Character Recognition from Google Street View Images

Indian Institute of Technology

Course Project Report – CS365A

By

Ritesh Kumar (11602) and Srikant Singh (12729)

Under the guidance of

Professor Amitabha Mukerjee

Abstract

Character recognition is the conversion of printed or handwritten text (present in an image) into machine – encoded format, so it can stored efficiently, searched and edited more quickly or used as an input to text mining and other such applications. Our project aims to recognize characters from natural images obtained from Google Street View. We have tried multiple algorithms to come forward with the most suitable one for this problem. We have employed Matlab and Python as the tools to implement these algorithms.

Acknowledgement

We would like to express our sincere thanks to our Instructor-In-Charge Professor Amitabha Mukerjee for allowing us the opportunity to undertake a project in his course. It was a quite a good learning experience as in a subject like Artificial Intelligence, not everything can be covered in class and hence a project gives us the opportunity to explore many interesting concepts.

Sincerely,
Ritesh Kumar (11602),
Srikant Singh (12729),
Indian Institute of Technology, Kanpur.

Table of Contents

Introduction

Our project aims at identifying characters of the English dataset (which comprises of the English alphabets and Hindu-Arabic numerals) from natural images (which includes images taken using handheld devices) obtained from Google Street View. The dataset has images of different sizes, along with different camera angles, lighting environments and image qualities. We have further reduced all the images to a size of 20*20px for further processing. Now our approach to this problem involves two steps, firstly we implement feature extraction from the images, then using the so obtained features, we train our model to predict the class of the given image.

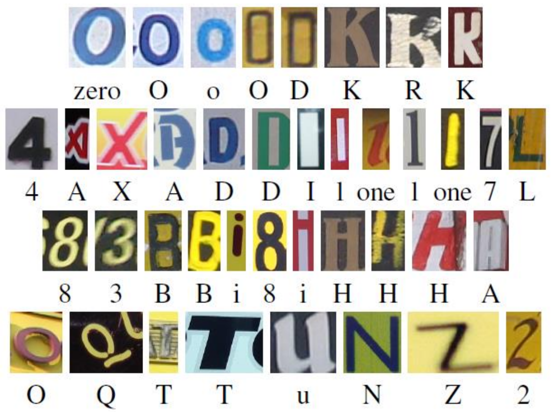

Figure 1 – Some of the Images which we use in our project

src: www.ee.surrey.ac.uk

Motivation

In the recent years, Google street view has increased in popularity manifold. With its increased usage, it becomes important to have a proper method to recognise images from Google street view. Such an implementation would have many benefits, like if we can recognize and tag such images, a lot of extra, useful information can be added to Google Maps. A good image recognition method can act as a precursor to many applications, like Image-to-Speech App for visually impaired people, which would help them to navigate and recognise their destinations. It can also be used to track various signs, hoardings and advertisements, by recognising their images and maintaining a count on them.

Related Works

Though a new avenue for research, many interesting works have been done in this field. One of the works involved text detection using sliding window followed by SVM and text recognition using Tesseract (an open source OCR), with a nominal success rate [1] .Another work involved the DistBelief [2] implementation of deep neural networks operating directly on the image pixels [3] .A very interesting work involved using 2 types of neural network – A thin deep neural network (Google Net) and a flat shallow neural net (Alex Network), with quite good results and high accuracies [4].

Dataset

We have used a very widely available and popular dataset known "Chars74K dataset" [5], which contains characters of both English character set as well as of the Kannada script. We however have based our project solely on the English character set. Furthermore as we are only interested in images obtained from Google Street view, our work mainly revolves on the set of 7705 characters obtained from natural images. The dataset is divided 62 classes (0-9, A-Z and a-z).

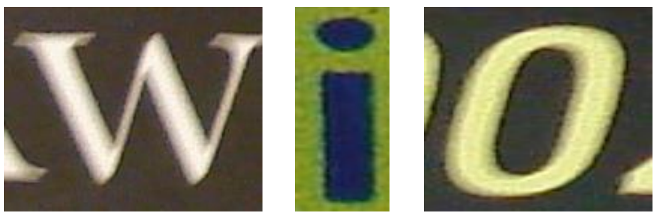

Figure 2 – Images from the Chat74k dataset

Rejected Methods

- *Use of Decision tree as a classifier* – Decision tree was considered to be used for classification, but was rejected on the grounds that, as they grow deeper, they start learning irregular patterns and show over fitting, a problem which arises due to the tendency for decision trees to show low bias and high variance.

- *Use of SIFT for feature extraction* – We also used SIFT with SVM for image recognition, but on a very constrained and small subset of the dataset, we were able to get an accuracy around 50%. As we increased the subset, the accuracy started dropping manifold. Also we had trouble running the algorithm for the full dataset as we ran into memory errors and issues. Hence the use of SIFT along with SVM was rejected.[6]

Methodology

Feature extraction –

1. *Image pixel vector* - The simplest method to extract image features was to use the image pixels themselves. We used the intensity of image pixels as a feature vector to train the learning models. We used the greyscale version of the image for processing. This method is quite fast as compared to other feature extraction methods but is one of the low accuracy models for feature extraction.

2. *Histogram of Oriented Gradients* – HOG is a feature descriptor in which we basically divide the image into cells and find the local histograms of gradient features over pixels in each cell, and after normalization the descriptors are fed into the learning model. We used the "extractHOGFeatures" function of MATLAB which returns a visual output to help determine the right cell size to use. We used cell sizes of 2*2, 4*4 and 8*8. As is visible in Figure 3, the best intuitive cell size is 2*2, but as we decrease the cell size, our computation complexity increases, while on larger cell sizes, enough information is not encoded for training.[7] Hence we used the cell size to be 4*4.

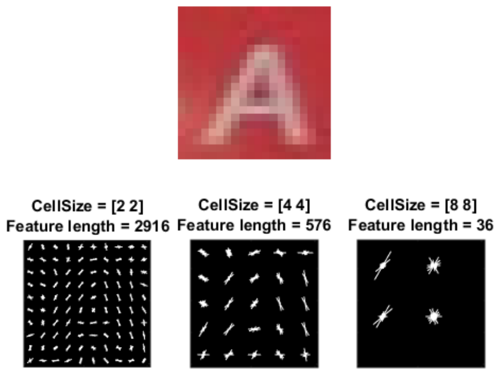

Figure 3 – Image extraction using HOG

Feature extraction –

We used three different techniques for training our model on the features that we obtained by our feature extraction methods –

1. *Random Forests* – They are extensions of decision trees incorporating averaging between multiple decision trees to give better results. They show lower variance than decision tree and there is no over fitting with increase in number of classifiers. The implementation of random forest was done by importing the "sklearn.ensemble" module in python. It gives better accuracies than LOOF-CV and the SVM classifier, while using common features.[10] We used n_estimator = 100 and 'entropy' criterion for our random forest model as it gives better accuracy.[8]

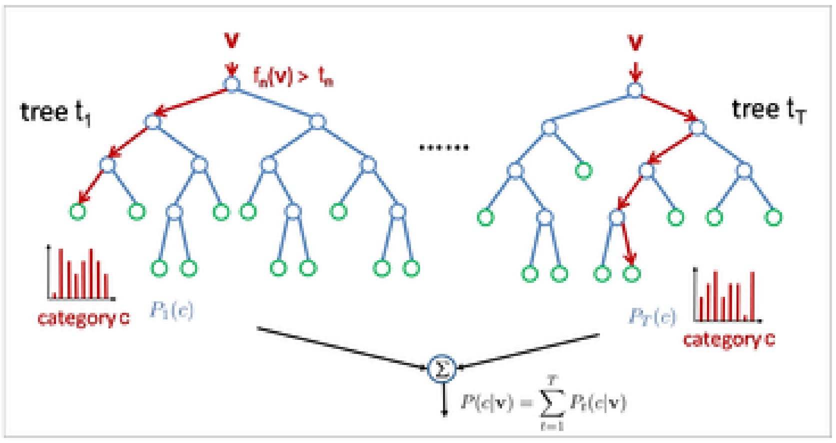

Figure 4 – Learning using a Random Forest

src: www.iis.ee.ic.ac.uk/~tkkim/iccv09_tutorial

2. *K – nearest neighbours with LOOF-CV* – LOOF-CV is present in many standard machine learning libraries, but here we take the advantage of the fact that LOOF-CV is particularly fast with k-NN. In LOOF-CV we simply remove one data point to test and train on rest, so it is just k-fold cross validation with k = 1. The k-Nearest Neighbour algorithm gives the most common label of the k nearest training point, which is then assigned to the test point. We tried different values of k in k-NN but k = 1 produced the best results. We used Euclidean distance function in k-NN.[8]

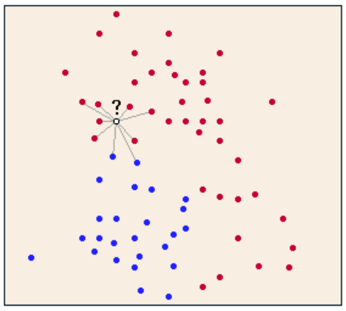

Figure 5 – k-NN classifier

src: http://www.statistics4u.com/fundstat_eng/cc_classif_knn.html

3. *Support Vector Machine* – Support vector machines are supervised learning that learn data, recognise patterns in them and based on those classify them. It is a non-probabilistic binary linear classifier. SVMs can also perform non-linear classification by using kernel methods implicitly mapping their inputs into high-dimensional feature spaces.[9] We used the "fitcecoc" function from Statistics Toolbox™ to create a multiclass classifier using binary SVMs and finally "predict" was used to predict the class of the test images.[7] We got an accuracy around 55% for a constrained subset of the dataset and the accuracy increased as we increased the subset to the full dataset.

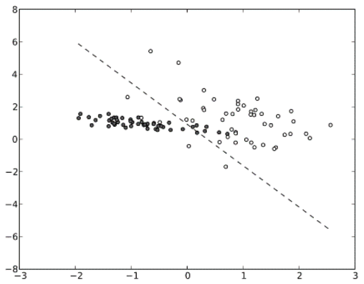

Figure 4 – Scatterplot showing a linear SVM's decision boundary

src: http://en.wikipedia.org/wiki/Support_vector_machine#/media/File:Linear-svm-scatterplot.svg

Results and Analysis

Method used	*Accuracy*
Pixel value vector – Random forest	*46.368%*
HOG – SVM	*77.029%*
Pixel value vector – kNN	*43.586%*

Table 1 – Methods vs. Accuracies

- Though random forests have been empirically proven [10] to be better than SVMs in classification, we can see that HOG-SVM gives quite better results than Pixel Value Vector – Random forests.
- From this we can safely conclude that HOG is a better feature extractor than Pixel Value Vector and feature extraction plays a significant role in final character recognition.
- The dataset we had didn't have any contextual information, as we only had single characters to be processes rather than text regions. Our methods suffered from declining accuracies as such, due to conflicting characters such as '0' , 'O' and 'D'; '1' , 'I' (uppercase 'i')and 'l' (lowercase 'L').

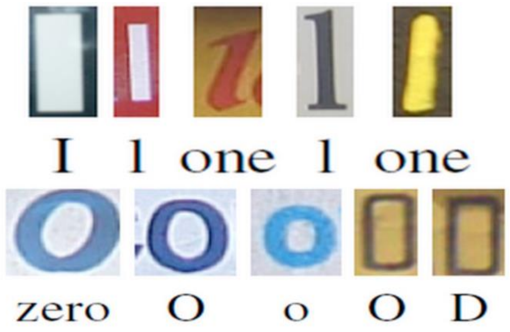

Figure 5 [4]

- In some cases (on manual inspection of misclassified images), it was found that cursive images were prone to be misclassified. For instance, this can happen if the classifier confuses a curly, cursive '4' with a similar looking '2', as shown in the figure below.

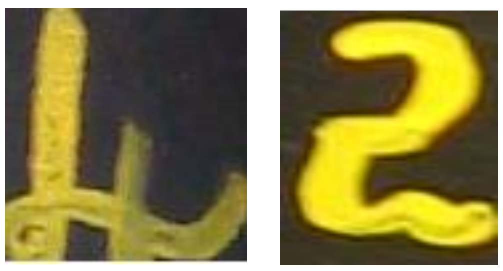

Figure 6

Future Prospect

- To solve the problem of declining accuracies and conflicting characters, we can implement boosting. Boosting allows the use of more than one classifiers, and in case of a conflict, we would have probabilities for an image to be classified as any particular character, which would be different for each classifier and a weighted sum across those can help us in resolving the conflicts.

- We intend to use neural networks as a classifier, as many character recognition algorithms have been developed using neural nets, among which deep convolution nets have a special mention. With the proper feature extractor, very good accuracies (of about 97.28% using GoogleNet and 91.22% using AlexNet)[4] have been obtained.

References

[1] Lintern, James. "Recognizing Text in Google Street View Images." Statistics 6 (2008).

[2] Dean, Jeffrey, et al. "Large scale distributed deep networks." Advances in Neural Information Processing Systems. 2012.

[3] Goodfellow, Ian J., et al. "Multi-digit number recognition from street view imagery using deep convolutional neural networks." arXiv preprint arXiv:1312.6082 (2013)

[4] Wang, Guan, and Jingrui Zhang. "Recognizing Characters From Google Street View Images."

[5] http://www.ee.surrey.ac.uk/CVSSP/demos/chars74k/

[6] https://github.com/shackenberg/Minimal-Bag-of-Visual-Words-Image-Classifier

[7] http://in.mathworks.com/help/vision/examples/digit-classification-using-hog-features.html

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

[9] http://en.wikipedia.org/wiki/Support_vector_machine

[10] Caruana, Rich, and Alexandru Niculescu-Mizil. "An empirical comparison of supervised learning algorithms." Proceedings of the 23rd international conference on Machine learning. ACM, 2006.