Multi-Digit Classifier

Alejandro Casanas
acasanas6@gatech.edu

Abstract. The goal of this project is to implement a system able to recognize and detect a sequence of digits in an image or video. The convolutional neural networks was trained and tested on the The Street View House Numbers (SVHN) dataset. I approach this project by implementing two Convolutional Neural Network, one which classify digits or no digits, and another to classify what sequence of digits it is.

1 Introduction

Recognizing and localizing a sequence of digits is an important problem we must solve. In this project, I explain my attempt to recognize and detect multi-digits numbers using the Google Street View imagery dataset.

There are many difficulties with multi-digits recognition because of the wide variety of fonts, scale, orientations, noise, and lighting, many of which I address in this paper. A few recent research papers have revolutionize object detection in recent years. One of them is You Only Look Once, or YOLO, which allows real time object detection and prediction(2). The YOLO algorithm places a grid, usually very small size, on top of your image and then applies a deep convolutional network to each of the grids and outputs a precise bounding box for each of the objects in the image. YOLO also uses non-maximum suppression to eliminate multiple bounding boxes for the same object. The other very common paper used for object detection is Faster R-CNN (R is for region) which finds a predefined number of clusters(regions) which may contain objects(3). Faster R-CNN uses anchors which are fixed bounding boxes thoughed the image with different sizes and ratios. Unlike YOLO which uses a grid. Then, using the object proposal windows a deep network classifies the content in the bounding boxes or discarded as background. Faster R-CNN has proven to be more accurate than YOLO architecture. However, the YOLO architecture can run in real time.

In this paper, I propose a two step approach, the first step is to use a sliding window with a binary convolutional neural network(CNN) to detect if the window has digits or not. This implementation determines the location of the sequence of digits. The second step is to use a multi-digit deep convolutional neural network to classify the sequence of digits in the cropped window. The multi-digit CNN is a very similar approach to Goodfellow paper (1).

2 Research method

A Convolutional Neural Network is a deep artificial neural network commonly used to find patterns in an image by convolving over regions of the image. The CNN is composed of hidden layers, which include convolution layer, pooling layer, fully connected layer, and normalization layer. The convolutional layer is in responsible for finding meaningful features in an image by applying different filters. Convolution layers also have an activation

function(most common is ReLU) to map the input into a range so that the network is able to learn. The pooling layer is responsible for reducing spatial information by shrinking the image into subregions and for each subregion it outputs a max or average value. A fully connected layer connect all the activations from one layer to the other. A normalization layer normalizes or maps unit values to a range so that the CNN learns faster.

The pipeline for multi-digit classifier detection system is as follows:

2.1 Input

The input is simply any image added to the system directory. This image is then resized and preprocessed before its sent to the system.

2.2 Preprocessing

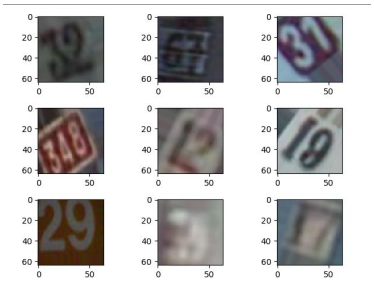

Figure 1. Data augmentation

In order for the convolutional neural network to learn from the data I used preprocessing techniques like subtracting by the mean and dividing by the standard deviation. I also tested rescaling every pixel in the range of 0 to 1.

In order to handle image invariance(orientation, location, scale, brightness, noise, etc) I used data augmentation techniques such as: random rotation, width and height shifts, brightness shift, and random zoom range. You can observe an example of the data augmentation method output on figure 1. I created a directory of this generated images and added the them to the training dataset for my system to be more robust and able to generalize better.

2.3 Sliding Window

In my approach I decided to use a sliding window because of the simplicity of the algorithm. However, in practice, sliding windows are very discourage. Sliding windows have a lot of disadvantages when it comes to object detection. A major disadvantage is that sliding windows are very computationally expensive. In order to migitage with this disadvantage, I run my binary classifier on all the windows at once, eliminating the overhead of running the CNN over each and every window. Another negative effect of using sliding windows is that the object you are looking for in the image might not be a square region, resulting in poor bounding box.

2.4 Binary Deep CNN Classifier

The binary classifier is used to detect if a cropped region of the image contains a digits. I trained this classifier with positive examples from the SVHN training dataset and negative examples from random crops outside the bounding boxes of the digits. I used this binary classifier to classify every window from the image at once and returns the regions with a high probability of digits being preset. After it finds all the possible regions above a certain threshold, the classifier sends the indexes along with the windows to the Multi-Digit CNN classifier. This model archived a 98% accuracy on detecting if digit is present or not.

2.5 Multi-digit Deep CNN Classifier

The Multi-digit CNN classifier is responsible for classifying the images sent from the binary CNN and picking the best window as the solution. The multi digit classifier model is set up to output a 5x11 matrix corresponding to the maximum length the classifier(max length is 5) and 11 one hot encoding for each digit between 0 to 9, and 10 for background. From the instruction, I use Stochastic Gradient Descent (SGD) on all my architecture to optimize the weights parameters and minimize the loss function. In order to set the learning rate hyperparameter I tested multiple values starting from the two extreme and then working my way through a value which I thought was a good pace for the CNN to learn. It was neither too fast nor too slow, and the CNN seem to progress until about the 25 epoch, which I decided to stop because the model's loss was starting to increase.

I implemented the following three architectures for the multi-digit classifier:

2.4.1 Custom Architecture

My custom architecture is based on the AlexNet and VGG16 architectures. I have deep neural network with 6 convolution hidden layers stack on top of eachother with increasing number of nodes. I used a 3x3 filter and a stride of 1. I also use a pool size of 2x2 and a ReLU activation function. I also added 5 dropout layers which deactivate ¼ of the neurons and improves generalization by forcing the layers to learn with different neurons. All layer connections are feedforward.

2..4.2 VGG 16 trained by me and a VGG16 pretrained on Imagenet

I reimplemented all the layers in the VGG16 architecture to get a better idea of the design and to have full control of the network. Since I coded the entire network, it was really simple to change the top layers to match the input of my system(64x64x3) instead of the default values(244x244x3). I also modified the output layers and added 5 Dense layers with a softmax activation to output the probabilities of each digit in the sequence. There was not much different in terms of scores by the two implementations. The VGG16 I implemented was only about 1% lower than the pretrained 'imagenet' classifier. For training the pre trained VGG16 classifier I used transfer learning with ImageNet weights which has already being trained on 1.2 million images and 1000 categories, compared to our 33,000 images on training set.

2.5 Pipeline flow

The multi digit localization process is done to locate the bounding box of the sequence of digits in an image. This bounding box detection is achieved using a sliding window method and a deep binary CNN. The sliding window algorithm starts from the top left corner of the image with a 28x28 window size and a stride of 6 and it will move to the end of the image until it reaches the bottom right corner. Once it has gathers all the windows at different scales(window scales: [(28, 28), (48, 48), (64, 64), (128, 128), (256, 256)]), the binary CNN categorizes the window as 1 if has digits, or 0 if it does not. If the deep neural network classifies the window as a potential sequence of digits, the window is sent to the multi-digit deep CNN which classifies the sequence of digits (up to 5 digits ranging from 1 to 99999) and keeping a maximum probability score of each digit.

After all the window sizes are done, the algorithm will pick the sequence of digits with the maximum sum of probability score. For example, if the algorithm saw a 5 on the first window, and a 15 on the second window it will pick 15 because it has a combined higher probability than 5 alone. I tried using non-maxima suppression which also picks the bounding box with the highest probability, and then looks at the overlapping bounding box with high intersection of union. However, sometimes it will pick multiple bounding boxes for the same sequence of digits even after processing. My results did not improve with non-maximum-suppression, so I decided to just keep the algorithm as simple as picking the highest probability.

Note, I also tried using image pyramid approach and constant window size. However, the bounding boxes seemed better with increasing window method.

3 Experiments

3.1 SVHN Data Set

The following results are example of where the multi-digit algorithm performed well and where it failed to detect the right sequence of digits. As you can see in the failed classifications table, the algorithm had trouble with some of the numbers that are not very clear in the image. I myself, might have some trouble determining the numbers. It also struggled a with sequence of numbers stack on top of each other since the training dataset did not contain many of these sequences the algorithm was unable to generalize well for this functionality.

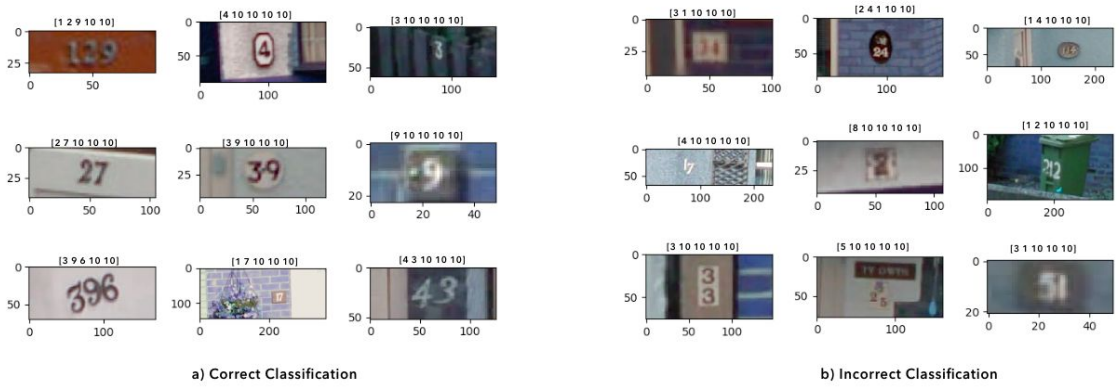

Figure 1. Performance on SVHN Test Data Set

3.2 Results

The best results archived by the these three architectures are compared in table 1. The current state of the art algorithm has a 1.02% test error according to the article by Cubuk and Zoph(5).

Architecture	Accuracy	Loss
Custom CNN	88%	0.37
VGG16	96%	0.08
VGG16 Pretrained	97%	0.08

Table 1. Performance on SVHN Test Data Set

Figure 3 illustrates the models accuracy as a function of epoch . As you can see I decided to stop my algorithm at around epoch 25 because loss started to increase.

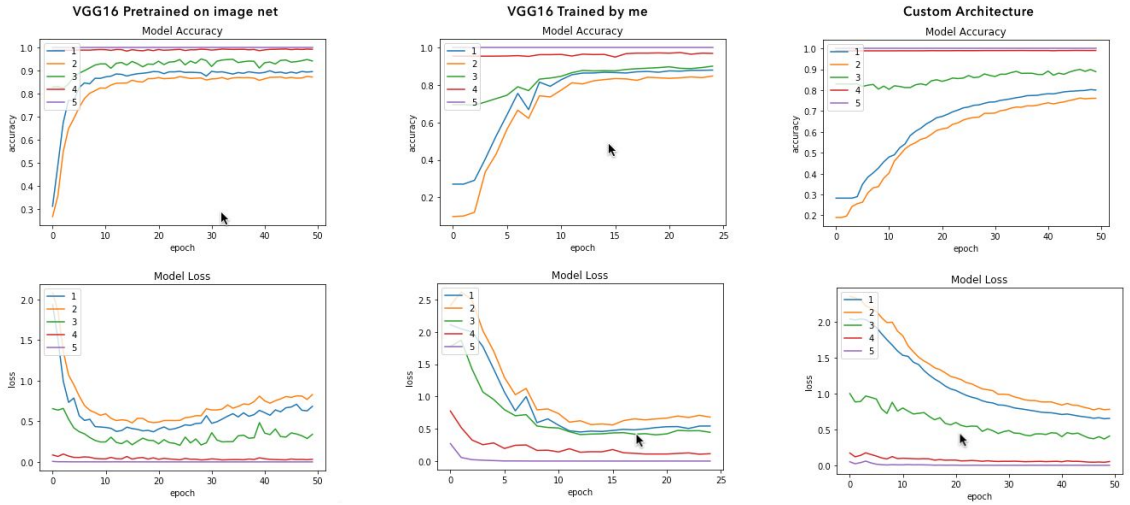

Figure 3. Accuracy and Loss graphs

4 Discussions

I believe the model proposed in this papers solve the problem of finding small sequence of digits in an image. I understand the model is slow and unlike modern architectures such as YOLO that perform object detection in near real time. Another problem with this implementation is that the sequence is bounded my a maximum length of five digits in this architecture. However, a larger length could be addressed by having a larger training set and minor modifications to the CNN model.

5 Links

Video link: https://youtu.be/4h1WYyUv6G8

Presentation: https://docs.google.com/presentation/d/1c-aIlABcAMnipdipvN-goG1h9t8S89LJG1nDYLtvExU/edit?usp=sharing

Weights: https://drive.google.com/open?id=1l8njJeJwaMu3cOW9cUK_sslraNrPdECj

6 References

1. Goodfellow, Ian & Bulatov, Yaroslav & Ibarz, Julian & Arnoud, Sacha & Shet, Vinay. (2013). Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks.
2. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: unified, real-time object detection. In: CVPR (2016)
3. S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In NIPS, 2015
4. The Street View House Numbers (SVHN) Dataset: http://ufldl.stanford.edu/housenumbers/
5. E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, "Autoaugment: Learning augmentation policies from data," arXiv preprint arXiv:1805.09501, 2018.
6. Simonyan, K. & Zisserman, A. Very deep convolutional networks for large-scale image recognition. In Proc. International Conference on Learning Representations http://arxiv.org/abs/1409.1556 (2014)