CAP5415-19Fall 0001

Project Report

Project Title:

Numerical Digit Detection and Classification on SVHN dataset

Course Instructor: Abhijit Mahalanobis

Prepared by:

Azwad Tamir - UCF-ID: 5063443

Nazmul Karim - UCF-ID: 4425821

Abstract

A numerical digit detection system has been build based on deep convolutional neural networks. The model is trained and tested on the SVHN dataset which consists of bulk multi-digit images of house numbers. The dataset contains two types of images. The type which consists of raw uncropped house number images has been chosen. The model consists of two parts; a detector and a classifier. The raw images are fed to the detector which creates bounding boxes around each of the separate digits of an image and crops the individual images of the digits. Next, the individual digit images are fed to the classifier, which classifies the images into 10 classes starting from '0' to '9'. The detector is based on resnet50 [1] and Yolo-v2 [2-3]. It is built from scratch using the PyTorch machine learning framework. The individual accuracy of the detector and the classifier has been evaluated. The detector reports a training accuracy of 91% and a test accuracy of 59% while the classifier reports a training accuracy of 94% and a test accuracy of 92.11%. The overall training and testing accuracy of the entire system is found to be 86% and 54.41% respectively.

Introduction

The ability to recognize multi-digit numbers from images has an important significance in many applications including digital map making and address localizations. There has been a lot of work on number detection from images in recent times especially on datasets like MNIST[4] and SVHN[5].

In this project, we have built a separate detector-classifier system based on Deep Convolutional Neural Networks (CNN) to recognize multi-digit numbers from images in the SVHN dataset. The PyTorch framework has been used to build the classifier network, while Tensorflow and Keras has been used to build the detector part. Here, the detector is trained to output bounding box coordinates of the different digits to separate them which is next fed to the classifier to recognize the individual digits. After training, the entire network was tested on the test dataset of SVHN to evaluate it. Also, to investigate the individual accuracies of the detectors and the classifier, the pre-cropped images from the SVHN dataset's type 2 examples were ran on the classifier and the results were recorded.

Technical description

The project architecture could be divided into two major parts. The first part is the detector, which puts bounding boxes on the individual digits in each of the image. It then crops out the bounding box region from each of the images and makes a dataset consisting of individual digit images. These cropped out images are then fed to the classifier which is trained to recognize the

label of the digit in each of the images. The detail descriptions of the detector and the classifier is given below:

Detector:

The detection network is based on the Resnet-50 and yolo-v2 with minor changes and fine tuning. The exact architecture of the network with each layer described in detail is given in the text file named '*detector_architecture.txt'*. It consists of 176 individual layers consisting of convolutional layers, activation layers, batch-normalization layers and skip connection layers. It has 23,668,267 trainable parameters and 53,120 non-trainable parameters.

The uncropped format-1 files are first downloaded from the official SVHN dataset website. Next, the images are imported into python using the Pillow library. The dataset also contains a .mat file named '*digitStruct.mat*' which contains the bounding box information for each of the digits of the images and the labels of the house numbers. The bounding box information is then extracted from the .mat file using the h5py library and saved in a numpy array. A mean square error loss function is used for detecting the bounding boxes in the yolo network. Also, the yolo network uses a cross-entropy loss function to recognize the digit labels but these predictions were ignored as a separate classifier is build for recognizing the digits.

Next, the network is trained using the SVHN training examples of 33404 images and the weights are saved. Also, the image bounding box coordinates are used to crop the images into individual digit images and saved in a folder. The cropped images are then fed into the classifier for individual image recognition. Lastly, the test images were also loaded into the detector and cropped using the learned weights. The cropped test images were saved in a different directory.

Classifier:

The classifier network is much smaller in comparison to the detector network. The detailed architecture of the classifier network is given in the text file named 'classifier_architecture.txt'.

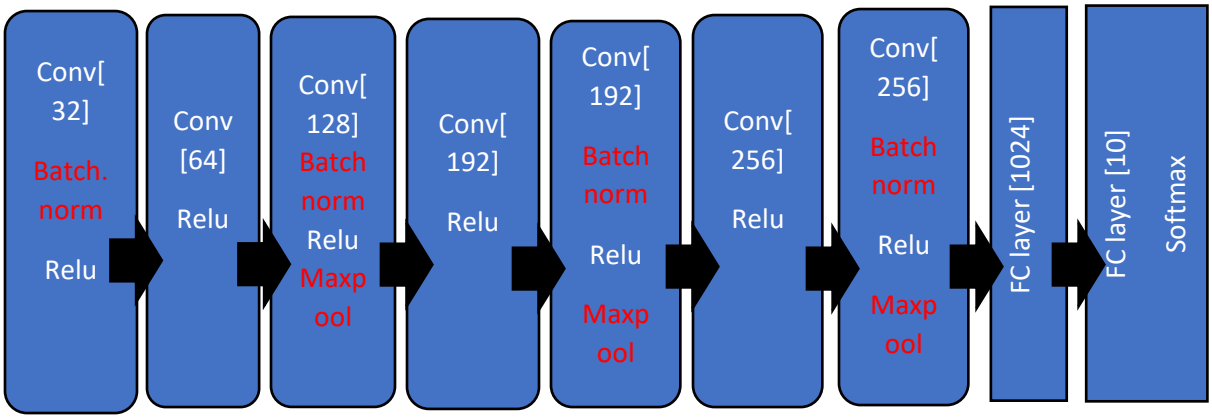

Fig. 1: The architecture of the classifier network. The no. of layers is given in third brackets

The layer wise summary of the architecture is given in fig. 1. It consists of 25 individual layers consisting of convolutional layers, activation layers and maxpooling layers. The Relu function is used as the activation function. A cross entropy loss function is used along with image normalization and softmax layer at the end and an Adam optimizer.

The process begins by loading up the cropped images from the detector using the pillow library. Next, the images are converted to numpy array and fed into a custom build torch dataset creator function. The dataset contains the numpy array cropped images and the true labels which are extracted from the digitStruct.mat file which was provided with the dataset. The torch dataset is then fed into a torch dataloader and used to train the classifier neural network. A barch size of 100 and learning rate of 0.001 was found to be the optimum value of the hyper parameters and the training ran for 30 epochs.

Lastly, the cropped test images were loaded into the classifier network and used to evaluate the performance of the entire system. Also, to find out the individual accuracies of the detector and the classifier, the classifier was also separately trained with the type-2 cropped examples of the SVHN dataset

Data set

The Street View House Number(SVHN) dataset was used for the training and testing of the networks. The SVHN dataset consists of two types of images. The type-1 images contains uncropped street number images of house numbers consisting of multiple digits. While, the type-2 consists of cropped images of individual digits. The type-1 images were mainly used on the project. Although, the type-2 cropped images were also used to train the classifier and test its individual accuracy on a separate experiment.

Fig 2. Examples of type-1 images

Fig 3. Examples of type-2 images

The type-1 images are of variable resolutions and sizes which makes it extra hard to train. The coordinates of the bounding boxes of each digits and the digit labels were both given in a .mat file named "*digitStruct.mat*" while the images were given in .png format. The type-2 images were all resized into 32X32 pixels and that along with its labels were combined into a single .mat file.

Results

The detector is trained with the 33,402 training images and the weights are saved in a .h5 file. Next, the trained detector is used to extract the cropped training images as well as the 13,068 test images. The detector was successful to detect 67689 training digit image files out of 73257 digit image files and was successful on 20,267 out of 26031 test images. This translates to an approximate efficiency of 77.86%. The entire image is resized and fed into the classifier for the example for which the detector failed to detect the bounding box. This is done so that the test accuracy of the classifier can be considered as the overall accuracy of the entire system.

Next, the cropped training images were used to train the classifier network for 30 epochs, with a batch size of 100 and a learning rate of 0.01. A training accuracy of 86 percent was attained. The network was then applied on the cropped test image developed by the detector. The test accuracy comes out to be 54.41%. This is the overall accuracy of the entire system.

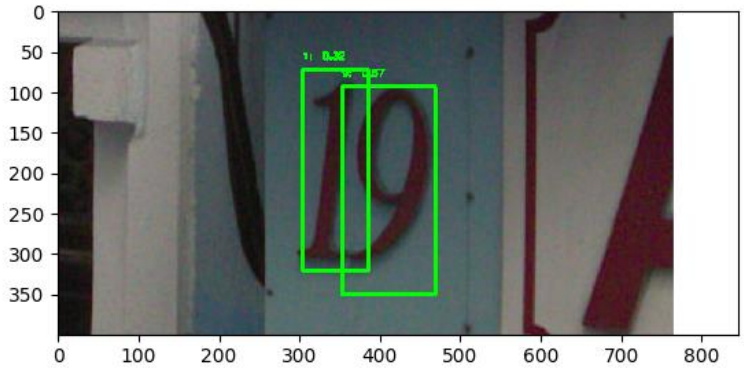

Fig.4. The output of the detector on a random example

Next, in order to estimate the accuracy of the individual systems, the pre-cropped images from the SVHN dataset are used to train and test the classifier network alone. This resulted in a training accuracy of 94% and a test accuracy of 92.11%. This is considered to be the individual accuracy of the classifier. This accuracy along with the overall accuracy of the system can now be used to estimate the individual accuracy of the detector. This gives out a detector efficiency of 91% and 59.07% on the training and test dataset respectively. Fig.4 shows the image of an example with the bounding box predicted by the detector superimposed on the image.

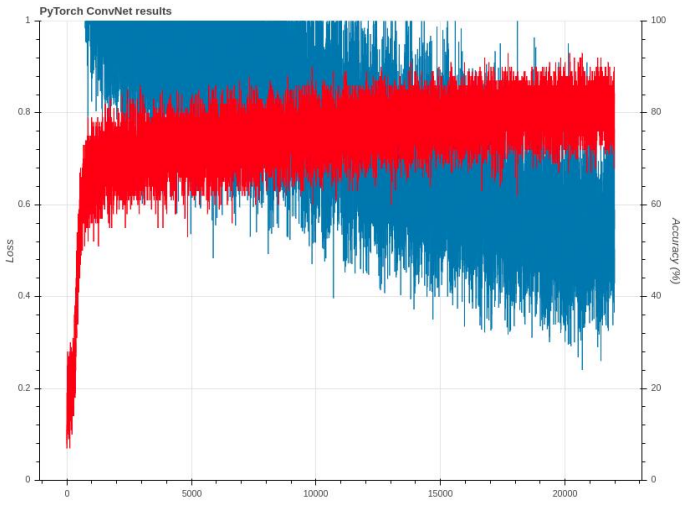

Fig. 5. Accuracy/loss vs iteration graph of the classifier training

	0	1	2	3	4	5	6	7	8	9
0	380	189	86	93	43	42	54	52	55	49
1	407	2789	502	385	321	270	253	288	200	219
2	217	394	2585	184	149	135	134	164	91	105
3	168	353	221	1569	167	140	133	113	86	95
4	163	396	229	182	1491	132	134	91	88	90
5	96	195	119	106	64	1439	68	66	50	49
6	94	245	118	81	84	80	1037	57	77	60
7	97	262	149	91	93	71	81	1116	54	53
8	59	134	67	82	57	41	43	27	926	41
9	63	142	73	109	54	34	40	45	33	834

Fig. 6. Confusion matrix of the test images for the entire system

The accuracy/loss vs iteration graph of the classifier training is given in Fig. 5. The confusion matrix of the classifier system is given in Fig. 6. This is a good representation of the precision and recall of the overall network.

The following files contains the majority of the custom build functions:

1. detector: This feeds the uncropped images into the detector and predicts the bounding box coordinates. This is then used to crop test and train images and the cropped images are saved in a folder for the use of the classifier.
2. classifier: This file contains the classifier which was built from scratch using the PyTorch machine learning framework.
3. convo.py: This function outputs convolution results given the kernel and the images.

The **main.py** is the executable file. The instructions for running the file is given in the text file named **readme_main.txt**. This file when ran takes a random test image, displays the output of the detector and also prints out the final digit prediction values of the classifier.

Conclusion

Our separate detector-classifier gets an overall test accuracy of 54.41% on the train dataset. In many instances, the detector bounding boxes of adjacent digits in an image overlap. This makes it harder for the classifier to identify the digit. This problem could be alleviated by combining the detector and classifier into one network where the inputs are the uncropped images and the output is the house number consisting of all the digits.

Furthermore, increasing the number of layers or training for more epochs may increase the accuracy of both the classifier and the detector. Another improvement that could increase the efficiency is cropping out irregular shapes from the images instead of fixed rectangular bounding boxes, but this would require a bigger network and more time to train effectively.

Transfer learning maybe another domain which could increase accuracy. The weights of the detector could be pretrained on much bigger datasets compared to SVHN and then the network could be further retrained on the SVHN dataset to fine tune it. This could increase the efficiency of the detector part of the network.

Reference:

[1] Kaiming He et al., "Deep Residual Learning for Image Recognition", Dec. 2015. Microsoft Research. Link:https://arxiv.org/abs/1512.03385

[2] J Redmon, A Farhadi, "YOLO9000: Better, Faster, Stronger", Proceedings of the IEEE conference on computer vision and pattern recognition, 2017.

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

[4] Yann LeCun, Corinna Cortes, Christopher J.C. Burges, "The MNIST database" Link: http://yann.lecun.com/exdb/mnist/

[5] The Street View House Number (SVHN) dataset, Link: http://ufldl.stanford.edu/housenumbers/