Multi-digit Number Recognition from Street View Imagery

Aditya Daflapurkar
B.E Computer Science
BITS Pilani
Goa,India
f2013091@goa.bits-pilani.ac.in

Aditya Singh
B.E Computer Science
BITS Pilani
Goa,India
f2013098@goa.bits-pilani.ac.in

***Abstract*—In this project, we intend to identify house numbers from Google street view images. We aim to solve this problem using convolutional neural network, in order to testify how well deep learning can automatize the process of background extraction, topology visualization, and photometric normalization. We also intend to compare the performance between neural networks of different architectures, and justify the reason behind this.**

I. Introduction

Extensive research has been dedicated to the effort of street view digit recognition. This is a general problem that falls under the category of character classification with natural scene background. It is a problem that plays an important role in our everyday life. The ability to recognize street view digits with varied backgrounds enables machines to automatically read off numerical information such as house number in a real-time manner. Another related prevalent usage would be the improvement in visual search engines efficiencies. As opposed to the well studied traditional optical character recognition with pure bitmap inputs, the input here comprises characters with various fonts, colors and backgrounds, which adds an entire scale of difficulty. The former was a simple problem, hence even classical machine learning algorithms such as multiclass SVM was sufficient to solve the problem satisfactorily (albeit not perfectly). To solve digit recognition in natural scene, due to the high flexibility of input arises from situations such as multiple fonts within a single class and background that obscures the foreground edges, using classical machine learning techniques would require way too much manual pre-processing work. Hence we plan to employ convolutional neural network (CNN), namely neural networks which parameters are tied across multiple neurons, testified to be especially useful in image recognition.

II. Data Preprocessing

A. Dataset

The SVHN(street view house numbers) training-dataset consists of 33402 house number images consisting of 73257 digits. Each of the images are of different dimensions and of different clarity. Along with the images a csv file is provided which contains the digits in each image with the bounding box coordinates of those digits.

B. Preprocessing

The train.csv file provided with the dataset contains one record for each digit in each image giving the bounding box coordinates of the digit. This file was processed and another csv file(svhn.csv) was created which contained 33402 records(1 record per image) and 11 columns(4 columns for bounding box coordinates of the image, 1 column for length of the number and 6 columns, one for each of the six digits in the image). The blank entries in digit columns were filled with value ‘10’ (denoting non-existence of digit) and the digits labels ‘0’ to ‘9’ represent valid digits detected in an image. The were some columns in the train.csv file in which bounding box coordinates had values equal to -1. Such negative coordinate values were replaced with 0. The images were cropped to fit in the bounding box coordinates in svhn.csv. These images were resized to 32x32 size as this was approximately the average image size in the t.

III. Input and Output for the model

The 32x32 images are given as input to the model. The length of the highest number in the training dataset is 6. But, this is number was treated as anomaly and was put into validation set. The output for the model is a list of digits which contains the digit sequence in the input image.

IV. Model Construction

We tried different CNN architectures and refined

hyperparameters and architecture to achieve higher accuracies. We will discuss the details of the architectures in this section.

A. Model Architecture 1

1. Implementation 0 (crude form)

At first, we started off with a simple model comprising the following layers. We have used five parallel softmax layers:

Layers: Input Layer (32 x 32 x 3)

Convolution 1 (Filters 16 | Receptive Field: 5 x 5 | Stride: 2 | Padding: Valid)

ReLU (Rectified Linear Unit Activation Max Pooling Receptive Field: 2 x 2 | Stride: 2 | Padding: Valid)

Convolution 2 (Filters: 32 | Receptive Field: 5 x 5 | Stride: 2 | Padding: Valid)

ReLU (Rectified Linear Unit Activation) Dropout (Keep probability of 0.9)

Fully Connected Layer (FC) (Nodes: 64)

Softmax Layer (5 softmax output layer units, each with 11 outputs)

For optimizer, we used Stochastic Gradient Descent (SGD) with a learning rate of 0.05. We split the available training dataset into training set(27835) images and test set(5567)images. We trained the model for maximum 15 epochs. With this model, we got a very low validation accuracy(2.99%) and the loss function curve was a plateau with slope becoming 0 at an average loss value of 6.

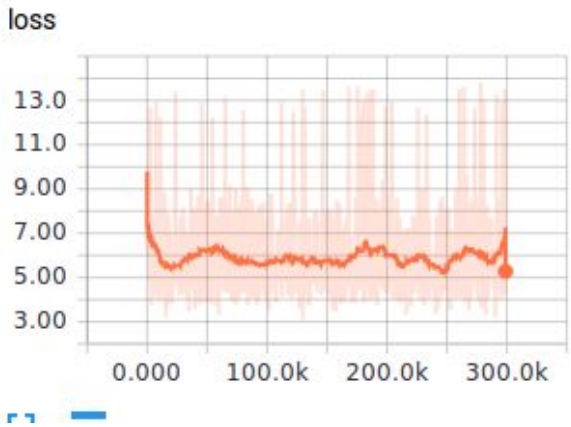

Figure 1. Loss function in Model 1 pass 1

Figure 1 shows loss function in Model 1 implementation 0.It can be seen that the value does not go below 5.0 even after convergence.

2. Implementation 1

We used Adagrad Optimizer with exponential decay with model architecture 1. Adagrad was chosen because our dataset is sparse for digits in numbers and because it adapts the learning rate to parameters performing larger updates to infrequent parameters and smaller updates to the frequent parameters. We also decreased the keep probability of dropout to 0.5 assuming that a high dropout value may have led to underfitting. Figure 2 shows the loss function in this implementation.

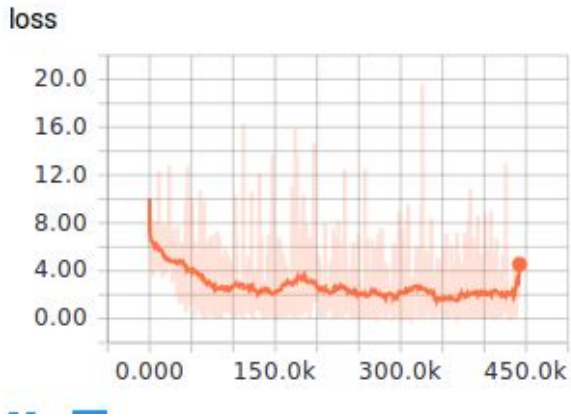

Figure 2. Loss function in Model 1 pass 2

As is seen from the figure, the loss function converged well in this pass, with average loss=2(approx.) in final epoch. The accuracy of this model on validation set is 52.97%. However this model was not very accurate while recognizing numbers with more than 2 digits.

B. Model Architecture 2

1. Training Pass 1

This model was similar to model architecture 1 except that we have added an additional convolutional layer. The dropout keep probability is set to 0.9 in order to avoid overfitting. Following are the layers in this model.

Layers: Input Layer (32 x 32 x 3)

Convolution 1 (Filters 16 | Receptive Field: 5 x 5 | Stride: 2 | Padding: Valid)

ReLU (Rectified Linear Unit Activation Max Pooling Receptive Field: 2 x 2 | Stride: 2 | Padding: Valid)

Convolution 2 (Filters: 32 | Receptive Field: 5 x 5 | Stride: 2 | Padding: Valid)

ReLU (Rectified Linear Unit Activation) Dropout (Keep probability of 0.9)

Convolution 3 (Filters: 64 | Receptive Field: 5 x 5 | Stride: 2 | Padding: Valid)

Fully Connected Layer (FC) (Nodes: 64)

Softmax Layer (5 softmax output layer units, each with 11 outputs)

Adagrad Optimizer with exponential decay was again used in this model. Following figure shows the loss function obtained for this pass.

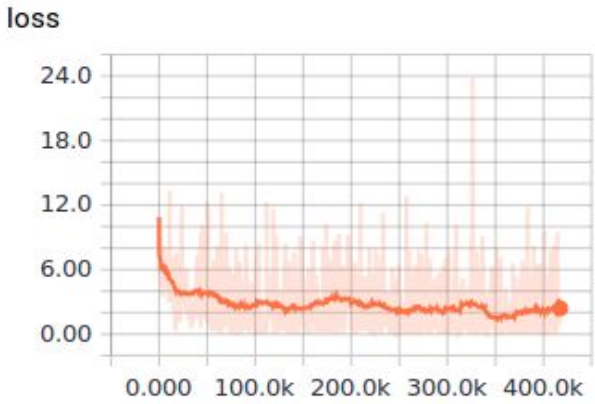

Figure 3. Loss function in Model 2 pass 1

We trained the model for 25 epochs and the figure shows that the loss function converged faster in this model. This model showed higher accuracy in predicting 3 digit numbers fro images. The validation accuracy for this model was 51.67%. This was lower than accuracy in model 1 implementation 1. We presumed it might be due to overfitting and thus we did pass 2 on this model.

2. Training Pass 2

We saw that model 1 loss function converged too quickly and it had good training accuracy but lower test accuracy. Assuming overfitting to be the reason behind this, we did a second training pass on this model with fewer number of epochs(11 epochs).

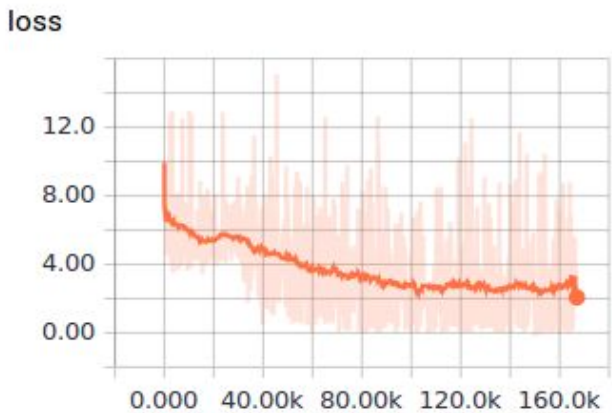

Figure 4. Loss function in Model 2 pass 2

This model was not overtrained after convergence as opposed to the model in pass 1. Thus it showed higher validation accuracy(54.29%).

V. Results and Cross-Validation

Comparing the two trials, we can see how the second model has a higher accuracy. Hence, since the second model has the best accuracy, we have decided to settle with this simple ConvNet that produced 54.29% on unseen data. We did a 6 fold cross-validation(each fold consisting 5567 images) on the available training dataset and we got the following results. Table 1 shows the cross validation results.

Validation Fold no.	Accuracy(%)
1	50.29
2	51.93
3	53.88
4	52.81
5	53.26
6	54.29

Table 1. Cross-validation results

The average cross validation accuracy was 52.74%.

VI. Area for Improvements

The accuracy of the second model can be further improved by adding more convolutional and fully connected layers to the network. Also, optimizer functions can be changed to Adam optimizer or Momentum optimizer. We did not use any renowned topology highlighted in the introduction such as VGGNet or ResNet .There is room for improvement by using existing weights trained on the ImageNet dataset using ResNet or other winning topologies and change the readout layer to the parallel readout layers. This may give a substantial boost to our test accuracy and reduce our training time.

References

[1]] Ian J. Goodfellow, Yaroslav Bulatov, Julian Ibarz, Sacha Arnoud, and Vinay Shet. Multi-digit number recognition from street view imagery using deep convolutional neural networks. 12 2013.