End-to-End Text Recognition with Convolutional Neural Networks

Tao Wang* David J. Wu* Adam Coates Andrew Y. Ng
Stanford University, 353 Serra Mall, Stanford, CA 94305
{*twangcat, dwu4, acoates, ang*}*@cs.stanford.edu*

Abstract

Full end-to-end text recognition in natural images is a challenging problem that has received much attention recently. Traditional systems in this area have relied on elaborate models incorporating carefully hand-engineered features or large amounts of prior knowledge. In this paper, we take a different route and combine the representational power of large, multilayer neural networks together with recent developments in unsupervised feature learning, which allows us to use a common framework to train highly-accurate text detector and character recognizer modules. Then, using only simple off-the-shelf methods, we integrate these two modules into a full end-to-end, lexicon-driven, scene text recognition system that achieves state-of-the-art performance on standard benchmarks, namely Street View Text and ICDAR 2003.

1 Introduction

Extracting textual information from natural images is a challenging problem with many practical applications. Unlike character recognition for scanned documents, recognizing text in unconstrained images is complicated by a wide range of variations in backgrounds, textures, fonts, and lighting conditions. As a result, many text detection and recognition systems rely on cleverly hand-engineered features [5, 4, 14] to represent the underlying data. Sophisticated models such as conditional random fields [11, 19] or pictorial structures [18] are also often required to combine the raw detection/recognition outputs into a complete system.

In this paper, we attack the problem from a different angle. For low-level data representation, we use an unsupervised feature learning algorithm that can automatically extract features from the given data. Such algorithms have enjoyed numerous successes in many related fields such as visual recognition [3] and action recognition [7]. In the case of text recognition, the system in [2] achieves competitive results in both text detection and character recognition using a simple and scalable feature learning architecture incorporating very little hand-engineering and prior knowledge.

*T. Wang and D. Wu contributed equally to this work.

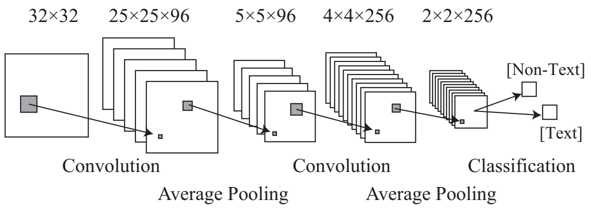

Figure 1. CNN used for text detection.

We integrate these learned features into a large, discriminatively-trained convolutional neural network (CNN). CNNs have enjoyed many successes in similar problems such as handwriting recognition [8], visual object recognition [1], and character recognition [16]. By leveraging the representational power of these networks, we are able to train highly accurate text detection and character recognition modules. Using these modules, we can build an end-to-end system with only simple post-processing techniques like non-maximal suppression (NMS)[13] and beam search [15]. Despite its simplicity, our system achieves state-of-the-art performance on standard test sets.

2 Learning Architecture

In this section, we describe our text detector and character recognizer modules, which are the essential building blocks of our full end-to-end system. Given a 32-by-32 pixel window, the detector decides whether the window contains a centered character. Similarly, the recognizer decides which of 62 characters (26 uppercase, 26 lowercase letters, and 10 digits) is in the window. As described at length in Section 3, we slide the

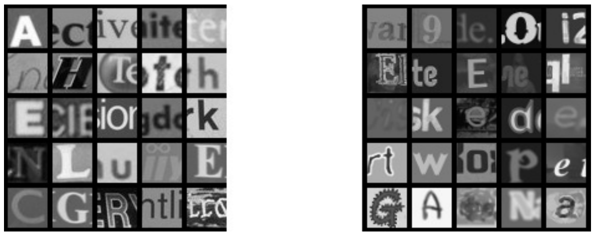
Figure 2. Examples from our training set. Left: from ICDAR. Right: synthetic data

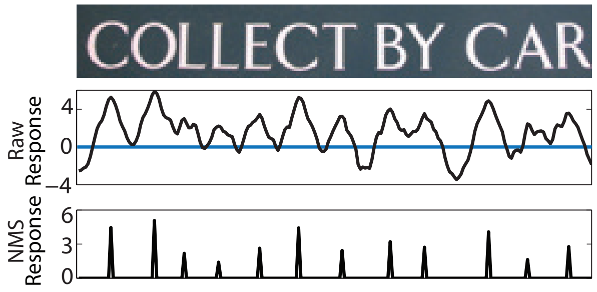

Figure 3. Detector responses in a line.

detector across a full scene image to identify candidate lines of text, on which we perform word-level segmentation and recognition to obtain the end-to-end results.

For both detection and recognition, we use a multi-layer, convolutional neural network (CNN) similar to [8, 16]. Our networks have two convolutional layers with n_1 and n_2 filters respectively. The network we use for detection with $n_1 = 96$ and $n_2 = 256$ is shown in Figure 1, while a larger, but structurally identical one ($n_1 = 115$ and $n_2 = 720$) is used for recognition.

We train the first layer of the network with an unsupervised learning algorithm similar to [2, 3]. In particular, given a set of 32-by-32 grayscale training images[1] as illustrated in Figure 2, we randomly extract m 8-by-8 patches, which are contrast normalized and ZCA whitened [6] to form input vectors $x^{(i)} \in \mathbb{R}^{64}, i \in \{1, ..., m\}$. We then use the variant of K-means described in [2] to learn a set of low-level filters $D \in \mathbb{R}^{64\times n_1}$. For a single normalized and whitened 8-by-8 patch x, we compute its first layer responses z by performing inner product with the filter bank followed by a scalar activation function: $z = \max\{0, |D^\top x| - \alpha\}$, where $\alpha = 0.5$ is a hyperparameter.

Given a 32-by-32 input image, we compute z for every 8-by-8 sub-window to obtain a 25-by-25-by-n_1 first layer response map. As is common in CNNs, we average pool over the first layer response map to bring its dimensions to 5-by-5-by-n_1. We stack another convolution and average pooling layer on top of the first layer to obtain a 2-by-2-by-n_2 second layer response map. These outputs are fully connected to the classification layer. We discriminatively train the network by back-propagating the L_2-SVM classification error,[2] but we fix the filters in the first convolution layer (learned from K-means). Given the size of the networks, fine-tuning is performed using multiple GPUs.

[1]Our dataset consists of examples from the ICDAR 2003 training images [10], the English subset of the Chars74k dataset [4], and synthetically generated examples.

[2]In the form of a squared hinge loss: $\max\{0, 1 - \theta^\top x\}^2$.

3 End-to-End Pipeline Integration

Our full end-to-end system combines a *lexicon* with our detection/recognition modules using post-processing techniques including NMS and beam search. Here we assume that we are given a lexicon (a list of tens to hundreds of candidate words) for a particular image. As argued in [18], this is often a valid assumption as we can use prior knowledge to constrain the search to just certain words in many applications. The pipeline mainly involves the following two stages:

(i) We run sliding window detection over high resolution input images to obtain a set of candidate lines of text. Using these detector responses, we also estimate locations for the spaces in the line.

(ii) We integrate the character responses with the candidate spacings using beam search [15] to obtain full end-to-end results.

First, given an input image, we identify horizontal lines of text using multiscale, sliding window detection. At each scale s, we evaluate the detector response $R_s[x, y]$ at each point (x, y) in the scaled image. As shown in Figure 3, windows centered on single characters at the right scale produce positive $R_s[x, y]$. We apply NMS [13] to $R_s[x, r]$ in each individual row r to estimate the character locations on a horizontal line. In particular, we define the NMS response

$$\tilde{R}_s[x, r] = \begin{cases} R_s[x, r] & \text{if } R_s[x, r] \geq R_s[x', r], \\ & \qquad \forall x' \ s.t.\ |x' - x| < \delta \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where δ is some width parameter. For a row r with non-zero $\tilde{R}_s[x, r]$, we form a line-level bounding box L_s^r with the same height as the sliding window at scale s. The left and right boundaries of L_s^r are defined as $\min(x)$ and $\max(x)$, s.t. $\tilde{R}_s[x, r] > 0$. This yields a set of possibly overlapping line-level bounding boxes. We score each box by averaging the nonzero values of $\tilde{R}_s[x, r]$. We then apply standard NMS to remove all

L's that overlaps by more than 50% with another box of a higher score, and obtain the final set of line-level bounding boxes $\tilde{\mathbf{L}}$. Since gaps between words produce sharply negative responses, we also estimate possible space locations within each L_s^r by applying the same NMS technique as above to the negative responses.

After identifying the horizontal lines of text, we jointly segment the lines of text into words and recognize each word in the line. Given a line-level bounding box L and its candidate space locations, we evaluate a number of possible word-level bounding boxes using a Viterbi-style algorithm and find the best segmentation scheme using a beam search technique similar to [9]. To evaluate a word-level bounding box B, we slide the character recognizer across it and obtain a $62 \times N$ score matrix M, where N is the number of sliding windows within the bounding box. Intuitively, a more positive $M(i, j)$ suggests a higher chance that the character with index i is centered on the location of the j^{th} window. Similar to the detection phase, we perform NMS over M to select the columns where a character is most likely to be present. The other columns of M are set to $-\infty$. We then find the lexicon word w^* that best matches a score matrix M as follows: given a lexicon word w, compute the alignment score

$$S_M^w = \max_{l^w \in L^w} \left(\sum_k^{|w|} M(w_k, l_k^w) \right) \qquad (2)$$

where l^w is the alignment vector[3] between the characters in w and the columns of M. S_M^w can be computed efficiently using a Viterbi-style alignment algorithm similar to [17].[4] We compute S_M^w for all lexicon words and label the word-level bounding-box B with the highest scoring word w^*. We take $S_B = S_M^{w^*}$ to be the *recognition score* of B.

Having defined the recognition score for a single bounding box, we can now systematically evaluate possible word-level segmentations using beam search [15], a variant of breadth first search that explores the top N possible partial segmentations according to some heuristic score. In our case, the heuristic score of a candidate segmentation is the sum of the S_B's over all the resulting bounding boxes in a line of text L. In order to deal with possible false positives from the text detection stage, we threshold individual segments based on their recognition scores. In that way, segments with low recognition scores are pruned out as being "non-text."

[3]For example, $l_4^w = 6$ means the 4th character in w aligns with the 6th column of M, or the 6th sliding window in a line of text.

[4]In practice, we also augment S_M^w with additional terms that encourage geometric consistency. For example, we penalize character spacings that are either too narrow or vary a lot within a single word.

Table 1. Cropped word recognition accuracies on ICDAR 2003 and SVT

Benchmark	I-WD-50	I-WD	SVT-WD
Our approach	**90%**	**84%**	70%
Wang, *et al.* [18]	76%	62%	57%
Mishra, *et al.* [11]	82%	-	**73%**

4 Experimental Results

In this section we present a detailed evaluation of our text recognition pipeline. We measure cropped character and word recognition accuracies, as well as end-to-end text recognition performance of our system on the ICDAR 2003 [10] and the Street View Text (SVT) [18] datasets. Apart from that, we also perform additional analysis to evaluate the importance of model size on different stages of the pipeline.

First we evaluate our character recognizer module on the ICDAR 2003 dataset. Our 62-way character classifier achieves state-of-the-art accuracy of 83.9% on cropped characters from the ICDAR 2003 test set. The best known previous result on the same benchmark is 81.7% reported by [2]

Our word recognition sub-system is evaluated on images of perfectly cropped words from the ICDAR 2003 and SVT datasets. We use the exact same test setup as [18]. More concretely, we measure word-level accuracy with a lexicon containing all the words from the ICDAR test set (called I-WD), and with lexicons consisting of the ground truth words for that image plus 50 random "distractor" words added from the test set (called I-WD-50). For the SVT dataset, we used the provided lexicons to evaluate the accuracy (called SVT-WD). Table 1 compares our results with [18] and the very recent work of [11].

We evaluate our final end-to-end system on both the ICDAR 2003 and SVT datasets, where we locate and recognize words in full scene images given a lexicon. For the SVT dataset, we use the provided lexicons; for the ICDAR 2003 dataset, we used lexicons of 5, 20 and 50 distractor words provided by the authors of [18], as well as the "FULL" lexicon consisting of all words in the test set. We call these benchmarks I-5, I-20, I-50 and I-FULL respectively. Like [18], we only consider alphanumeric words with at least 3 characters. Figure 5 shows some sample outputs of our system. We follow the standard evaluation criterion described in [10] to compute the precision and recall. Figure 4 shows precision and recall plots for the different benchmarks on the ICDAR 2003 dataset.

As a standard way of summarizing results, we also

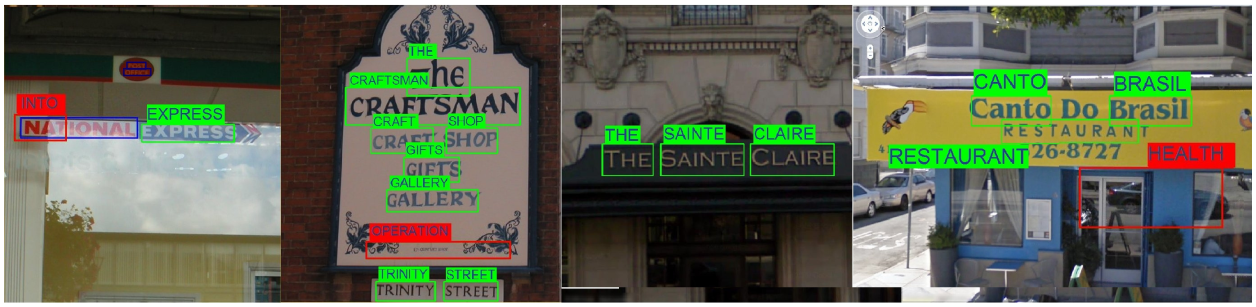

Figure 5. Example output bounding boxes of our end-to-end system on I-FULL and SVT benchmarks. Green: correct detections. Red: false positives. Blue: misses.

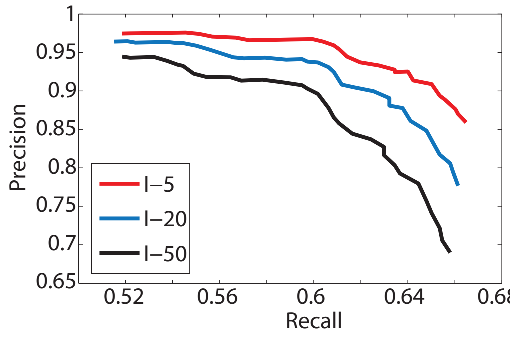

Figure 4. End-to-end PR curves on ICDAR 2003 dataset using lexicons with 5, 20, and 50 distractor words.

report the highest F-scores over the PR curves and compare with [18] in Table 2. Our system achieves higher F-scores in every case. Moreover, the margin of improvement is much higher on the harder benchmarks (0.16 for I-FULL and 0.08 for SVT), suggesting that our system is robust in more general settings.

In addition to settings with a known lexicon, we also extend our system to the more general setting by using a large lexicon $\mathbb{L}$ of common words. Since it is infeasible to search over all words in this case, we limit our search to a small subset $P \in \mathbb{L}$ of "visually plausible" words. We first perform NMS on the score matrix M across positions and character classes, and then threshold it with different values to obtain a set of raw strings. The raw strings are fed into Hunspell[5] to yield a set of suggested words as our smaller lexicon P, Using this simple setup, we achieve scores of 0.54/0.30/0.38 (precision/recall/F-score) on the ICDAR dataset. This

[5]Hunspell is an open source spell checking software available at http://hunspell.sourceforge.net/. We augment its default lexicon with a corpus of English proper names to better handle text in scenes.

Table 2. F-scores from end-to-end evaluation on ICDAR 2003 and SVT datasets.

Benchmark	I-5	I-20	I-50	I-FULL	SVT
Our approach	**.76**	**.74**	**.72**	**.67**	**.46**
Wang, *et al.* [18]	.72	.70	.68	.51	.38

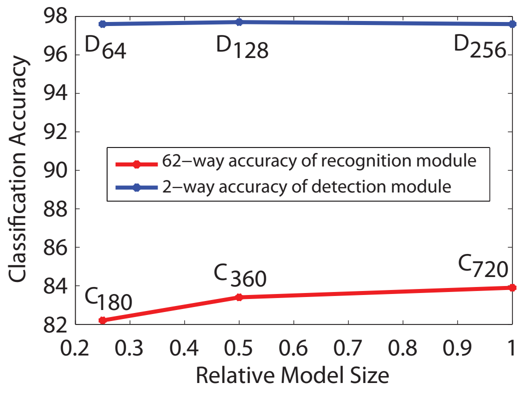

Figure 6. Accuracies of the detection and recognition modules on cropped patches

is comparable to the best known result 0.42/0.39/0.40 obtained with a general lexicon by [14].

In order to analyze the impact of model size on different stages of the pipeline, we also train detection and recognition modules with fewer second layer convolutional filters. The detection modules have $n_2 = 64$ and 128 compared to 256 in our full model. We call the detection modules D_{64}, D_{128} and D_{256} respectively. Similarly, we call the recognition modules C_{180}, C_{360} and C_{720}, which corresponds to $n_2 = 180$, 360 and 720. The smaller models have about $^1/_4$ and $^1/_2$ number of learnable parameters compared to the full models.

To evaluate the performance of the detection mod-

Table 3. Classification and end-to-end results of different recognition modules

Recognition module	C_{180}	C_{360}	C_{720}
Classification accuracy	82.2%	83.4%	83.9%
End-to-end F-score	.6330	.6333	.6723

ules, we construct a 2-way (character vs. non-character) classification dataset by cropping patches from the ICDAR test images. The recognition modules are evaluated on cropped characters only. As shown in Figure 6, the 62-way classification accuracy increases as model size gets larger, while the 2-way classification results remain unchanged. This suggests that larger model sizes yield better recognition modules, but not necessarily better detection modules.

Finally, we evaluate the the 3 different recognition modules on the I-FULL benchmark, with D_{256} as the detector for all 3 cases. The end-to-end F-scores are listed against the respective classification accuracies in Table 3. The results suggests that higher character classification accuracy does give rise to better end-to-end results. This trend is consistent with the findings of [12] on house number recognition in natural images.

5 Conclusion

In this paper, we have considered a novel approach for end-to-end text recognition. By leveraging large, multi-layer CNNs, we train powerful and robust text detection and recognition modules. Because of this increase in representational power, we are able to use simple non-maximal suppression and beam search techniques to construct a complete system. This represents a departure from previous systems which have generally relied on intricate graphical models or elaborately hand-engineered systems. As evidence of the power of this approach, we have demonstrated state-of-the-art results in character recognition as well as lexicon-driven cropped word recognition and end-to-end recognition. Even more, we can easily extend our model to the general-purpose setting by leveraging conventional open-source spell checkers and in doing so, achieve performance comparable to state-of-the-art.

References

[1] D. C. Ciresan, U. Meier, J. Masci, L. M. Gambardella, and J. Schmidhuber. High performance neural networks for visual object classification. Technical Report IDSIA-01-11, Dalle Molle Institute for Artificial Intelligence, 2011.

[2] 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.

[3] A. Coates, H. Lee, and A. Y. Ng. An analysis of single-layer networks in unsupervised feature learning. In *AISTATS*, 2011.

[4] T. E. de Campos, B. R. Babu, and M. Varma. Character recognition in natural images. In *VISAPP*, 2009.

[5] B. Epshtein, E. Oyek, and Y. Wexler. Detecting text in natural scenes with stroke width transform. In *CVPR*, 2010.

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

[7] Q. V. Le, W. Y. Zou, S. Y. Yeung, and A. Y. Ng. Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. In *CVPR*, 2011.

[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.

[9] C.-L. Liu, M. Koga, and H. Fujisawa. Lexicon-driven segmentation and recognition of handwritten character strings for japanese address reading. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(11):1425–1437, Nov. 2002.

[10] S. Lucas, A. Panaretos, L. Sosa, A. Tang, S. Wong, and R. Young. ICDAR 2003 robust reading competitions. *ICDAR*, 2003.

[11] A. Mishra, K. Alahari, and C. V. Jawahar. Top-down and bottom-up cues for scene text recognition. In *CVPR*, 2012.

[12] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

[13] A. Neubeck and L. Gool. Efficient non-maximum suppression. In *ICPR*, 2006.

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

[15] S. J. Russell, P. Norvig, J. F. Candy, J. M. Malik, and D. D. Edwards. *Artificial intelligence: a modern approach*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1996.

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

[17] S. Sarawagi and W. W. Cohen. Semi-markov conditional random fields for information extraction. In *NIPS*, pages 1185–1192, 2004.

[18] K. Wang, B. Babenko, and S. Belongie. End-to-end scene text recognition. In *ICCV*, 2011.

[19] J. J. Weinman, E. Learned-Miller, and A. R. Hanson. A discriminative semi-markov model for robust scene text recognition. In *ICPR*, Dec. 2008.