A Project Report

on

Text recognition in natural scenes

carried out as part of the course CS1634 Submitted by

Shivangi Pandey

159101132

6th sem Btech CSE

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

In

Computer Science & Engineering

MANIPAL UNIVERSITY
JAIPUR

Department of Computer Science & Engineering,
School of Computing and IT,
Manipal University Jaipur,
April, 2018

Table of contents :

Abstract

Detecting and recognizing text in natural images is a challenging problem that has recently received attention. Most methods attempting this have relied on elaborate models incorporating carefully hand crafted features or large amounts of prior knowledge. All methods assume the knowledge of a lexicon of 20-30 words that form the super set of all possible words that can occur in a test image. This makes the problem a lot easier than it would otherwise be. In this report, we present a method of detecting as well as recognizing text contained in natural images using Convolutional Neural Networks (CNNs). We use two CNNs, one trained for the character detection task and another for the character recognition task, as building blocks for the algorithm. We demonstrate state-of-the-art experimental results for two different tasks: character detection and character recognition, while also showing qualitative results for text recognition. We use the standard Chars74k and CIFAR datasets for the same.

2. Introduction

Being able to locate, detect and identify the text present in any natural image presents us with many opportunities. It finds application in many fields, ranging from brand detection recognition in images and videos to systems that aid blind people navigate. A system that is capable of automatically sifting through a set of images (from sources such as Instagram/Pinterest) and identifying presence of a certain brand is of immense monetary value to that brand. It allows for targeted marketing/advertising to a highly selective audience. However, the task of text detection, localization and recognition is a difficult problem, owing to a lot of variability present in

natural images, in terms of illumination, viewpoint, orientation and to a certain extent, even the intraclass variability that is inherent in characters of different fonts/styles and those written by different people. For this reason, a lot of text detection and recognition systems rely on cleverly hand-crafted features [2] [3] to represent images. Some state-of-the-art methods also rely on sophisticated models like conditional random fields, on top of the raw detection/recognition outputs in the end-to-end systems. In our work, given an image, we label the region of interest containing text using a bounding box and find the word predictions. A sample output of our full end-to-end text recognition system is illustrated in Fig 1. All characters occuring in the image are assumed to be in one of 62 possible classes (a-z, A-Z, 0-9).

2.1 Motivation

The automatic detection and recognition of text in natural images, text spotting, is an important challenge for visual understanding.
The human invention of language, and the cognitive ability to conceptualise, process, and communicate ideas through language, is one of the dening abilities of our species, and is associated with the outstanding intelligence of humans [Roth and Black, 2005]. Text, as the physical incarnation of language, is one of the basic tools for preserving and communicating information and ideas { indeed, text systems have been in existence for many millennia, since 6600 BC, when graphical symbols were carved in tortoise shells [Li and Wang, 2003]. Today, much of the modern world is designed to be interpreted through the use of labels and other textual cues, and so text nds itself scattered throughout many images and videos, and can often be the main semantic focus of an image. It is therefore imperative to create a visual interface with text to allow rich machine interaction with the human world.
Through the use of text spotting, we can enable automated systems to decode the text within images and videos, allowing a greater understanding of media.

3. Literature Review

We provide an overview of this thesis' target problem domain text spotting. We review a range of different methods for text spotting, and its constituent sub-problems of text detection and text recognition.

3.1 Text Spotting

This literature review will focus on the recognition of text from natural images, as opposed to recognising text from scanned (or fronto-parallel photographed) documents a much less complex problem.
Since text spotting is usually decomposed into two salient part, detection and recognition, with previous work sometimes constrained to an individual sub-task or the end-to-end text spotting problem as a whole, we focus on reviewing each sub-problem separately.

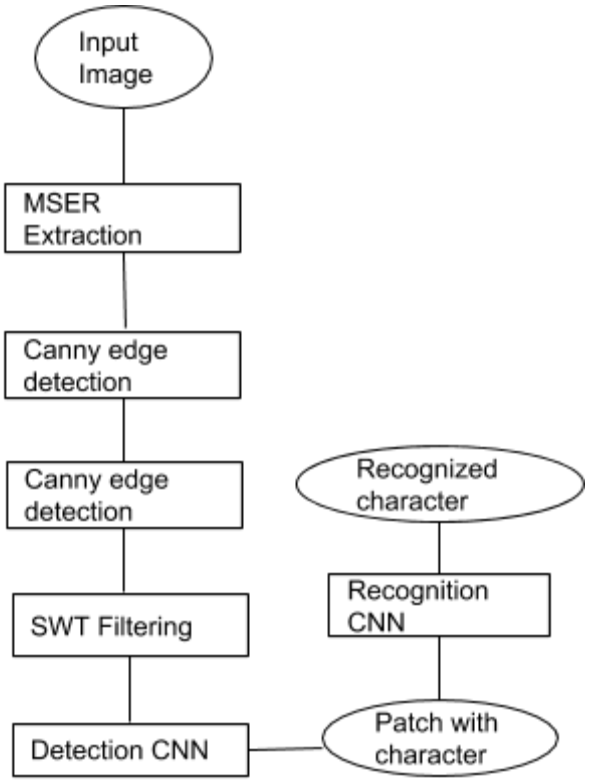
3.2 Related Work

Until recently, the end-to-end text recognition problem had only received a modest amount of interest from the vision community. In [6], the authors propose a recognition pipeline comprising multiple feedback loops for hypothesis validation. Briefly, they find Maximally Stable Extremal Regions (MSERs) and then classify these regions as either character or non-character regions. They form possible lines containing text and use character recognition, followed by typographic and language modeling to refine the recognition output. In [7], the authors use Random Ferns for character detection and the HOG descriptor [1] to describe image patches. They evaluate the word detection and recognition performance of a two-step approach comprising a text detector and an OCR engine. They show that their HOG-based system outperforms that using a conventional OCR engine. In [8], the authors use the output score of a Convolutional Neural Network (CNN) for character detection and another CNN for recognizing characters that have been detected. They use a multi-scale sliding window approach and consider windows in different rows independently. They make the assumption that text in the image occurs horizontally. For each row, the detection scores of different sliding windows are computed and NMS (non-max suppression) is used to suppress spurious peaks. This way, bounding boxes are constructed for rows with non-zero number of peaks. Another NMS is performed to prune bounding boxes with 50% overlap or more. The character recognition CNN is then used on each of these and beam-search is performed to output words. They assume the knowledge of a lexicon of about 20-30 words (which are different for different images, and potentially procured using Geo-location and other meta data), and they only output words from this lexicon. The method used in [7] also assumes the knowledge of such a lexicon.

4. Methodology and Framework

4.1 System Architecture

Deviating from some of the above approaches that rely on hand-crafted features, we use CNNs for this problem. We now briefly give an overview of our system
First we will do preprocessing on the image. In the proposed system preprocessing consist of mainly three steps. Candidate generation, Component level classifier and Non text filtering. Again candidate generation consist of a number of steps. Among a number of CC (Connected Component) extraction methods, we have adopted the MSER algorithm because it shows good performance with a small computation cost. The MSER(Maximally Stable Extremal Region) algorithm yields CCs that are either darker or brighter than their surroundings. We tackle the problems of detection and recognition in two cascaded stages. Filtering of the input image is done using canny edge detector. The output of the MSER extraction and filtering is combined using logical OR. Next step is to filter the image with the values of stroke widths. On the filtered image instance, we consider a sliding window over the whole span of the image after preprocessing the image at multiple scales. For each of these windows, we consider the image patch within that window and pass it to the character detection engine. We train the detection CNN in such a way that this character detection engine outputs a high score if there is a centered character in the input image patch. For these firings from the detection engine, we pass the corresponding patches to the recognition engine to determine which character is present in them. We use two Convolutional Neural Networks, one for the detection part and another for the recognition part.



Overview of proposed system

4.2 Algorithms and techniques

4.2.1 MSER Extraction

Among a number of CC extraction methods, we have adopted the MSER algorithm because it shows good performance with a small computation cost . This algorithm can be considered as a process to find local binarization results that are stable over a range of thresholds, and this property allows us to find most of the text components. The MSER algorithm yields CCs that are either darker or brighter than their surroundings. We have illustrated brighter CCs by assigning random colors to them. Note that many CCs are overlapping due to the properties of stable regions. More MSER extraction examples which show brighter and darker CCs, respectively.

4.2.2 Canny Edge Detection (Laplacian filter)

Canny edge detection is a technique to extract useful structural information from different vision objects and dramatically reduce the amount of data to be processed. It has been widely applied in

various computer vision systems. Canny has found that the requirements for the application of edge detection on diverse vision systems are relatively similar. Thus, an edge detection solution to address these requirements can be implemented in a wide range of situations. The general criteria for edge detection include:

1. Detection of edge with low error rate, which means that the detection should accurately catch as many edges shown in the image as possible
2. The edge point detected from the operator should accurately localize on the center of the edge.
3. A given edge in the image should only be marked once, and where possible, image noise should not create false edges.

4.2.3 Stroke Width Transform

The Stroke Width Transform (SWT for short) is a local image operator which computes per pixel the width of the most likely stroke containing the pixel. The output of the SWT is an image of size equal to the size of the input image where each element contains the width of the stroke associated with the pixel. We define a stroke to be a contiguous part of an image that forms a band of a nearly constant width. We do not assume to know the actual width of the stroke but rather recover it. The SWT operator described Here is linear in the number of edge pixels in the image and also linear in the maximal stroke width, determined at the training stage. To extract connected components from the image, SWT is adopted for its effectiveness and efficiency. In addition, it provides a way to discover connected components from edge map directly.

4.2.4 Applying sliding window

On the filtered image instance, we consider a sliding window over the whole span of the image after preprocessing the image at multiple scales. For each of these windows, we consider the image patch within that window and pass it to the character detection engine.

4.2.5 Detection CNN

The detection CNN takes as input a 32x32 image patch from the sliding window and outputs the probabilities associated with two classes:

- character present
- character absent.

We call the probability associated with the character present class as the ‘detection score’ of that input patch. We use character containing patches from Chars74k dataset [2] as positive examples and images from CIFAR-10 dataset [4] as negative examples, in addition to synthetically generated images as described in the following section on data augmentation.

4.2.6 Recognition CNN

For the recognition task, we trained our CNN using the 70,000 images from the Chars74k dataset, and classified each input image into one of the 62 possible classes. We used 45,000 images for training and 12,000 for testing and validation.

4.3 Network Architecture

As explained before, we use two CNNs, one for the character detection task and another for the character recognition task. We have used 2 CNN architectures in total, and we label them as CNN-1 and CNN-2. The architectures are enumerated in Tables 1 and 2. We use rectified linear units (ReLU) as our non-linearities and use dropout as regularization.

Detection CNN

Layer	Architecture
CONV1	filter:5; stride:1; pad: 2; depth:32
POOL1 (MAX)	filter:3; stride:2
ReLU1	-
CONV2	filter:5; stride:1; pad: 2; depth:32
ReLU2	-
POOL2 (AVE)	filter:3; stride:2
CONV3	filter:5; stride:1; pad: 2; depth:64
ReLU3	-
POOL3 (AVE)	filter:3; stride:2
Affine	output: 64
Affine	loss: softmax

Table 1

Recognition CNN

Layer	Architecture
CONV1	filter:5; stride:1; depth:20
POOL1 (MAX)	filter:2; stride:2
CONV2	filter:5; stride:1; depth:50
POOL2 (MAX)	filter:2; stride:2
Affine	output:500
ReLU	-
Affine	loss: softmax

Table 2

5. Work Done

5.1 Datasets and Evaluation Metrics

We evaluate the two CNNs separately on the Chars74K dataset [2]. The English dataset contains 70,000 images of characters from natural images and characters of different computer fonts. The dataset comprises annotated characters from 62 classes (A-Z, a-z and 0-9). For the character detection task, we trained our CNN using the 20,000 images from the Chars74k dataset as positive examples, and we used 20,000 images from the CIFAR-10 [4] dataset. The idea is to get the CNN to learn whether or not there exists a centered character in the current input image it is fed. We used 70,000 images for training and 12,000 images for testing and validation. For the recognition task, we trained our CNN using the 74,000 images from the Chars74k dataset, and classified each input image into one of the 62 possible classes.

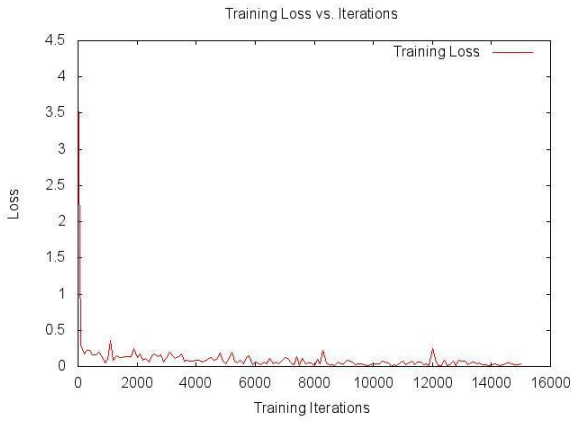
Training Data (60 %)	(45931, 32, 32, 1)	(45931, 62)
Validation Data (40 %)	(15268, 32, 32, 1)	(15268, 62)
Test Data (40 %)	(15268, 32, 32, 1)	(15268, 62)

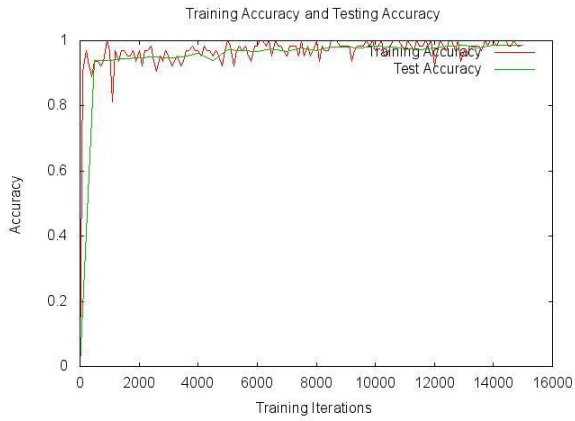
6 Experimental Results

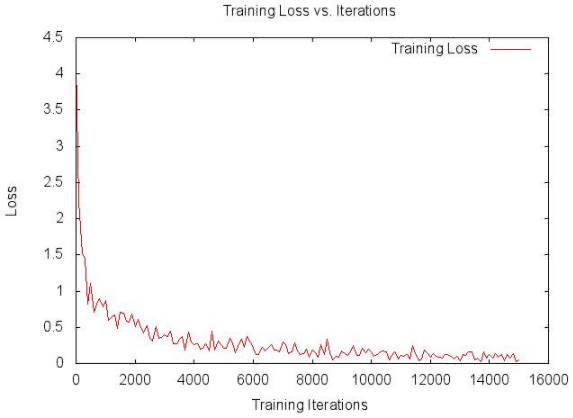
We have used both CNN 1 and 2 architectures for the character detection task and CNN 2 for the character recognition task. The batch sizes during Stochastic (mini-batch) Gradient Descent (SGD) were 500 images for detection CNN and 1000 images for recognition CNN. The results have been summarized in Table 3. We see that with architecture 2, our CNNs achieve higher accuracies than the state-of-the- art method in [8], for both the character detection and character recognition tasks.

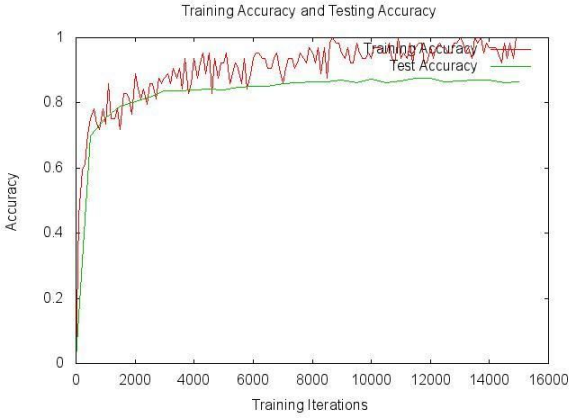
Accuracy			Hyper-parameters			
Training	**Validation**	**Test**	**Batch Size**	**Alpha**	**lambda**	**Epochs**
Result 4						
96.0	84.3	83.9	1000	0.001	1.5	3000
Result 5						
95.4	84.2	83.7	1000	0.001	1.0	1650
Result 6						
96.7	83.9	83.4	1000	0.001	1.5	1500

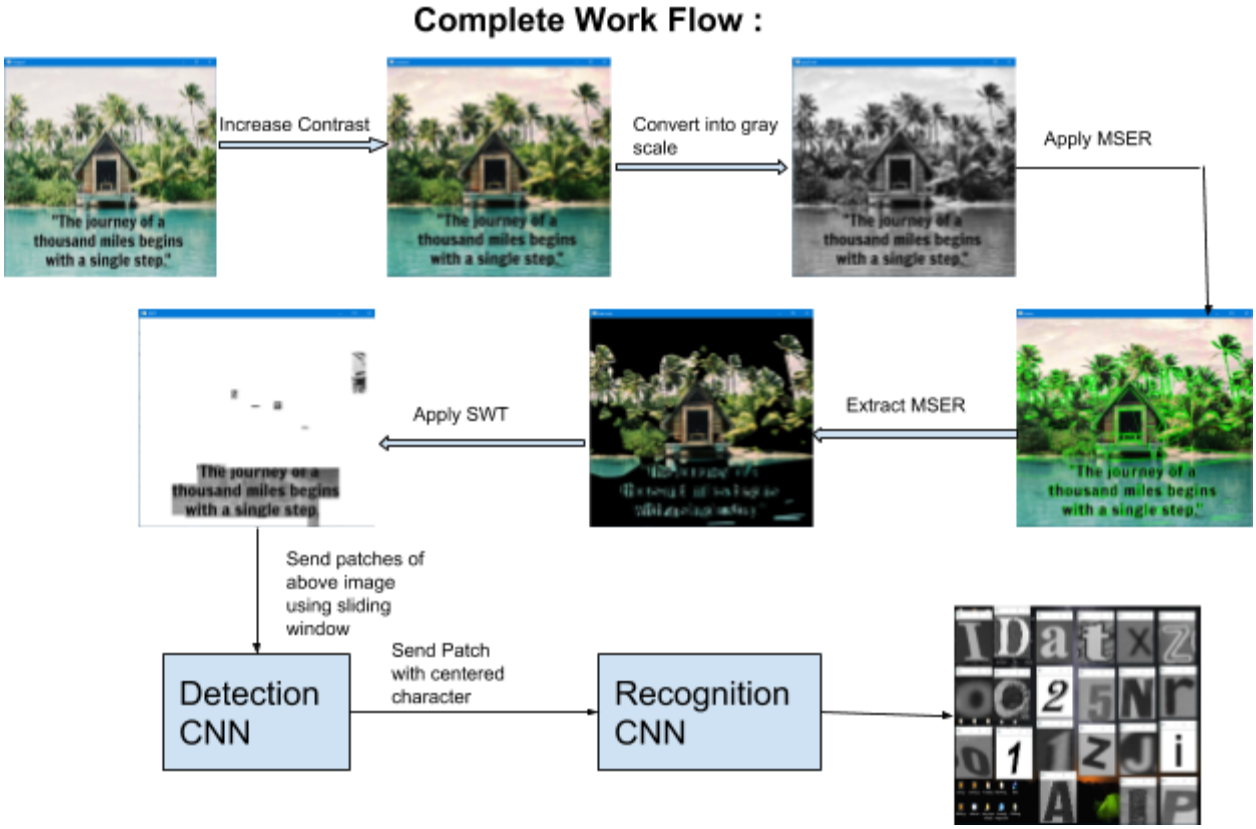
Accuracy			Hyper-parameters			
Training	**Validation**	**Test**	**Batch Size**	**Alpha**	**lambda**	**Epochs**
Result 1						
85.0	81.0	80.1	250	0.001	--	850
Result 2						
85.5	81.1	81.3	500	0.001	--	850
Result 3						
83.0	76.8	79.9	1000	0.01		850






Training Loss vs. Iterations
Training Loss
Loss
Training Iterations
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
0
2000
4000
6000
8000
10000
12000
14000
16000


Training Accuracy and Testing Accuracy
Training Accuracy
Test Accuracy
Accuracy
Training Iterations
0
0.2
0.4
0.6
0.8
1
0
2000
4000
6000
8000
10000
12000
14000
16000


Complete Work Flow :
Increase Contrast
Convert into gray scale
Apply MSER
Extract MSER
Apply SWT
Send patches of above image using sliding window
Send Patch with centered character
Detection CNN
Recognition CNN

7. Conclusion and Future Work

In this report, we have described a method to perform end-to-end text detection and recognition in natural images. We have demonstrated state-of-the-art results on character detection and recognition tasks.. In our experiments, we observed that the system gets confused between 0, O, D , 1 and I etc. To help resolve these cases better, it would help to have a CNN dedicated to predicting whether a given patch contains a letter or a digit, and after that predict the character. Secondly, since the letter I occupies less width than other letters, the sliding window that contains I will also contain its preceding or succeeding letters. So, it would be helpful to train the recognition CNN to recognize bi-grams that contain I, instead of I alone.

8. References

8.1 Research Papers

[1] Detecting and Recognizing Text in Natural Images using Convolutional Networks Aditya Srinivas Timmaraju, Vikesh Khanna Stanford University Stanford, CA - 94305

[2] TEXT DETECTION AND RECOGNITION IN NATURAL IMAGES Shabana M A1, Alphy Jose2, Ani Sunny3

[3] B. Epshtein, E. Ofek, and Y. Wexler. Detecting text in natural scenes with stroke width transform. Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on. IEEE.

[4] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. Computer Science Department, University of Toronto, Technical Report 1.4 (2009): 7.

[5] S. Lucas, A. Panaretos, L. Sosa, A. Tang, S. Wong, and R. Young. Icdar 2003 robust reading competition. ICDAR, 2003.

[6] L. Neumann and J. Matas. A method for text localization and recognition in real-world images. In ACCV, 2010.

[7] K.Wang, B. Babenko, and S. Belongie. End-to-end scene text recognition. Computer Vision (ICCV), 2011 IEEE International Conference on. IEEE, 2011.

[8] T. Wang, D. Wu, A. Coates, and A. Y. Ng. End-to-end text recognition with convolutional neural networks. Pattern Recognition (ICPR), 2012 21st International Conference on. IEEE.

8.1 Web Links

1. [1]https://ankivil.com/kaggle-first-steps-with-julia-chars74k-first-place-using-convolutional-neural-networks
2. https://www.coursera.org/googlecloud
3. http://machinethink.net/blog/machine-learning-device-or-cloud/
4. https://github.com/futurice/android-best-practices
5. https://medium.com/mindorks/how-to-pass-large-data-between-server-and-client-android-securely-345fed551651
6. https://www.youtube.com/watch?t=1770s&v=f6Bf3gl4hWY
7. https://zablo.net/blog/post/stroke-width-transform-swt-python
8. https://github.com/openpaperwork/libpillowfight
9. http://cv-tricks.com/tensorflow-tutorial/training-convolutional-neural-network-for-image-classification/
10. https://www.kaggle.com/pouryaayria/convolutional-neural-networks-tutorial-tensorflow