Bachelor thesis
Computer Science

Radboud University

Recognizing Text in the Wild

Author:
Twan Cuijpers
s4378911

First supervisor/assessor:
dr. Twan van Laarhoven
`T.vanLaarhoven@cs.ru.nl`

Second assessor:
prof. dr. Tom Heskes
`tomh@cs.ru.nl`

April 3, 2017

Abstract

Numerous applications are possible for system that is able to detect and recognize text in natural scene images. Extracting text from scene images is harder than document images because of varying fonts and backgrounds. The system in this thesis is based on work described by Wang, Wu, Coates and Ng in a paper and by Wu in a thesis. A convolutional neural network was used by Wang et al. for both detection and recognition of characters in an image.

Compared to Wang et al. a number of changes were made. The networks for detection and recognition do not use unsupervised learning of features. Also, batch normalization was added to both networks. An augmented dataset was made for training the character recognition classifier. To simplify the system, language model and lexicon were omitted. As experiment the CNNs were replaced with residual networks. Using a residual network for character recognition, a higher test accuracy was achieved than the ones mentioned by Wu.

Contents

Chapter 1

Introduction

Scene text recognition is the task of recognizing text in scene images. Scene images are images from the environment. A well known source of scene images is the service Google Street View. Existing OCR (Optical Character Recognition) solutions do not work well on scene images. OCR solutions are typically focused on recognizing text from document images with black characters in certain fonts on a white background. Natural scene images can differ from document images in used fonts, text orientation, shadows, backgrounds and other disturbing effects. Because of the different nature in which the text occurs, text recognition solutions for document images like Tesseract[1] do not perform well on scene images [20].

Various applications are possible for a system that is able to read text in scene images. In transportation such a system could be used to assist drivers by reading text on traffic signs. The ability to read vehicle number plates can be used by law enforcement or by the proprietors of parking areas. An other application could help visually disabled persons by reading text in their vicinity and passing it to a text-to-speech solution to help them perceive the environment.

The goal of this thesis is to partially reproduce a system used for reading scene text and try to improve it. The system described in this thesis is based on a paper written by Wang, Wu, Coates and Ng [21] and a thesis by Wu [22]. The thesis by Wu provides a more detailed description of the system that is also described in the paper. To solve the problem a sliding window detector using multiple scales is used on an image in grayscale. A convolutional neural network is used to determine whether a window contains text or not. The locations of the windows containing text serve as input for the non-maximum suppression. Non-maximum suppression reduces the number of windows when the overlap is above a given threshold. For character recognition another convolutional neural network is used. The input for this CNN is the set of bounding boxes after performing non-maximum

[1] `https://github.com/tesseract-ocr`

suppression. This system is able to read ‘text in the wild’ because the classifiers are able to detect and recognize characters from images with different fonts, backgrounds and lighting conditions. Since thirteen scales are used for the detector, it is likely that the character locations in the image will be considered. The system in this thesis differs from the one by Wang et al. [21] on some points. Some pre-processing steps and the language model were left out in this thesis. This thesis adds the use of batch normalization and experiments with deep residual networks.

For building and training both convolutional neural network classifiers the Python library Lasagne[2] was used. Lasagne is based on Theano[3], a Python library for efficient computations on large arrays. Training convolutional neural networks requires a lot of computing power. Often graphics processing units (GPUs) are used for this since they are faster than central processing units (CPUs) at performing vector and matrix operations in parallel.

The second chapter of this thesis discusses the background of the used methods. Chapter 3 describes how the system performs both text detection and text recognition using methods from the second chapter. Chapter 4 discusses the literature of text recognition in natural scene images. The fifth chapter presents the conclusions of this thesis.

[2]`https://github.com/Lasagne/Lasagne`

[3]`https://github.com/Theano/Theano`

Chapter 2

Preliminaries

This chapter contains descriptions of methods that were used to read ‘text in the wild’.

2.1 Sliding window detector

A sliding window detector is a method used for the detection of objects in images. The method uses a fixed size window and a classifier [22]. The window slides over an image both horizontally and vertically. A classifier is trained to detect a certain type of object. On each position of the window, the classifier determines whether a object has been detected.

A sliding window detector can make use of multiple scales. The scales are applied to the image over which the window moves and the size of the window stays fixed. Using multiple scales is useful for detecting objects in an image whose dimensions do not match with the fixed size window. In the case of an object, that is larger than the window size, after rescaling the image with a certain scale smaller than 1.0 the object fits into the window. When the object is a lot smaller than the window size, rescaling the image with a scale larger than 1.0 helps to make the object fit better into the window. There is a possibility that when the object is too small or too large for the window, that the classifier will not be able to detect the object inside the window. For all selected scales the sliding window detector runs over the image resized with that scale. To decrease the running time of the sliding window the step size, at which the window slides over the image, can be increased.

2.2 Non-maximum suppression

Non-maximum suppression (NMS) is a post-processing method used in computer vision [17]. The input for this post-processing method is the response map of a classifier used for object detection. The classifier that is used assigns a score to each window [22]. Reducing the number of overlapping windows is the goal of applying non-maximum suppression. The result of NMS is a smaller set of bounding boxes.

When a sliding window runs over an image, the same object can occur in multiple windows. The same object can be in the window when it slides a few pixels in all directions. Ideally there is only one bounding box per object in the image.

A common version of NMS starts with the highest scoring window [17]. Windows that are very close to the highest scoring window, and that do not have a maximal score, are suppressed. This ensures that a single object that previously occurred in multiple windows, now can only occur in a single bounding box. Whether a window is suppressed depends on the value of a threshold. Until there are no windows left, the window with the highest score is selected and its close neighbors are suppressed. Figure 2.1 shows an example of non-maximum suppression.

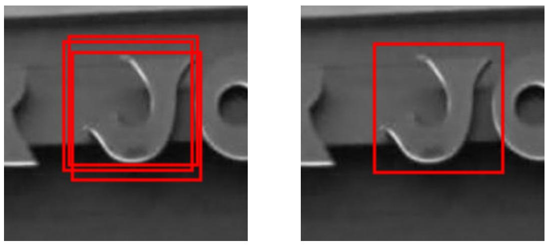

Figure 2.1: Desired effect of applying non-maximum suppression. Left: image with multiple boxes around the same object. Right: after applying NMS a single bounding box is left.

2.3 Artificial neural networks

The concept of artificial neural networks (ANNs) is based on biological neurons in brains [18]. The mathematical model of a neuron was made by McCulloch and Pitts in 1943. An artificial neural network is a network of nodes and links between them. The links in an ANN have weights associated with them. A node, also known as unit, has both incoming and outgoing links. The input function of a node is the sum of the input values multiplied with the weights. An activation function is a nonlinear function that determines the output values given the sum of weighted inputs. Examples of activation functions are step functions, sign functions and sigmoid functions.

A feed-forward neural network is a kind of artificial neural network where the outputs of a previous layer are the inputs of the next layer. The links only go in one direction; towards the next layer in the network. In a multilayer network the input layer is the layer with input units and the output layer consists of output units. If there are units between the input and output units, these units are called hidden units [16]. These hidden units reside in hidden layers. Since feed-forward networks do not contain cycles previous outputs have no influence on the computations. Networks that do contain cycles are so-called recurrent neural networks. The presence of cycles makes performing calculations less straightforward.

A simple kind of ANN is the perceptron; it consists of a single layer and is feed-forward [18]. As action function it uses a step function. Because the perceptron does not have hidden layers, the functions it can represent are limited. An example of function, that cannot be represented by a perceptron, is the XOR function [16]. In order to be represented by a perceptron, a function has to be linearly separable.

In order to train an artificial neural network, values for the weights of links between units and other parameters have to be found. The goal is finding values that minimize the error function on the training set. To find the minimum of the error function gradient descent is used. Gradient descent finds new weights and the step size at which this happens is called the learning rate. In networks that are more complex than perceptrons, there are multiple weights between input and output. Back-propagation learning is a method that looks at the outputs and tries to minimize the error by going backwards in the network to determine the weights that contributed to the error [18]. After determining the responsible weights, their values will be updated.

2.4 Convolutional neural networks

Convolutional neural networks (CNNs or ConvNets) are a kind of multilayer feed-forward neural network [22]. They are based on how the visual cortex in certain animals works [14]. The first uses of CNNs are from the early 90's. CNNs are mainly used on images and are not affected by small transformations on the input images [2]. A CNN consists of a combination of certain kinds of layers. A neuron in a layer of a CNN has an area with units from the previous layer as input [12]. A receptive field is the area in a layer that serves as input for a neuron in the next layer. In a CNN these receptive fields are local. This means that not all units in a layer serve as input for every unit in the following layer. Features found in early layers will be combined in later layers of the network. For training CNNs back-propagation can be used [2].

Convolutional neural networks are composed of the convolutional layers, sub-sampling layers and fully-connected layers. A convolutional layer has a number of filters, also called kernels. Filters are sets of weights used to make feature maps [13]. Those feature maps are areas that share the same set of weights. The same operation will repeatedly be performed on different parts of the input by the units of a feature map [12].

Sub-sampling or pooling layers are layers that perform a sub-sampling operation. The use of sub-sampling reduces the precision of where features were detected because the resolution of the feature map is reduced [13]. Pooling layers are placed after convolutional layers [2]. In pooling layers the output from the previous layer is taken and rectangular areas, 'pools', are taken comparable to a sliding window. The stride defines what step size is used to take the rectangular areas. The operation that is performed on the units in the rectangular areas depends on the kind of pooling that is used. When 'max pooling' is used the maximum of the units in the area is taken. In the case of 'average pooling' the results are the averages of units in the areas. Figure 2.2 shows an example of pooling.

1	1	2	3
3	1	4	1
5	5	3	2
4	6	7	5

3	4	4
5	5	4
6	7	7

3	4
6	7

Figure 2.2: An example of 'max pooling'. The left table is the output from the previous layer. The two other tables are the result of performing 'max pooling' with pool shape 2 by 2. The center table uses stride 1 and the right table stride 2.

Fully-connected layers are used as final layer in convolutional neural networks [2]. The number of outputs of the fully-connected layer is equal to the number of classes. Since all units of this layer are connected with all units from the previous layer, a large number of weights are involved.

The architecture of an CNN is description of what layers a network consists and their parameters. This includes the number of convolutional layers followed by pooling layers [2]. For convolutional layers the number of filters is specified and for subsampling layer the type of pooling with pool shape and stride can be specified. Figure 2.3 shows an example ConvNet architecture.

Batch normalization is a technique that can be used on deep neural networks to reduce 'internal covariate shift', a problem which makes training the network harder [9]. By applying normalizing on batches the initialization becomes less important and training the classifier will take fewer iterations. When batch normalization is applied to a network, higher learning rates can be used.

Residual networks (ResNets) are a kind of convolutional neural network

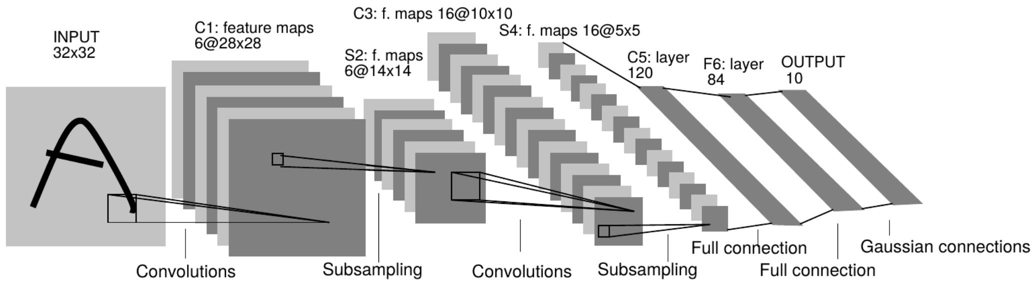

Figure 2.3: The architecture of LeNet-5. LeNet-5 is a convolutional neural network used for digit recognition [13].

with a large number of layers. In their work He et al. [7] added connections from a layer to another layer skipping layers in between. These shortcut connections add the output of a layer to the other layer. A residual block is a block of two convolutional layers with filter sizes of 3 by 3 and a shortcut connection skipping both convolutional layers. Batch normalization is used after every convolution. The number of layers of a residual network depends on how many times the residual blocks are repeated [4]. Problems usually associated with training networks consisting of a large number of layers do not occur when using the shortcut connections. In their paper He et al. [7] describe image recognition experiments with networks ranging from 18 to over 1000 layers using ImageNet and CIFAR-10 datasets. In a follow-up paper He et al. [8] describe improvements of their previous work about residual networks.

Chapter 3

Research

This chapter describes the approach used for reading text from natural scene images. The chapter is divided in sections about detection and recognition of text, what datasets were used and the experiments that were conducted.

3.1 Text detection

For text detection a sliding window was used with a convolutional neural network as classifier. The approach is based on the one described by Wang et al. and Wu and differs from it on a few points. As experiments different datasets and classifiers were used for the detection of text in images.

3.1.1 Sliding window detector

The sliding window detector used in this thesis uses a window of 32 by 32 pixels. The source image below the window is converted from color to grayscale before the window slides over the image. This is because the system in this thesis is based on that by Wang et al. [21], which makes use of grayscale images. An advantage of using grayscale images over color images is the smaller amount of data. A disadvantage is the loss of information since the three color channels are converted into one grayscale channel.

A sliding window with multiple scales is used to detect characters of varying size. Thirteen different scales are used for the image ranging from 0.1 to 1.5. Every step the detector moves one pixel, so all candidate windows of 32 by 32 pixels are considered at each of the thirteen scales. Compared to sliding the window one pixel per step moving a larger number of pixels at a time is faster but yields less accurate response maps. Figure 3.1 shows an image at different scales with a window.

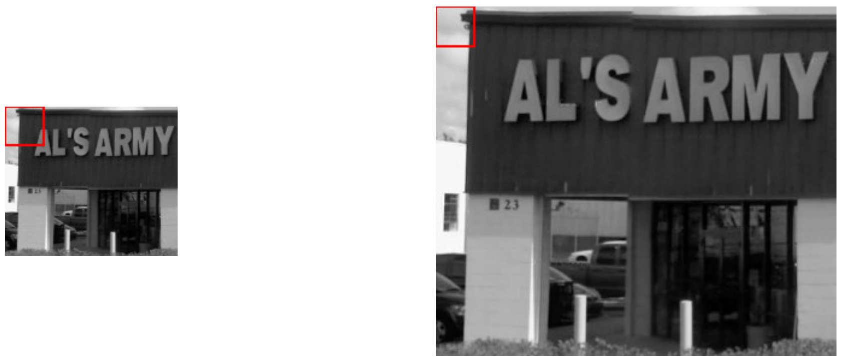

Figure 3.1: Part of an image from the Street View Text dataset [20] on the left at scale 0.3 and on the right at 0.7. The red square in the top-left corner is the window of 32 by 32 pixels. On the left, using scale 0.3, the individual characters on the storefront fit into the window. At scale 0.7 the characters do not fit into the window.

3.1.2 Text detection CNN

The convolutional neural network used for text detection has two classes; text and no-text. The architecture of this network is identical to the network used by Wang et al. [21]. As input the CNN receives grayscale images of 32 by 32 pixels, which is the size of the window. The first convolutional layer with 96 filters has an output of 25 by 25 by 96. Batch normalization is applied on this layer. A pooling layer comes after the first convolutional layer and performs average pooling. This results in a 5 by 5 by 96 output which serves as input for the next convolutional layer. This second convolutional layer, with 256 filters, also has batch normalization applied to it. Its output is of shape 4 by 4 by 256 and serves as input for an average pooling layer. At the end a fully-connected layer is located for this binary classification. Figure 3.2 shows the architecture of the convolutional neural network that was used.

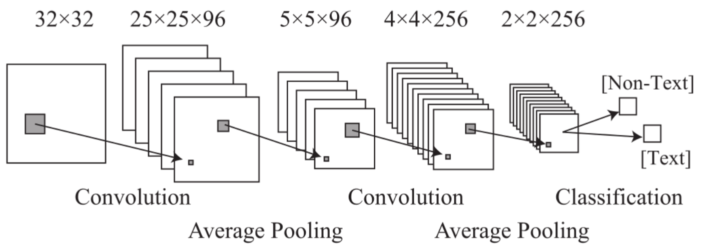

Figure 3.2: The CNN used for text detection by Wang et al. [21]. Image taken from their paper.

While the architecture of the network is identical, the method for learning the classifier in this thesis differs from Wang et al. in their paper. For both convolutional neural networks Wang et al. make use unsupervised pre-training [22], something that is absent in this thesis. They use an algorithm to extract features from the training dataset and use them as filters in the first convolutional layer of the networks. This thesis adds batch normalization to the classifiers for detection and recognition.

The detector classifier by Wang et al. was trained on a combination of multiple datasets. This combination included images from the ICDAR 2003 Robust Character Recognition dataset, the part of the Chars74K dataset [1]with characters from the Latin alphabet and images they generated themselves.

The network in this thesis was trained on a different combination of datasets with images of centered characters and images without centered characters. More about this combination of datasets can be found in subsection 3.4.1. Figure 3.3 shows the response map generated by the CNN trained on this dataset.

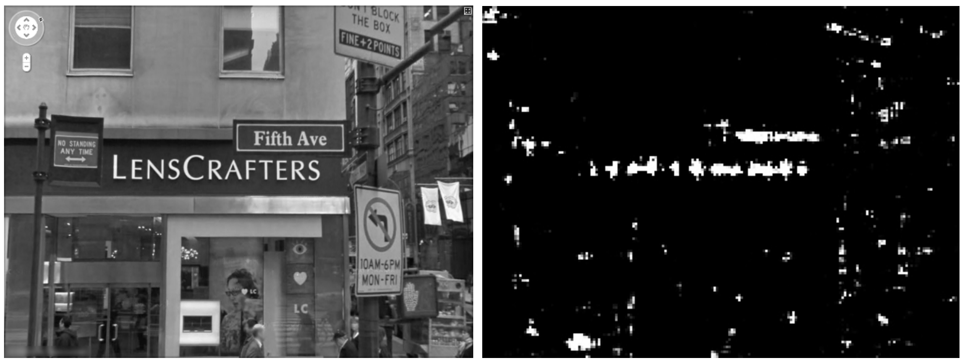

Figure 3.3: An image from Street View Text dataset at scale 0.6 and the corresponding detector response map at scale 0.6. White parts in the response map indicate the presence of text.

3.1.3 Non-maximum suppression

To retrieve boxes from the detector response maps the method and implementation by Wang et al. [21] were used. The result of the sliding window detector with the text detection network from subsection 3.1.2 is thirteen response maps. Each of these thirteen response maps, one for each scale of the sliding window detector, contains the scores from the text detection CNN classifier. In the response map a high score indicates the presence of a centered character. Because most lines of text are horizontally oriented, for all rows in a response map the peak scores will be identified. Depending on the number of peaks, how high the peak scores are and the distance

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

between them, a line will be identified as containing text when the average height of the peaks in close proximity to each other exceed a threshold. To determine the width of a text area, the outermost peaks are used. In words the distances between two characters are roughly the same. When the separations between peaks vary, there probably is space between the characters and the two text segments receive separate bounding boxes. All bounding boxes have a score attached to them that is determined by averaging the scores of the peaks inside that box.

When the bounding boxes for all thirteen scales have be found, non-maximum suppression is applied to the set of all boxes. When the overlap between two bounding boxes in NMS is above a certain threshold, the bounding box with the lowest score will be suppressed. The bounding boxes that remain are the local maximums in the sets of boxes. Ideally a text segment that appears in multiple boxes only appears in a single box after non-maximum suppression has been applied to the set of bounding boxes. This to prevent that one text segment in the input image of the system ends up as multiple text segments in the output of the system. Figure 3.4 shows boxes that were found. Changes in the methodology for finding boxes and additional post-processing steps could help improve the quality of found boxes.

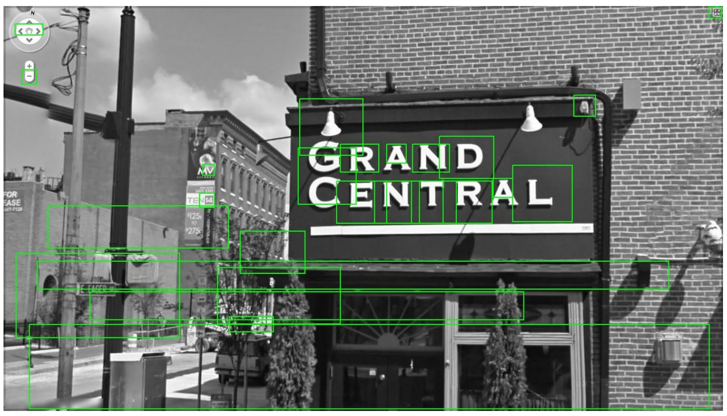

Figure 3.4: Boxes found in an image from the Street View Text dataset. A number of boxes do not surround text. Boxes on the storefront contain individual characters instead of the desired words 'GRAND' and 'CENTRAL'.

3.2 Text recognition

Like Wang et al. a CNN was used in this thesis for recognizing text. The end-to-end system combines the sliding window, the character detector classifier, non-maximum suppression and the network for character recognition.

3.2.1 Text recognition CNN

For recognizing characters a 62 class convolutional neural network is used. The 62 classes are the ten digits and the characters of the Latin alphabet in both lowercase and uppercase. The architecture of this classifier is similar to the architecture of the network used for determining whether a window does contain text, that is shown in figure 3.2. Just like the other network the input is a grayscale image of 32 by 32 pixels. This network for text recognition differs from the network for text detection in a few ways. On both convolutional layers of the network the number of filters is different. The first layer has 115 filters compared to 96 on the first convolutional layer of the detection CNN. On the second convolutional layer there are 720 filters instead of 256 filters. Wang et al. gave the network for recognition a higher number of filters than the network for detection because of the higher number of classes [22]. Like in the previous CNN batch normalization was used on both convolutional layers and the unsupervised pretraining was omitted.

The classifier was trained on an augmented version of the ICDAR 2003 Robust Character Recognition dataset. More about the augmented set can be found in subsection 3.4.2. Like the classifier for character detection, Wang et al. trained their classifier for character recognition on the ICDAR 2003 dataset but also on the Chars74K dataset and a synthetic dataset they made. For training both classifier they used around 100.000 images [22]. Table 3.1 shows a comparison of accuracies for character recognition.

Model	Accuracy
Wang et al. [19]	64%
Coates et al. [3]	81.7%
Wu [22]	83.9%
Alsharif et al. [1]	86.0%
This work	**78.6%**

Table 3.1: Accuracies on ICDAR 2003 Robust Character Recognition test set.

The test accuracy of this work in table 3.1 is lower than Wu, on which it is based. A possible explanation for this difference is the data used for training the classifiers. The classifier in this work was trained on an augmented version of the ICDAR 2003 Robust Character Recognition dataset. Wu trained his CNN classifier on the ordinary version of that dataset and two other datasets. Because the augmented set is based on a small set of images with a number of slight changes made to them, the variation in the character images is not much larger than in the original set. Possibly the combination of datasets used by Wu to train the classifier results in a better classification because of the more varied training material.

The CNN in this work does not use the unsupervised pretraining steps that Wu implemented to simplify the method. Therefore no pretraining results are used in the first convolutional layer of the network. Further small differences in the implementation may have contributed to the lower accuracy on the test set.

Section 3.4 features descriptions of experiments regarding character recognition. This includes training the CNN classifier on augmented datasets and using residual networks instead of a convolutional neural network.

3.2.2 End-to-end system

The end-to-end system combines the previously discussed sliding window detector with both convolutional neural networks. The input of the end-to-end system is an image file depicting a natural scene. Running the sliding window detector with the character detection classifier results in detector response maps for all thirteen scales. This part of the end-to-end pipeline was implemented in Python using the Lasagne library for the CNNs. For training the character detector a GPU was used.

The second part of the system receives detector response maps as input and tries to identify text lines. Boxes of text segment are generated from the response maps for all scales. Non-maximum suppression is applied to the set of bounding boxes for all scales to get rid of largely overlapping bounding boxes. For this part the MATLAB implementation made and published by Wang et al. [21] is used.

The third and last part of the end-to-end system receives the bounding boxes coordinates of the text segments as input. Separate images are made from these bounding boxes and resized to a height of 32 pixel while maintaining aspect ratio. A window of 32 by 32 pixels slides of these images and the character recognition classifier is used. Like the first part of the end-to-end system, the implementation was made in Python using Lasagne and a GPU was used for training.

As opposed to the system Wang et al. and Wu no language model or lexicon was used. The end-to-end output is determined solely by the sliding window in the third part. No post-processing techniques were used to restrict the output to a certain list of strings.

3.3 Datasets

This section contains descriptions of datasets that were used in this thesis. These datasets or parts of these datasets were used for training the classifiers for detecting and recognizing characters.

3.3.1 ICDAR 2003 Robust Character Recognition

ICDAR 2003 Robust Character Recognition is a one of the three ICDAR 2003 Robust Reading competitions [2]. The other categories being 'Robust Reading and Text Locating' and 'Robust Word Recognition'. The Robust Character Recognition dataset is divided in a test set and training set with both above 5000 images of characters. Images in the set are in color and have varying fonts, text colors, backgrounds and orientation.

The dataset used for character detection, from subsection 3.4.1, contains training data from this dataset. An augmented version of the training set was used for learning the text recognition classifier. Subsection 3.4.2 describes an experiment involving this dataset. The test set of this dataset was used to evaluate the performance of character recognition. Table 3.1 shows a comparison of test accuracies from this dataset and figure 3.5 contains examples of images from the ICDAR 2003 Robust Character Recognition dataset.

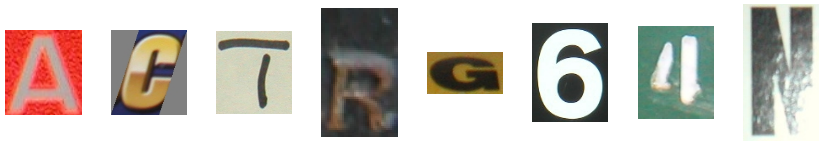

Figure 3.5: Characters from the ICDAR 2003 Robust Character Recognition dataset.

3.3.2 ICDAR 2003 Robust Reading and Text Locating

This dataset is also from the ICDAR 2003 Robust Reading competitions. The dataset consists of color images from objects with text on them. It is divided in a test set with 251 images and a training set with 258 images. All images come with coordinates to indicate what areas of the image contain text. The images in this dataset mostly of small objects and signs. Parts of this set were used to produce the dataset for character detection described as an experiment in section 3.4.1. Figure 3.6 contains some image from the Robust Reading and Text Locating dataset.

[2]http://www.iapr-tc11.org/mediawiki/index.php/ICDAR_2003_Robust_Reading_Competitions

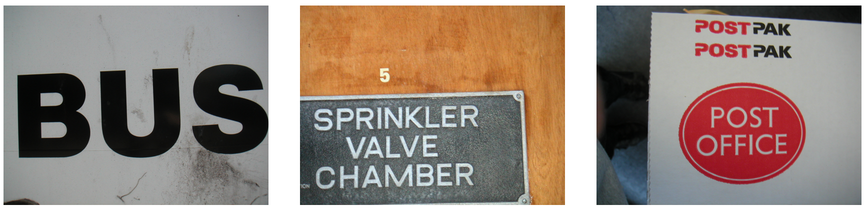

Figure 3.6: Images from the ICDAR 2003 Robust Reading and Text Locating dataset.

3.3.3 Street View Text

The Street View Text (SVT) dataset[3] by Kai Wang et al. [20] is a dataset containing images taken from Google Street View. It contains 350 color images with annotations. For each image there is a lexicon of words and a coordinates of bounding boxes in which some of the words from the lexicon occur. The text in images is mostly on storefronts, house numbers and banners. Like the previous dataset, parts of this dataset were used to learn a classifier for character detection in subsection 3.4.1. Figure 3.7 shows a few examples of images from the Street View Text dataset.

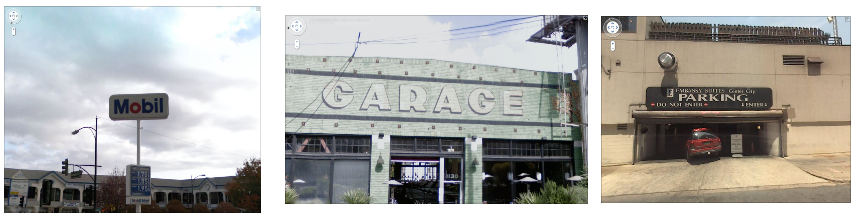

Figure 3.7: Images from the Street View Text dataset.

3.4 Experiments

This section describes the experiments that have been conducted. The focus of first two experiments is on different datasets for both character detection and character recognition. Residual networks, mentioned in the end of section 2.4, were used for both detection and recognition in the other two experiments.

3.4.1 Dataset for character detection

For character detection a dataset was made from the three datasets described in section 3.3. The text instances in the dataset are from the augmented version of the ICDAR 2003 Robust Character Recognition dataset described

[3]http://vision.ucsd.edu/~kai/svt/

in 3.4.2 No-text instances in the dataset are patches of 32 by 32 pixels, the size of the sliding window, taken from both the ICDAR 2003 Robust Reading and Text Locating and the Street View Text dataset. Both datasets come with coordinates describing rectangles with text in them. A sliding window was used on the images of these datasets and the patches, that did not intersect with the rectangles containing text, were used as no-text instances. Using this method both training set and test set were generated.

A toy training dataset for character detection was provided by Wang et al. and Wu in a package with parts of source code for their system. This toy dataset contains 15000 32 by 32 grayscale images. Figure 3.8 shows a response map using the toy dataset and the dataset made from three sets mentioned in section 3.3.

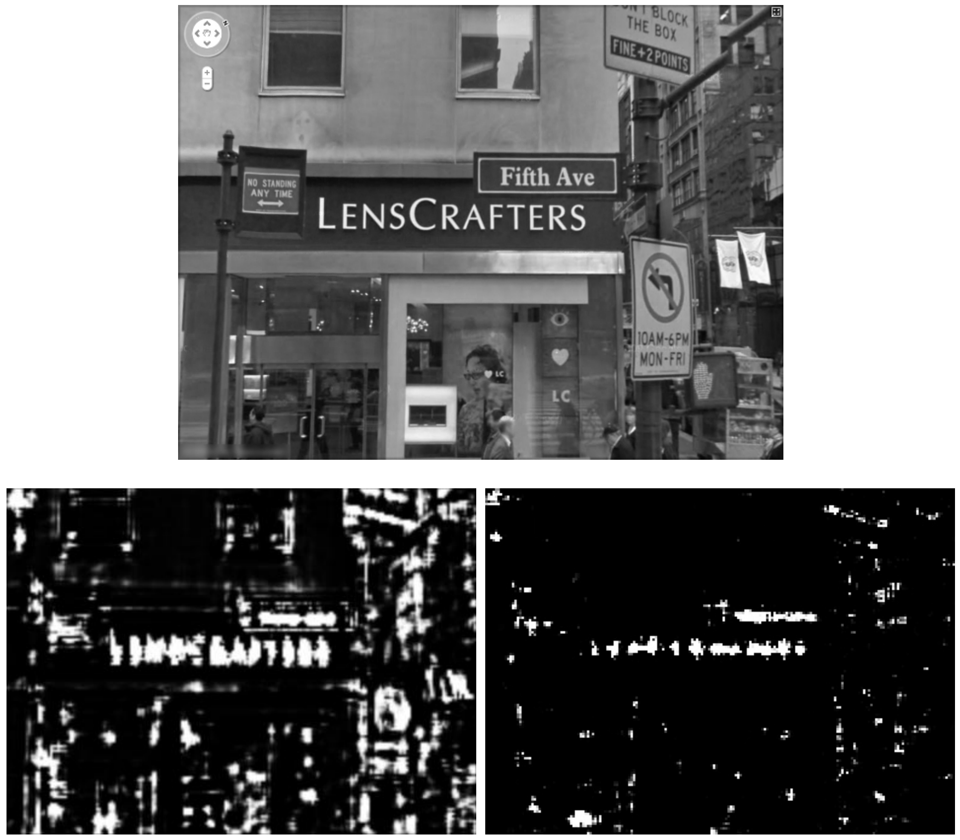

Figure 3.8: An image taken from the Street View Text dataset at scale 0.6, the response map using the small dataset by Wang et al. and Wu for training and the response map using the dataset from this subsection. White parts in the response map indicate the presence of text. The first and third image were taken from figure 3.1.

The first response map, using the dataset by Wang et al. and Wu, contains more white areas than the second response map and the white areas are less concentrated. The convolutional neural network trained on the combination of datasets achieved a test accuracy of 83.5% compared to 79.8% by the CNN trained on the toy dataset.

3.4.2 Augmented dataset for character recognition

Data augmentation was used to create a larger training dataset for the character recognition classifier from subsection 3.2.1. This dataset is based on the ICDAR 2003 Robust Character Recognition dataset described in subsection 3.3.1. The ICDAR 2003 Robust Character Recognition training set consists of 5980 color images of characters. The goal of the augmented dataset is to train a better performing classifier by using a larger training set. All images from the ordinary training set are also in the augmented training dataset. Three different operations were performed on images of the original set; rotations, shifts and inverting the grayscale values. Images were shifted a few pixels up, down, left or right. Rotations performed on the images are one, two or three degrees either clockwise or counterclockwise. With 95680 images the augmented training set is sixteen times the size of the original training dataset.

The test accuracy of the CNN when trained on the original ICDAR 2003 Robust Character Recognition dataset is 69.0%. Using the augmented version resulted in a test accuracy of 78.6%, the value mentioned earlier in table 3.1. The larger amount of training data resulted in an increased accuracy by almost ten percent.

3.4.3 Text detection with residual networks

Instead of using the detector CNN architecture from subsection 3.1.2, a residual network was used in this experiment. The reason for this is that ResNets, mentioned in section 2.4 about CNNs, were successfully used by He et al. [7] for object recognition on the CIFAR-10 and ImageNet datasets. For both training and testing the detection dataset from subsection 3.4.1 was used. The ResNet implementation was based a replication of the work by He et al. on the CIFAR-10 dataset from the Lasagne Recipes[4], a collection of examples for the Lasagne library in Python. Residual networks with 3, 5, 7 and 9 residual blocks with respectively 20, 32, 44 and 56 layers were used. The CNN originally used for detection had 83.2% accuracy on the test set. All four residual networks achieved test accuracies of around 83.6%. The use of ResNets for character detection appears to be only a minor improvement over the convolutional neural network.

[4]`https://github.com/Lasagne/Recipes`

Figure 3.9 shows that character on the storefront were detected both by the convolutional neural network and the residual network. The response map of the residual network has a smaller number of white spots, places classified as text, than the other response map.

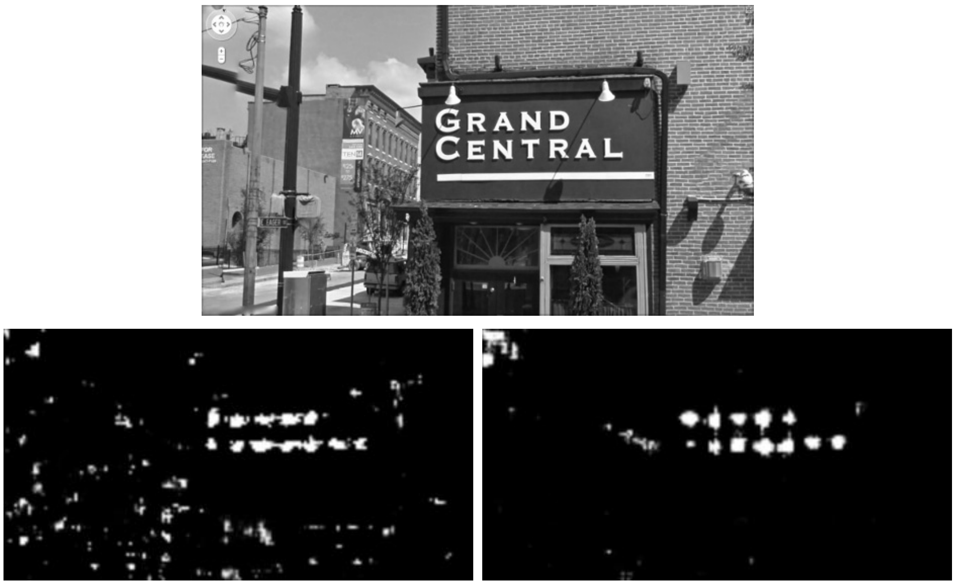

Figure 3.9: An image from the Street View Text dataset, the response map using a CNN and using ResNet-20 (20 layers; 3 residual blocks) at scale 0.4.

3.4.4 Text recognition with residual networks

As an alternative to the CNN architecture from subsection 3.2.1, a residual network was used for text recognition. Like the previous experiment with ResNets, the implementation was based on an example from the Lasagne Recipes. Table 3.2 contains the achieved testing accuracies with the CNN and the four residual variants.

Model	Accuracy
CNN	78.6%
ResNet-20	85.0%
ResNet-32	86.4%
ResNet-44	86.4%
ResNet-56	86.4%

Table 3.2: Accuracies on ICDAR 2003 Robust Character Recognition test set. Accuracy for CNN model was taken from table 3.1.

All residual networks perform better on the test set than the convolutional neural network. The accuracies on the ICDAR 2003 Robust Character Recognition set for the ResNets with 32, 44 and 56 layers are higher than all other entries listed table 3.1.

Chapter 4

Related Work

The task of recognizing text in natural scene images can be split in two parts; detection and recognition. In this chapter different approaches to text detection and text recognition from the literature are discussed.

4.1 Text detection

The goal of text detection is to find areas in an image in which text occurs. Three commonly used methods for detecting text in images are described below.

One method for detecting objects in general is the sliding window detector. It uses a fixed size window that slides over the entire image making small steps and a classifier to determine whether the window contains an objects. The image over which the fixed size window moves can be resized using multiple scales to detect objects with a size different from the window size. Different kinds of classifiers were used with a sliding window for text detection. Both Jaderberg et al. [11] and Wang et al. [21] used a CNN with two classes, text and no-text, to determine whether the current window contains a centered character. Instead of applying the sliding window on the source image, like Jaderberg et al. and Wang et al. did [21], HOG features can be used. HOG features, Histogram of Oriented Gradients, are used to describe the shape of objects in images. Mishra et al. [15] and Wang et al. [19] respectively trained a SVM classifier and random ferns on HOG features.

Another method used for text detection is Maximally Stable Extremal Regions (MSER). Extremal regions in images are areas that differ from other areas in intensity [1]. An advantage of this method is the fast computation of these extremal regions compared to a multiscale sliding window. Alsharif et al. used MSER to retrieve possible locations of text in the image and cluster them to find candidate lines of text.

Stroke Width Transform (SWT) is text detection method that tries to find parts that contain text based on stroke width [5]. Unlike the sliding window detector and MSER, SWT is a text specific detection method. SWT tries to determine for each pixel in an image what the width of the stroke containing that pixel is. These stroke widths are used to derive what areas contain text since characters usually have a constant stroke width opposed to other elements in the image. Jaderberg et al. make use of SWT for text detection [11] in a system for making annotations of images on Flickr.

4.2 Text recognition

The goal of text recognition is to identify characters and to form words and sentences. In this section various approaches found in the literature are discussed.

The output of the text detection phase usually is a set of bounding boxes around parts of text. Non-maximum suppression (NMS) was used to lower the number of boxes before recognition by Alsharif et al. [1], Jaderberg et al. [11] and Wang et al. [21].

Different classifiers were used to recognize characters in the literature. Most classifiers take an image of a character as input. For a comparable task, recognizing numbers in images from Google Street View, Goodfellow et al. [6] made use of convolutional neural networks for both localization and recognition of numbers. Both Wang et al. [21] Alsharif et al. [1] use a CNN classifier with character images as input for character recognition.

In their paper Jaderberg et al. [10] take word images as input for a CNN instead of character images. They trained the CNN on a synthetic dataset with images of around 90000 different English words from Hunspell[1]. Other systems, that take character images as input, form words out of the detected characters. To form these words models are used in conjunction with lexicons. A statistical model was used by Mishra et al. [15] to form words. Wang et al. [21] used beam search to obtain the words from a given lexicon.

[1]`http://hunspell.github.io/`

Chapter 5

Conclusions

The aim of this thesis was to locate and recognize 'text in the wild'. This problem is more complex than Optical Character Recognition because of less predictable fonts, image backgrounds and character orientation. For this a partial reproduction of the system described by Wang et al. [21] and Wu [22] was made. This work differs from the work by Wang et al. and Wu in several ways. Both the CNN for character detection and for character recognition in this thesis do not make use of unsupervised prelearning and were not trained on the datasets used by Wang et al. and Wu. Batch normalization, a method that should make learning networks easier, was added to both CNNs in this work. And in this work no beam search and lexicon are used to retrieve the end-to-end results.

For character detection a dataset was made that is based on three other datasets. The accuracies of CNNs and ResNets trained on this dataset are around 83%. The performance of the CNN used for detection could not be compared to Wang et al. because their test dataset and test accuracy were not published.

For finding bounding boxes with text the approach and implementation by Wang et al. and Wu was used. This approach, that assumes a horizontal orientation of the text, works reasonably with method for text detection.

The character recognition CNN classifier achieves 78.6% accuracy on the ICDAR 2003 Robust Character Recognition test set, which is lower than the 83.9% achieved by Wu [22]. This lower score can be explained by different datasets for training, lack of unsupervised pretraining and possible implementation differences. In the last experiment, using residual networks for character recognition, the highest accuracy was 86.4%. This exceeds the accuracy of 86.0% by Alsharif et al. [1] mentioned in table 3.1. For character recognition residual networks are a good alternative to the CNN classifier.

A possible area for improvement is in the retrieving text segment bounding boxes from response maps. Adding a language model and lexicon to restrict the output would make comparing end-to-end results possible.

Bibliography

[1] Ouais Alsharif and Joelle Pineau. End-to-End Text Recognition with Hybrid HMM Maxout Models. *arXiv preprint arXiv:1310.1811*, 2013.

[2] Christopher M Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[3] Adam Coates, Blake Carpenter, Carl Case, Sanjeev Satheesh, Bipin Suresh, Tao Wang, David J Wu, and Andrew Y Ng. Text Detection and Character Recognition in Scene Images with Unsupervised Feature Learning. In *2011 International Conference on Document Analysis and Recognition*, pages 440–445. IEEE, 2011.

[4] Mohammad Sadegh Ebrahimi and Hossein Karkeh Abadi. Study of Residual Networks for Image Recognition.

[5] Boris Epshtein, Eyal Ofek, and Yonatan Wexler. Detecting text in natural scenes with stroke width transform. In *2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2963–2970. IEEE, 2010.

[6] 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.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity Mappings in Deep Residual Networks. In *European Conference on Computer Vision*, pages 630–645. Springer, 2016.

[9] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[10] Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Reading Text in the Wild with Convolutional Neural Networks. *CoRR*, abs/1412.1842, 2014.

[11] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Deep Features for Text Spotting. In *European Conference on Computer Vision*, pages 512–528. Springer, 2014.

[12] Yann LeCun and Yoshua Bengio. Convolutional Networks for Images, Speech, and Time-Series. In *The Handbook of Brain Theory and Neural Networks*. MIT Press, 1995.

[13] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[14] Yann LeCun, Koray Kavukcuoglu, and Clément Farabet. Convolutional networks and applications in vision. In *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, pages 253–256. IEEE, 2010.

[15] Anand Mishra, Karteek Alahari, and CV Jawahar. Scene text recognition using higher order language priors. In *BMVC 2012-23rd British Machine Vision Conference*. BMVA, 2012.

[16] David L Poole and Alan K Mackworth. *Artificial Intelligence: foundations of computational agents*. Cambridge University Press, 2010.

[17] Rasmus Rothe, Matthieu Guillaumin, and Luc Van Gool. Non-maximum suppression for object detection by passing messages between windows. In *Asian Conference on Computer Vision*, pages 290–306. Springer, 2014.

[18] Stuart J Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2 edition, 2003.

[19] Kai Wang, Boris Babenko, and Serge Belongie. End-to-end scene text recognition. In *2011 International Conference on Computer Vision*, pages 1457–1464. IEEE, 2011.

[20] Kai Wang and Serge Belongie. Word spotting in the wild. In *European Conference on Computer Vision*, pages 591–604. Springer, 2010.

[21] Tao Wang, David J Wu, Adam Coates, and Andrew Y Ng. End-to-end text recognition with convolutional neural networks. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 3304–3308. IEEE, 2012.

[22] David J Wu. End-to-End Text Recognition with Convolutional Neural Networks. Undergraduate honors thesis, Stanford University School of Engineering, 2012.