Google Street View Character Recognition

Jiyue Wang and Peng Hui How
Stanford University
450 Serra Mall, Stanford, CA 94305
{jiyue, phow1}@stanford.edu

Abstract

In this project, we intend to identify characters (0-9, a-z, A-Z) 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.

1. Introduction

Extensive research has been dedicated to the effort of street view character 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 character with varied backgrounds enables machines to automatically read off textual information such as road signs 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 character 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.

2. Related Work

Digit recognition was one of the earliest problem successfully tackled by CNN. Back in 1998 when LeCun et.al. first formalized the concept of CNN, the LeNet5 architecture was used to solve handwritten digit recognition (processed as bitmap) in the famous MNIST (Mixed National Institute of Standards and Technology) database, achieving an accuracy rate of 0.99.

On a similar note, Google managed to tackle a much more challenging version of this problem [3], transcribing a sequence of street view house numbers under natural scene and lightning under the SVHN (Street View House Number) dataset, with an unpreceeding overall sequence transcription accuracy of 0.96 and character level accuracy of 0.98, employing a CNN with 11 layers, consisting of eight convolutional hidden layers, one locally connected hidden layer, and two densely connected hidden layers, without any skipped connection. This architecture took approximately six days on 10 replicas in DistBelief, an infrastructure that uses 16000 CPU cores spread across 1000 machines. Repeating the same is beyond the reach for regular consumption, in particular in the absence of such gigantic computational resources.

In our work, we explore CNNs with comparable performance on a 62-class natural scene character recognition problem using Caffe framework [4]. We experimented the different effect of architectural choices including various depth, dropouts, leaky ReLu, learning rates, and regularizations. To compensate for the small streetview character dataset, we experimented on transfer learning on different trained models. However, it turns out that training from scratch on a three-layer CNN is even better than transfer learning on some four-layer model. We also did data augmentation to improve the test accuracy.

All our network trainings were done using the Caffe framework.

3. Problem Statement

Our main data source is the Chars74K dataset [1]. It has 7705 characters from natural images which are further classified into 'good images' and 'bad images' (the latter are somtimes hardly distinguishable by human eyes), 3410 hand drawn characters using a tablet PC, and 62992 synthesized characters from computer fonts.

Here are some example images in the dataset:

Figure 1. Streetview characters in chars74k

Figure 2. Synthesized fonts in chars74k

Our emphasis is on the natural images, thus we used synthesized characters and hand drawn characters as pre-train data. After examining the Google street view data, we decided to pre-train CNN on the synthesized characters and do transfer learning. We also tested transfer learning from LeNet5 (trained on MNIST). At first we believed that the topological structure was the most important factor to classify a number/character, hence we transformed the natural images into grayscale before doing transfer learning from LeNet5. Later, we tried to train CNN from scratch on raw RGB data of natural images and obtained a slightly better result. This might because some information were lost when we transformed from RGB to grayscale.

We compared our results against traditional methods such as SVM. Some archived results using traditional methods in this area include work from Microsoft research [2]. From the previous work, we know that the best accuracy is 0.55 (obtained by multiple kernel learning). We thus expect higher accuracy for CNN.

In the next section, we would discuss about our various experimental procedures, mainly in transfer learning, RGB versus greyscale, adjusting learning rate and regularization rate, changing number of outputs in various layers, choice of nonlinear function (ReLU Versus Leaky ReLU), and dropout rates (if using this option).

4. Experiment

4.1. Pre-Processing

This procedure consists of resizing followed by data augmentation. We first resized all the images to the same size, i.e. 64 by 64. Initially, we believed that all it matters is the topology of the character itself, and this is invariant to skewed scaling. Also, considering we need to do transfer learning on models pre-trained on grayscale data, we tranformed all the images from RGB to grayscale. We then splitted the data into training set (4 / 5) and validation set (1 / 5) and train different models using Caffe + GPU.

Originally, there were only 7705 'good' natural scene characters in the dataset. The limited amount of dataset will tend to overfit the trained model. Thus, we perform data augmentation by adding various forms of noise (Gaussian, Poisson, Salt and Pepper) to the original image. On top of this, we also perform randomized rotations to the noisy images, as illustrated below:

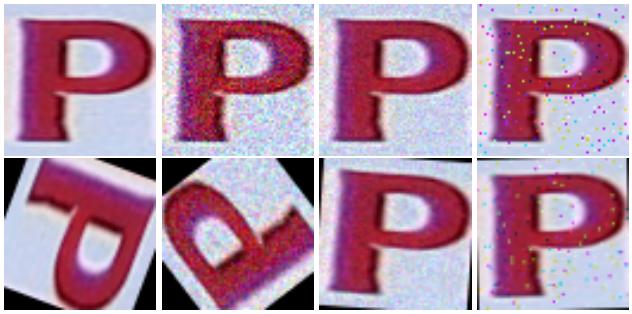

Figure 3. Top: Original, Gaussian, Poisson, and Salt Pepper Noise respectively, Bottom: Randomized rotations on top of the respective image directly above it

Note: data augmentation was only performed after we noticed that our accuracy rate training from scratch was capped at 0.82.

4.2. HOG + SVM

We treated the accuracy level of the effort of HOG + SVM as our baseline. We tested on the streetview characters. Unlike the CIFAR-10 data, we originally thought color was irrelevant in this classification task. Hence, we first performed HOG feature extraction from the grayscaled images, followed by linear SVM.

We performed our experiment on both the digit (0-9) subset and the entire dataset. In the digit subset, we have 2125 data total and we randomly picked 1400 of them to train the SVM, and have 350 data points for validation and finally test the model on 350 data points. Using HOG features + linear SVM, we have achieved an accuracy of approximately 0.76 on both the validation set and the test set for digits exclusively. However, when we trained SVM on the entire dataset, it only obtained an accuracy of 0.45. The

results show that HOG feature is good at classifying just numbers, but it is not sufficient to distinguish all the numbers and English characters.

We are still trying to find a way to visualize the weight of the HOG features. If the background area carries relatively small weights as compared to that of the foreground, it may well suggest that the model is robust to the background, and thus there is no need for us to include background subtraction as part of the preprocessing step .

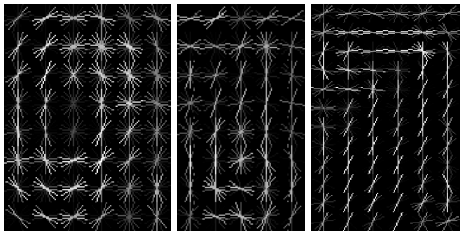

Figure 4. HOG features for 0, 4 and 7 respectively

4.3. Convolutional Neural Network

We tried different CNN architectures and the results are summarized in Table2 (before data augmentation) and Table3 (after data augmentation). We have included the results on synthesized fonts in Table1 as reference. We will discuss the details of the architectures in this section.

In general, we kill the process whenever the loss rate stops to drop. In our case, most of the networks stops to learn after 2000-3000 iterations. If any network stops learning before this, we decrease the learning rate. If the accuracy rate is way lower than expectation after many iterations, we would increase the learning rate. We detect the need of stronger regularization by comparing the loss when evaluating against training and test set.

4.3.1 Transfer Learning using MNIST

We first tried transfer learning on LeNet5 as we think the dataset might not be large enough to train the net from scratch. We only changed the number of output in last FC layer (from 10 to 62) and keep the remaining layers unchanged. We obtained an accuracy of 0.76 on 'good images', which is far better than the baseline. However, in order to do transfer learning on LeNet5, we resized all the images to 28 by 28, which actually doesn't allow too much variation on the archetecture. In fact, we suspect that compression like this might have removed some useful details from the images. The limited variation in MNIST datasets (only 10-class digits, all similar thickness, similar borders) doesn't help as compared to training synthesized font from scratch.

4.3.2 Transfer Learning from Synthesized Fonts

Streetview images are printed fonts, which are more similar to the synthesized fonts than to handwritten ones. In addition to this, there are approximately 9 times more synthesized fonts present in the dataset than that of the natural images. Thus, we hypothesized that performing transfer learning from synthesized font might provide a good basis for fine tuning when it comes to training on natural images. Besides, it's easy to augment the font datasets.

We prefer this instead of fine tuning directly based on LeNet5, because in the case of synthesized fonts, the input size is 64 by 64, which allows more pooling layer than in LeNet5, which input size is only 28 by 28. Increasing pooling layers might allow the CNN to visualize more general structures. Hence, we trained different CNN on Fnt data and then did transfer learning. In this case, we tried both training from scratch and fine tuning from LeNet5 on synthesized fonts.

Starting from four layer networks, we tested different parameters for convolutional layer (Conv) and fully connected layer (FC) and we added dropout and LeakyReLu to reduce overfitting and increase accuracy. In our experiments, dropout was helpful but LeakyReLu didn't work as we expected. When we increased to six layers, it achieved 0.90 accuracy on Fnt test data. We didn't try deeper neural networks as we thought six layers were sufficient for a problem of this scale. In the end, we did transfer learning and get an accuracy of 0.79. (We tested tranfer learning on all the previous font CNNs, but the results were similar.)

4.3.3 Training from Scratch

In the end, we trained CNN from scratch on natural images. We obtained the same accuracies on two different strategies tried. First we trained on 'good' images and tested on 'good' images. Then we first trained on 'bad' images, then trained on 'good' images. Both of them gives the same answer. The final model achieved 0.60 on 'bad' images and 0.82 on 'good' images.

We have tested on increasing the number of layers. There wasn't any noticeable difference, in terms of both loss history and both train and test accuracies. Overfitting wasn't an issue.

4.4. What Did Not Work

Noticing the early stagnancy in loss drop despite slow learning rate, we suspected saturation hence applied leaky ReLU. Howevever, we did not notice any improvement. We suspect the problem lied in the lack of training data, hence we applied data augmentation.

Unfortunately, data augmentation did not help significantly, this might be because the noise added was not sufficiently strong. If the noise were stronger, it would have

mimicked the effect of obscuring the edge, as well as a noisy background. Another possible problem is that we randomized the rotation, whereas in reality most characters segmented are only slightly slanted. Perhaps capping the rotation to a small range might be more helpful.

Training on synthesized fonts by transfer learning from LeNet5 was not useful, either. Even though the initial accuracy rate was higher, this edge was quickly catched up after a few hundred iterations. Maybe this is because the structure of MNIST digits was overly simplistic for our usage - uniform thicknesses, similar edges, nearly binary images, and the absence of background. These are precisely the factors that our dataset lack.

Transfer learning from synthesized fonts did not make things better. The synthesized characters in this dataset are only made up of 4 different fonts, whereas the fonts in natural scene characters have much more varieties. In fact, this might be the reason of high accuracy rate while training synthesized fonts from scratch.

4.5. Results

Architecture	Accuracy on fnt
[Conv ->ReLu] * 2->Pool -> FC (1024) ->ReLu ->FC (62)	0.84
[Conv ->ReLu] * 2->Pool -> FC (4096) ->Dropout (0.5) -> ReLu ->FC (62)	0.86
[Conv ->LeakyReLu] * 2-> Pool ->FC (4096) -> Dropout (0.5) -> LeakyReLu ->FC (62)	0.85
[Conv ->ReLu] * 2->Pool ->FC (1024) ->ReLu ->Dropout (0.5)->FC (62)	0.88
[Conv ->ReLu ->Pool] * 2->[Conv -> ReLu] * 2 ->FC (2048) ->ReLu -> Dropout (0.5)->FC (62)	0.90

Table 1. Accuracy on Fnt images for various CNN architectures

Model	Accuracy on Streetview Images
SVM on numbers	0.76
SVM on numbers & characters	0.45
CNN from scratch (RGB) [Conv->ReLu ->Pool] * 2 ->Conv->Relu ->FC (2048) ->ReLu ->Dropout (0.5) -> FC (62)	0.82
CNN transfer learning (GrayScale)	0.79

Table 2. Accuracy on Street View images, before data augmentation

Model	Accuracy on Streetview Images
CNN from scratch (RGB) [Conv->ReLu ->Pool] * 2 ->Conv->Relu ->FC (2048) ->ReLu ->Dropout (0.5) -> FC (62)	0.76
CNN from scratch (RGB) [Conv->ReLu ->Pool] * 2 ->[Conv->Relu] * 2 ->FC (2048) ->ReLu ->Dropout (0.5) -> FC (62)	0.81
[Conv ->ReLu ->Pool] * 2->[Conv -> ReLu] * 2 ->FC (2048) ->ReLu -> Dropout (0.5)->FC (62)	0.82

Table 3. Accuracy on Street View images, after data augmentation

5. Conclusion

From transfer learning on MNIST to transfer learning on Fnt data to training from scratch on natural image data, we gradually modified our strategy and tried out new architectures. We found that dropout, adding more layers, using RGB instead of grayscale worked while transfer learning, while LeakyReLu didn't help (details are discussed in section 4.4). Finally, we achieved an accuracy of 0.82 on 'good images' and 0.60 on 'bad images'. This result is far better than the baseline 0.55 which uses traditional methods.

There remains a lot to do. First of all, we haven't analysed the error distribution of CNN. It's hard for human to distinguish between '1' and 'l', 'O' and '0', '1' and '7', so how does CNN perform on these classes? We plan to use the IPython user interface in Caffe to check the result. We can use the 'heat map' [6]and 'data gradient'[5]to visualize the area that CNN is using to make classification and see if it can ignore the background. Currently, we still don't know how to use Caffe framework to classify batches of images

(seems that we can only do this through the IPython UI). Once we figure this out, we will test our model on Kaggle competition [1].

References

[1] T. E. de Campos, B. R. Babu, and M. Varma. Character recognition in natural images. In *Proceedings of the International Conference on Computer Vision Theory and Applications, Lisbon, Portugal*, February 2009.

[2] Teo de Campos, Bodla Rakesh Babu, and Manik Varma. Character recognition in natural images. 2009.

[3] 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. *arXiv preprint arXiv:1312.6082*, 2013.

[4] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[5] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[6] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Computer Vision–ECCV 2014*, pages 818–833. Springer, 2014.

[1] http://www.kaggle.com/c/street-view-getting-started-with-julia