Multi Digit Number Recognition on Google Street View House Number Dataset using CNNs

Sethurathienam Iyer
2013B4A7720G
BITS-Pilani, Goa Campus

Arpit Pandey
2013A7PS075G
BITS-Pilani, Goa Campus

***Abstract*—Single digit recognition is pretty much a solved task in Computer Vision with state of art deep learning architectures achieving accuracy as high as 99% in the MNIST dataset. Multi-digit recognition in natural images, however is a challenging problem due to variability in appearance, the need of digit segmentation. This project aims to predict multiple digits from Google's The Street View House Numbers (SVHN) dataset using Convolutional Neural Networks. We also see possibility of swapping out the topology of ConvNet, except the readout layers to other proven working ConvNets such as GoogLeNet, VGGNet.**

***Index Terms*—CNN, SVHN**

I. Introduction

This project explores how convolution neural networks can be used to effectively identify a series of digits from real-world images that are obtained from SVHN dataset.

Recognizing the street numbers from natural scene images has many applications but it is quite challenging because of enormous appearance variations in natural images, e.g. different fonts, scales, rotations and illumination conditions.

We start off with a simple ConvNet structure (LeNet-5) and we make refinements to determine the optimal model for identifying multiple digits from real world images.

II. Task Definition

The aim of the project is to recognize accurately and classify the digits from natural images. Important thing to note is that, instead of the standard identification of numbers (like in MINST), we need to detect and classify sequence of numbers. Fortunately, the length of the sequences is bounded and hence, we can use some model which fixes the length to some constant, say $N = 5$.

We can approach this problem through two different techniques. One technique is to crop out all the individual digits and make every one of them having uniform size, followed by applying convolution neural networks on isolated numbers to recognize the numbers in natural images. During the testing phase, Use object localisation methods like *Histogram of Gradients(HOG) with Non Maximum Supression (NMS)* and crop the detected digits and use ConvNet for identification.

See Fig. 1 for one such successful bounding after applying HOG and NMS. Unfortunately, this doesn't generalize well for every image in our dataset.

Second technique combines the localization and classification using RCNNs,You Look Only Once (YOLOs) etc.

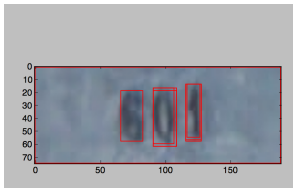

Fig. 1. Bounding box of image 21 after applying HOG and NMS

The first technique is not quite effective in SVHN dataset because of huge variation of natural images in terms of apperarence,contrast,resolution of images etc. Naturally, we approach the problem via the second technique.

III. Exploring the data

We used various python scripts to explore the data and found various interesting properties which are presented below.

A. Clarity of the Data

There are 10 classes in the data, 1 for each digit and 10 is denoted by '0'.When we went into the raw data to explore the images, we noticed few images had undesirable properties, like Image 141 was barely perceptible even for a person (Refer Fig 2).Fortunately, many images in the dataset are clear (Refer Fig 3) easily predictable by ConvNets.

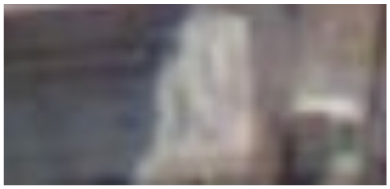

Fig. 2. Image 141 of SVHN dataset

Fig. 3. Image 180 of SVHN dataset

B. Data Preprocessing

We take the original image containing all digits and focus on recognizing them simeltaneously. First, we will find a bounding box which has individual charecter bounding boxes and then expand this image by 30% in both x and y direction, crop the image to this box and resize the crop to 64×64 pixels.

This is effectively discussed in the next section.

C. Directory Structure

To initialise the *SVHN* object, we input a dataset folder. The Directory structure of the dataset folder is as follows When initialised, we preprocess all the images in the folder

```
dataset
    - train_images
        - 1.png
        - 2.png
        - ...
    - train.csv
```

Fig. 4. Directory Structure of dataset folder

(As discussed in B).

IV. Scikit-Learn Approach

First, we approached this problem using scikit learn. We first used re-trained LeNet-5 on single digit images of SVHN dataset to generate an one dimensional feature vector and then trained 5 logistic regressors to detect the 5 digits separately. Using grid search, we selected the hyper parameters and the best accuracy we could come up was pretty low, Hence we decided to use deep learning.

V. Tensorflow approach

In this section, we will discuss how did we approach this using Tensorflow.

A. Reading data from files

A typical pipeline for reading records from files has the following stages:

1. The list of filenames
2. Optional filename shuffling
3. Optional epoch limit
4. Filename queue
5. A Reader for the file format
6. A decoder for a record read by the reader
7. Optional preprocessing
8. Example queue

Tensorflow supports data in TFRecords format and we have to convert the directory structure given in subsection C into a TFRecord. To convert it to TFRecord, we define a Reader class which has the method read and convert. The main function which does the job of converting to tfrecord is *convert_to_tfrecords* function. The whole code for this is written in *convert_to_tfrecords.py* and it is written using *tf.app*.

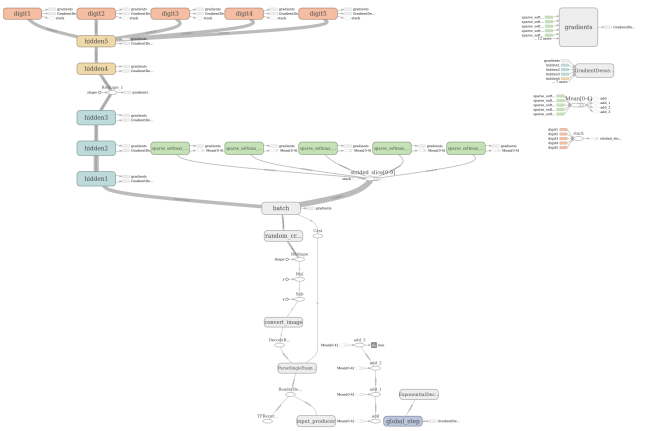

Fig. 5. Dataflow graph of the deepnet

To read from TFRecords file, we use *tf.TFRecordReader()* which gives the output images and digits in serialized way.

Next, we build a batch of images and digits using *tf.train.batch* which is written in *batch.py*. So, the data now is preprocessed and stored in tfrecord file and can be effectively extracted out and batched.

We made a 75-25 split between train and validation and hence, allocated 24944 images for training and 8457 images for validation.

B. Model Architecture

The Model architecture is given in Fig.5. This contains over 3 convolutional networks and 2 fully connected layers (dense layer) followed by 5 softmax to predict 5 digits .

We had to do lot of trail and error to come up with optimal parameters and it was possible to do so because of low number of Convolutional layers in the architecture.

C. Accuracy and Loss Graphs

Since we used less number of layers, the max validation accuracy we got was 70.4%. Fig 6 and 7 show the Tensorboard visualizations for accuracy and loss. We used Mini Batch Gradient descent with exponentially decaying learning rate.

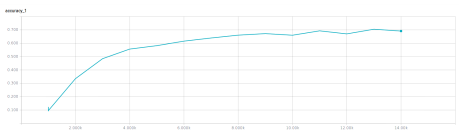

Fig. 6. Variation of Accuracy over 14000 iterations

Fig. 7. Variation of loss over 14000 iterations

The training was done for one hour on NVIDIA GeForce GTX 965m GPU. We believe accuracy could be increased by adding more hidden layers and giving more time to train.

VI. Visualization of Predictions

For visualizing the prediction for any random RGB image, we first preprocessed the image (Steps can be seen in Fig 5) and resized it to $(54, 54)$. Image should be in .png format as we are using *tf.image.decode_png* to decode the image. Fig 8 shows some of the correct predictions.

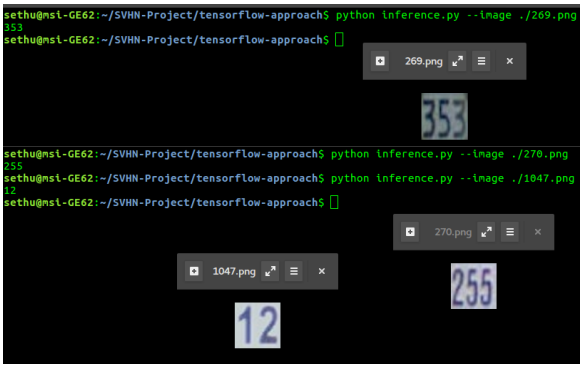

Fig. 8. Visualization of Prediction using inference.py

VII. Conclusion

In this project, we did the problem using both the approaches. In scikit-learn approach, the preprocessing of data wasn't much of a challenge but the models took lot of CPU resources and time to train over 27000 entries. In the end, accuracy was very bad (near 9%.).

The preprocessing of data in Tensorflow approach was very hard for us as we wanted to do mini batch gradient descent and we were not acquainted with the process of making TFRecord files. Majority of the time was taken by this phase of the project. After we finished this, we wanted the results as soon as possible so went with a model which has very less convolutional layers and much to our surprise, the model was good enough to produce 70% validation accuracy in just one hour of training. We conclude that for general OCR related tasks, Deep learning can give much quicker (if the dataset is huge) and much better results than traditional machine learning methods like Logistic Regression.

Acknowledgment

We would like to thank the TAs for giving this project as final project and clearing doubts on the way. We would like to specially thank Andrej Karpathy's blog [1] for clearing most of the mystery about Convolutional Neural Networks.

References

[1] A.Karpathy, http://cs231n.github.io/convolutional-networks.
[2] Danill's blog, http://warmspringwinds.github.io/tensorflow/tf-slim/2016/12/21/tfrecords-guide/
[3] Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks (https://arxiv.org/abs/1312.6082)
[4] C. Olah,http://colah.github.io/posts/2014-07-Conv-Nets-Modular/
[5] Tensorflow Documentation, https://www.tensorflow.org/programmers_guide/reading_data from-files