Robust Text Reading in Natural Scene Images

Tao Wang, David Wu
Stanford Computer Science Department
353 Serra Mall, Stanford, CA 94305
twangcat@stanford.edu, dwu4@stanford.edu

Abstract

In this paper, we consider applying multilayer, convolutional neural networks to construct a complete end-to-end text recognition system with performance comparable to the current state-of-the-art. Such a system demonstrates the possibility of using a single, unified architecture for both text detection and recognition with minimal need for elaborate hand-engineering of features.

1. Introduction

Extracting textual information from natural images is a challenging problem with many practical applications. While current state-of-the-art methods achieve nearly perfect performance on Object Character Recognition (OCR) for scanned documents, the more general problem of recognizing text in unconstrained images is not quite so simple. Recognizing text in scene images is more challenging due to the many possible variations in backgrounds, textures, fonts, and lighting. As a result of these variations, many high-performing text detection and character recognition systems combine cleverly hand-engineered features [1, 5] or carefully-designed multi-step pipelines [7, 10].

In this paper, we approach the problem from a different angle by using systems that can learn the underlying features best suited for a particular problem. Like many previous text recognition systems, we postulate a multi-stage approach. We begin by training a text detector that performs a binary classification task: determine whether a given image patch contains text or does not contain text. Then, given a full image, we use a sliding window approach to compute bounding boxes for regions of text. Now, in the second step, we train a character classifier that achieves high accuracy on identifying a character within a given patch. Here, we use a 62-way classifier, with one class for each alphanumeric character (both uppercase and lowercase). Given the bounding boxes from the text detector, we again use a sliding window approach with the character classifier to identify the word in the bounding box. Unlike many previous works on end-to-end systems that have seperate learning architectures for the text detection stage and character classification stage, our system uses essentially the same learning architecture for both tasks.

2. Problem Statement

In training the text detector, we have opted to define a positive example as an example in which a single character appears centered in the 32-by-32 window, and a negative otherwise. In particular, windows containing partially occluded characters, part of a character, or off-centered characters are considered negatives. This particular definition will help with computing tight bounding boxes for regions of text, which will be conducive for the character recognizer. Given this criterion, we have assembled a dataset consisting of examples taken from the ICDAR 2003 training images [9], the English subset of the Chars74k dataset [5], and the sign-reading dataset from Weinman, *et al*. [12]. We further augment our dataset using synthetically generated examples. To evaluate the performance of the complete end-to-end system, we intend to example word-level precisions and recalls on the ICDAR dataset [9]. We are also considering using the harder, but perhaps, more relevant Street View Dataset [7]. We also evaluate each of the individual components of the system. For text recognition, we evaluate the precision and recall of the computed bounding boxes of the full ICDAR dataset. For the character classifier, we can measure accuracies on individual characters, as well as overall accuracies at the word level.

3. Learning Architecture

We now describe our architecture used to learn the feature representations and train the classifiers used for both the detection and recognition systems. We consider a multilayer convolutional neural architecture similar to [2, 8, 11] for both components of the system. In particular, we adopt a two-layer convolutional structure, with an average-pooling layer following each convolutional layer. The responses from the second pooling layer are then combined in a fully

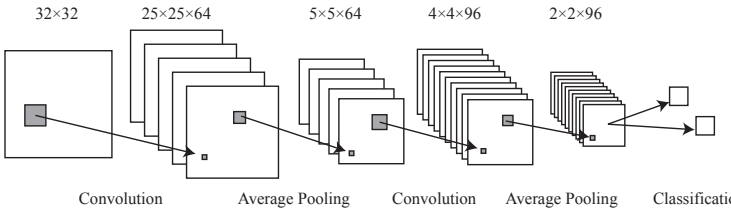

Figure 1. Convolution neural network used for detection and recognition. The only difference between the CNN used for detection and recognition is the sizes of the convolutional layers.

connected classification layer, where we have one output unit for each class (binary in the case of text detection, 62-way in the case of character recognition). We pretrain the first convolutional layer with filters generated by a feature learning algorithm and fine-tune the overall network by backpropagation of the classification error. Finally, we can integrate the text detector and character recognizer to construct a complete end-to-end text recognition system.

3.1. Unsupervised pretraining

We begin by using an unsupervised learning algorithm to pretrain the filters used for both detection and recognition. Here, we use a pipeline that resembles the architectures described in [3, 4]. We briefly outline the key components of this system:

1. Collect a set of m small image patches from the training set. As in [3], we use 8x8 grayscale patches. This yields a set of m vectors of pixels $\tilde{x}^{(i)} \in \mathbb{R}^{64}, i \in \{1, \ldots, m\}$.

2. Normalize each vector $\tilde{x}^{(i)}$ for brightness and contrast (subtract out the mean and divide by the standard deviation). We then whiten the patches $\tilde{x}^{(i)}$ using ZCA whitening [6] to yield a new set of vectors $x^{(i)}$.

3. Apply an unsupervised learning algorithm on the preprocessed patches $x^{(i)}$ to build a mapping from input patches to feature vectors $z^{(i)} = f(x^{(i)})$. In this paper, we adhere to the variant of the K-means algorithm described in [3] where we learn a dictionary $D \in \mathbb{R}^{64 \times D}$ of normalized basis vectors.

3.2. Convolutional layers

Our two-layer convolutional architecture is shown in Figure 1. For the first convolutional layer, we use 8-by-8 filters. In the case of detection, we have 64 filters. We evaluate these filters convolutionally over the entire image, yielding a 25-by-25-by-64 response map. Like [3], we now apply a scalar nonlinearity function to these responses: $z = \max\{0, |x| - \alpha\}$ where x here denotes an element in this response map and α is a hyperparameter to be chosen. In this paper, we take $\alpha = 0.5$. As is standard in the literature on convolutional architectures, we now apply a spatial pooling step. This has the benefit of reducing the dimensionality of the response maps, and thus, renders the network easier to train. Here, we opt for average pooling, in which we sum over the values in a 5-by-5 grid over the 25-by-25-by-64 response map. Average pooling has the added benefit that it allows our model a degree of translational invariance; since we are summing over distinct blocks of the image, a slight translation will not lead to a completely different response. After the first average-pooling step, we introduce one additional convolutional and average pooling layer on top. The outputs of the second pooling layer are then fully connected to a classification layer. In the case of detection, we have two nodes (for binary classification) and for recognition, we have 62 nodes (one for each class). Note that we perform the backpropagation of the classification error on the GPU to speed up training and validation time.

3.3. Text detector and character classifier training

Since we are using centered characters as the definition of text, we have compiled a dataset consisting of examples from the ICDAR 2003 training images [9], the English subset of the Chars74k dataset [5], and the sign-reading dataset from Weinman, *et al*. [12], as well as as synthetically generated examples shown in Figure 2. Combined, our dataset consists of 75,000 positive examples and 150,000 negative examples. For the character classifier training, we use the same dataset; however, we only use the positive character examples in this case. We have considered a 63-class system where we introduce a class for non-characters such as spaces, background, and so forth. However, this does not work very well with word level recognition and we have opted back to a 63-class system.

Figure 2. Left: Images from the ICDAR-Character dataset. Right: Synthetic data we generate

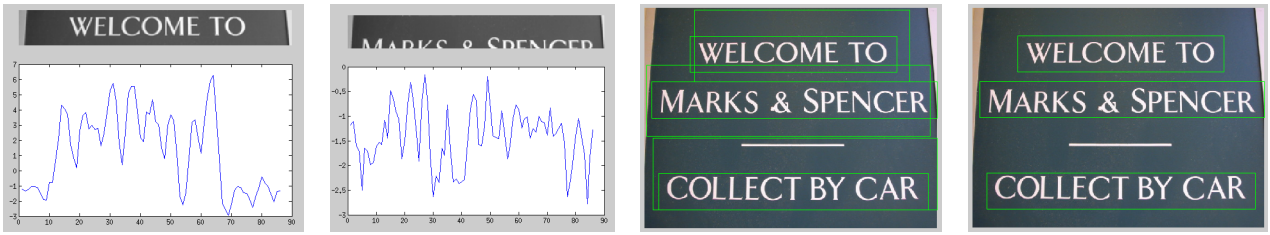

Figure 3. From left to right, response map for a centered line of text, response map for a partial line of text, bounding boxes before non-maximum suppression, bounding boxes after non-maximum suppression.

4. Experiments

4.1. Text detection

Given an input image with dimensions, we begin by identifying regions of text using a sliding window approach. At each position in the image, the detector assigns it a score denoting the likelihood that the given window contains text. Here, a positive response suggests that the given window contains text and a negative response suggests the contrary. To allow for different-sized text, we evaluate the detector responses over different scales. After this process is complete, we have a response map that contains a score at each location over different scales. Now, noting that text tends to be aligned horizontally, we go through each horizontal line in the response map and compute the window with maximum score. We show two lines and their respective responses in Figure 3. As we can see from Figure 3, if there is a word centered on a line, then the sum of the responses for the window containing the given word will have a high positive score. In lines that do not contain text, the scores will generally be negative. By thresholding this score, we can restrict our focus to just certain lines in the image. To go from a horizontal window spanning the width of the image to a bounding box for a word, we examine the detector responses across the line, and apply non-maximal suppression (NMS). Using the locations of the peaks (generally corresponding to a centered character), we extract a word-level bounding box by taking the leftmost peak and the rightmost peak. We now rescore each bounding box by averaging the detector response over the length of the bounding box. At this point, we have a set of candidate bounding boxes computed over different scales. To further filter this set of boxes, we again apply non-maximal suppression to the candidate bounding boxes. Results from this NMS are given in the last pair of images in Figure 3. In this particular case, we see that the pipeline has properly identified the regions of text in the image.

In terms of quantitative results, if we score the bounding boxes on the ICDAR test set using the ICDAR criterion [9], we obtain an F_1 measure of **0.39**, which is significantly worse than current state-of-the-art systems which obtain F_1 measures between 0.66 and 0.69 [1, 10] . However, we note that this metric evaluates *word*-level bounding boxes while our detector tends to output *line*-level bounding boxes. The next step here will be to perform word-level segmentation, which should lead to an improvement in word-level precisions and recalls.

4.2. Character Classification

The character classification accuracy is mainly evaluated on the ICDAR-Character dataset, which consists of centered characters that are cropped out from the ICDAR robust reading dataset. We are still in the process of running finetuning on a large dataset and do not have concrete numbers yet.

References

[1] Y. W. B. Epshtein, E. Oyek. Detecting text in natural scenes with stroke width transform. 2010. 1, 3

[2] D. C. Ciresan, U. Meier, J. Masci, L. M. Gambardella, and J. Schmidhuber. High performance neural networks for vi-

sual object classification. Technical Report IDSIA-01-11, Dalle Molle Institute for Artificial Intelligence, 2011. 1

[3] A. Coates, B. Carpenter, C. Case, S. Satheesh, B. Suresh, T. Wang, D. J. Wu, and A. Y. Ng. Text detection and character recognition in scene images with unsupervised feature learning. In *ICDAR*, 2011. 2

[4] A. Coates, H. Lee, and A. Y. Ng. An analysis of single-layer networks in unsupervised feature learning. In *International Conference on Artificial Intelligence and Statistics*, 2011. 2

[5] 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. 1, 2

[6] A. Hyvarinen and E. Oja. Independent component analysis: algorithms and applications. *Neural networks*, 13(4-5):411–430, 2000. 2

[7] S. B. K. Wang, B. Babenko. End-to-end scene text recognition. 2011. 1

[8] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1:541–551, 1989. 1

[9] S. Lucas, A. Panaretos, L. Sosa, A. Tang, S. Wong, and R. Young. ICDAR 2003 robust reading competitions. *International Conference on Document Analysis and Recognition*, 2003. 1, 2, 3

[10] Y. Pan, X. Hou, and C. Liu. Text localization in natural scene images based on conditional random field. In *International Conference on Document Analysis and Recognition*, 2009. 1, 3

[11] Z. Saidane and C. Garcia. Automatic scene text recognition using a convolutional neural network. In *Workshop on Camera-Based Document Analysis and Recognition*, 2007. 1

[12] J. Weinman, E. Learned-Miller, and A. R. Hanson. Scene text recognition using similarity and a lexicon with sparse belief propagation. In *Transactions on Pattern Analysis and Machine Intelligence*, volume 31, 2009. 1, 2