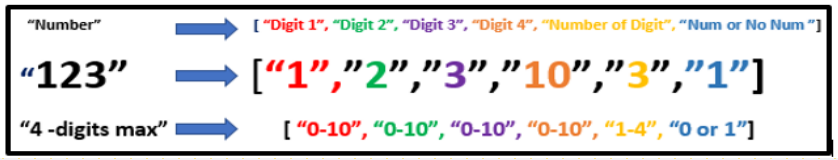
10 classes, 1 for each digit. Digit '1' has label 1, '9' has label 9 and '0' has label 10.

Multi Digit Classifier

The SVHN full dataset is comprised of colored images was used.

Each of these sub datasets have an accompanying reference file (digitStruct.mat) to help in image extraction. The extracted image being variable in size is resized to a standard 64 pixels by 64 pixels to create our final training, validation and testing dataset.

The labels required considerable manipulation and Goodfellow's paper was used as a guide to build the model



Approach 3- Multi Digit Classifier

Single Digit Classifier

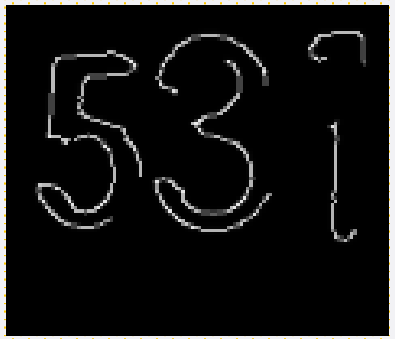
conv2d (Conv2D)	(None, 30, 30, 64)	1792	
max_pooling2d (MaxPooling2D)	(None, 15, 15, 64)	0	
conv2d_1 (Conv2D)	(None, 13, 13, 64)	36928	
max_pooling2d_1 (MaxPooling2	(None, 6, 6, 64)	0	
flatten (Flatten)	(None, 2304)	0	
dense (Dense)	(None, 128)	295040	
dense_1 (Dense)	(None, 10)	1290	

Multi Digit Classifier

dropout_4 (Dropout)	(None, 4, 4, 256)	0	max_pooling2d_4[0][0]
conv2d_5 (Conv2D)	(None, 4, 4, 256)	590080	dropout_4[0][0]
max_pooling2d_5 (MaxPooling2D)	(None, 2, 2, 256)	0	conv2d_5[0][0]
dropout_5 (Dropout)	(None, 2, 2, 256)	0	max_pooling2d_5[0][0]
flatten_1 (Flatten)	(None, 1024)	0	dropout_5[0][0]
dense_1 (Dense)	(None, 128)	131200	flatten_1[0][0]
dropout_6 (Dropout)	(None, 128)	0	dense_1[0][0]
dense_2 (Dense)	(None, 11)	1419	dropout_6[0][0]
dense_3 (Dense)	(None, 11)	1419	dropout_6[0][0]
dense_4 (Dense)	(None, 11)	1419	dropout_6[0][0]
dense_5 (Dense)	(None, 11)	1419	dropout_6[0][0]
dense_6 (Dense)	(None, 5)	645	dropout_6[0][0]
dense_7 (Dense)	(None, 2)	258	dropout_6[0][0]

Image Preprocessing to speed up Detection

Applying a combination of Blurring, Canny filters, masking and thresholding to extract relevant image areas to avoid extensive use of sliding windows and increase computation



Thank You.

Anupriya