Deep Features for Text Spotting

Max Jaderberg, Andrea Vedaldi, Andrew Zisserman

Visual Geometry Group, Department of Engineering Science, University of Oxford

Abstract. The goal of this work is text spotting in natural images. This is divided into two sequential tasks: detecting words regions in the image, and recognizing the words within these regions. We make the following contributions: first, we develop a Convolutional Neural Network (CNN) classifier that can be used for both tasks. The CNN has a novel architecture that enables efficient feature sharing (by using a number of layers in common) for text detection, character case-sensitive and insensitive classification, and bigram classification. It exceeds the state-of-the-art performance for all of these. Second, we make a number of technical changes over the traditional CNN architectures, including no downsampling for a per-pixel sliding window, and multi-mode learning with a mixture of linear models (maxout). Third, we have a method of automated data mining of Flickr, that generates word and character level annotations. Finally, these components are used together to form an end-to-end, state-of-the-art text spotting system. We evaluate the text-spotting system on two standard benchmarks, the ICDAR Robust Reading data set and the Street View Text data set, and demonstrate improvements over the state-of-the-art on multiple measures.

1 Introduction

While text recognition from scanned documents is well studied and there are many available systems, the automatic detection and recognition of text within images – text spotting (Fig.1) – is far less developed. However, text contained within images can be of great semantic value, and so is an important step towards both information retrieval and autonomous systems. For example, text spotting of numbers in street view data allows the automatic localization of houses numbers in maps [20], reading street and shop signs gives robotic vehicles scene context [39], and indexing large volumes of video data with text obtained by text spotting enables fast and accurate retrieval of video data from a text search [26].

Text spotting in natural images is usually divided into two tasks [12]: text detection, and word recognition. Text detection involves generating candidate bounding boxes that are likely to contain lines of text, while word recognition takes each candidate bounding box, and attempts to recognize the text depicted within it, or potentially reject the bounding box as a false positive detection.

In this paper we show that a very high quality character classifier can improve over the state-of-the-art for both the word detection and recognition tasks of this

(a) (b)

Fig. 1. (a) An end-to-end text spotting result from the presented system on the SVT dataset. (b) Randomly sampled cropped word data automatically mined from Flickr with a weak baseline system, generating extra training data.

pipeline. To achieve this we use a Convolutional Neural Network (CNN) [27] and generate a per-pixel text/no-text saliency map, a case-sensitive and case-insensitive character saliency map, and a bigram saliency map. The text saliency map drives the proposal of word bounding boxes, while the character and bigram saliency maps assist in recognizing the word within each bounding box through a combination of soft costs. Our work is inspired by the excellent performance of CNNs for character classification [6, 8, 47]. Our contributions are threefold:

First, we introduce a method to share features [44] which allows us to extend our character classifiers to other tasks such as character detection and bigram classification at a very small extra cost: we first generate a single rich feature set, by training a strongly supervised character classifier, and then use the intermediate hidden layers as features for the text detection, character case-sensitive and insensitive classification, and bigram classification. This procedure makes best use of the available training data: plentiful for character/non-character but less so for the other tasks. It is reminiscent of the Caffe idea [14], but here it is not necessary to have external sources of training data.

A second key novelty in the context of text detection is to leverage the convolutional structure of the CNN to process the entire image in one go instead of running CNN classifiers on each cropped character proposal [27] . This allows us to generate efficiently, in a single pass, all the features required to detect word bounding boxes, and that we use for recognizing words from a fixed lexicon using the Viterbi algorithm. We also make a technical contribution in showing that our CNN architecture using maxout [21] as the non-linear activation function has superior performance to the more standard rectified linear unit.

Our third contribution is a method for automatically mining and annotating data (Fig.1). Since CNNs can have many millions of trainable parameters, we require a large corpus of training data to minimize overfitting, and mining is useful to cheaply extend available data. Our mining method crawls images from the Internet to automatically generate word level and character level bounding box annotations, and a separate method is used to automatically generate character level bounding box annotations when only word level bounding box annotations are supplied.

In the following we first describe the data mining procedure (Sect. 2) and then the CNN architecture and training (Sect. 3). Our end-to-end (image in, text out) text spotting pipeline is described in Sect. 4. Finally, Sect. 5 evaluates the method on a number of standard benchmarks. We show that the performance exceeds the state of the art across multiple measures.

Related Work. Decomposing the text-spotting problem into text detection and text recognition was first proposed by [12]. Authors have subsequently focused solely on text detection [7, 11, 16, 50, 51], or text recognition [31, 36, 41], or on combining both in end-to-end systems [40, 39, 49, 32–34, 45, 35, 6, 8, 48].

Text detection methods are either based on connected components (CCs) [11, 16, 50, 49, 32–35] or sliding windows [40, 7, 39, 45]. *Connected component methods* segment pixels into characters, then group these into words. For example, Epshtein *et al.* take characters as CCs of the stroke width transform [16], while Neumann and Matas [34, 33] use Extremal Regions [29], or more recently oriented strokes [35], as CCs representing characters. *Sliding window methods* approach text spotting as a standard task of object detection. For example, Wang *et al.* [45] use a random ferns [38] sliding window classifier to find characters in an image, grouping them using a pictorial structures model [18] for a fixed lexicon. Wang & Wu *et al.* [47] build on the fixed lexicon problem by using CNNs [27] with unsupervised pre-training as in [13]. Alsharif *et al.* [6] and Bissacco *et al.* [8], also use CNNs for character classification – both methods over-segment a word bounding box and find an approximate solution to the optimal word recognition result, in [8] using beam search and in [6] using a Hidden Markov Model.

The works by Mishra *et al.* [31] and Novikova *et al.* [36] focus purely on text recognition – assuming a perfect text detector has produced cropped images of words. In [36], Novikova combines both visual and lexicon consistency into a single probabilistic model.

2 Data mining for word and character annotations

In this section we describe a method for automatically mining suitable photo sharing websites to acquire word and character level annotated data. This annotation is used to provide additional training data for the CNN in Sect. 5.

Word Mining. Photo sharing websites such as Flickr [3] contain a large range of scenes, including those containing text. In particular, the "Typography and Lettering" group on Flickr [4] contains mainly photos or graphics containing text. As the text depicted in the scenes are the focus of the images, the user given titles of the images often include the text in the scene. Capitalizing on this weakly supervised information, we develop a system to find title text within the image, automatically generating word and character level bounding box annotations.

Using a weak baseline text-spotting system based on the Stroke Width Transform (SWT) [16] and described in Sect. 5, we generate candidate word detections

for each image from Flickr. If a detected word is the same as any of the image's title text words, and there are the same number of characters from the SWT detection phase as word characters, we say that this is an accurate word detection, and use this detection as positive text training data. We set the parameters so that the recall of this process is very low (out of 130000 images, only 15000 words were found), but the precision is greater than 99%. This means the precision is high enough for the mined Flickr data to be used as positive training data, but the recall is too low for it to be used for background no-text training data. We will refer to this dataset as *FlickrType*, which contains 6792 images, 14920 words, and 71579 characters. Fig. 1 shows some positive cropped words randomly sampled from the automatically generated FlickrType dataset.

Although this procedure will cause a bias towards scene text that can be found with a simple end-to-end pipeline, it still generates more training examples that can be used to prevent the overfitting of our models.

Automatic Character Annotation. In addition to mining data from Flickr, we also use the word recognition system described in Sect. 4.2 to automatically generate *character* bounding box annotations for datasets which only have *word* level bounding box annotations. For each cropped word, we perform the optimal fitting of the groundtruth text to the character map using the method described in Sect. 4.2. This places inter-character breakpoints with implied character centers, which can be used as rough character bounding boxes. We do this for the SVT and Oxford Cornmarket datasets (that are described in section 5), allowing us to train and test on an extra 22,000 cropped characters from those datasets.

3 Feature learning using a Convolutional Neural Network

The workhorse of a text-spotting system is the character classifier. The output of this classifier is used to recognize words and, in our system, to detect image regions that contain text. Text-spotting systems appear to be particularly sensitive to the performance of character classification; for example, in [8] increasing the accuracy of the character classifier by 7% led to a 25% increase in word recognition. In this section we therefore concentrate on maximizing the performance of this component.

To classify an image patch x in one of the possible characters (or background), we extract a set of features $\Phi(x) = (\phi_1(x), \phi_2(x), ..., \phi_K(x))$ and then learn a binary classifier f_c for each character c of the alphabet C. Classifiers are learned to yield a posterior probability distribution $p(c|x) = f_c(\Phi(x))$ over characters and the latter is maximized to recognize the character $\bar{c}$ contained in patch x: $\bar{c} = \text{argmax}_{c \in C}\, p(c|x)$. Traditionally, features Φ are manually engineered and optimized through a laborious trial-and-error cycle involving adjusting the features and re-learning the classifiers. In this work, we propose instead to *learn* the representation using a CNN [27], jointly optimizing the performance of the features as well as of the classifiers. As noted in the recent literature, a well designed

learnable representation of this type can in fact yield substantial performance gains [25].

CNNs are obtained by stacking multiple layers of features. A *convolutional layer* consist of K linear filters followed by a non-linear response function. The input to a convolutional layer is a *feature map* $z_i(u,v)$ where $(u,v) \in \Omega_i$ are spatial coordinates and $z_i(u,v) \in \mathbb{R}^C$ contains C scalar features or *channels* $z_i^c(u,v)$. The output is a new feature map z_{i+1} such that $z_{i+1}^k = h_i(W_{ik} * z_i + b_{ik})$, where W_{ik} and b_{ik} denote the k-th filter kernel and bias respectively, and h_i is a non-linear activation function such as the *Rectified Linear Unit* (ReLU) $h_i(z) = \max\{0, z\}$. Convolutional layers can be intertwined with *normalization*, *subsampling*, and *max-pooling layers* which build translation invariance in local neighborhoods. The process starts with $z_1 = x$ and ends by connecting the last feature map to a logistic regressor for classification. All the parameters of the model are jointly optimized to minimize the classification loss over a training set using Stochastic Gradient Descent (SGD), back-propagation, and other improvements discussed in Sect. 3.1.

Instead of using ReLUs as activation function h_i, in our experiments it was found empirically that *maxout* [21] yields superior performance. Maxout, in particular when used in the final classification layer, can be thought of as taking the maximum response over a mixture of n linear models, allowing the CNN to easily model multiple modes of the data. The maxout of two feature channels z_i^1 and z_i^2 is simply their pointwise maximum: $h_i(z_i(u,v)) = \max\{z_i^1(u,v), z_i^2(u,v)\}$. More generally, the k'-th maxout operator $h^{k'}$ is obtained by selecting a subset $G_{k'i} \subset \{1, 2, \ldots, K\}$ of feature channels and computing the maximum over them: $h_i^{k'}(z_i(u,v)) = \max_{k \in G_{k'i}} z_i^k(u,v)$. While different grouping strategies are possible, here groups are formed by taking g consecutive channels of the input map: $G_{1i} = \{1, 2, \ldots, g\}$, $G_{2i} = \{g+1, g+2, \ldots, 2g\}$ and so on. Hence, given K feature channels as input, maxout constructs $K' = K/g$ new channels.

3.1 Training and implementation details

This section discusses the details of learning the character classifiers. Training is divided into two stages. In the first stage, a case-insensitive CNN character classifier is learned. In the second stage, the resulting feature maps are applied to other classification problems as needed. The output is four state-of-the-art CNN classifiers: a character/background classifier, a case-insensitive character classifier, a case-sensitive character classifier, and a bigram classifier.

Stage 1: Bootstrapping the case-insensitive classifier. The case-insensitive classifier uses a four-layer CNN outputting a probability $p(c|x)$ over an alphabet C including all 26 letters, 10 digits, and a noise/background (no-text) class, giving a total of 37 classes (Fig. 2) The input $z_1 = x$ of the CNN are grayscale cropped character images of 24×24 pixels, zero-centered and normalized by subtracting the patch mean and dividing by the standard deviation.

Due to the small input size, no spatial pooling or downsampling is performed. Starting from the first layer, the input image is convolved with 96 filters of size

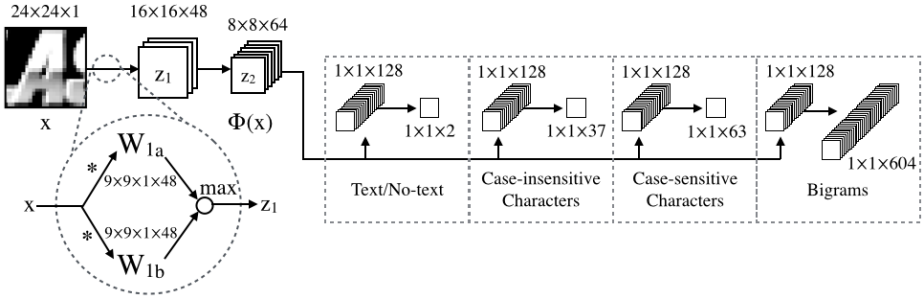


Fig. 2. Convolutional Neural Networks. The method uses four CNNs. These share the first two layers, computing "generic" character features and terminate in layers specialized into text/no-text classification, case-insensitive and case-sensitive character classification, and bigram classification. Each connection between feature maps consists of convolutions with maxout groups.

Fig. 3. Visualizations of each character class learnt from the 37-way case-insensitive character classifier CNN. Each image is synthetically generated by maximizing the posterior probability of a particular class. This is implemented by back-propagating the error from a cost layer that aims to maximize the score of that class [43, 17].

9×9, resulting in a map of size 16×16 (to avoid boundary effects) and 96 channels. The 96 channels are then pooled with maxout in group of size $g = 2$, resulting in 48 channels. The sequence continues by convolving with 128, 512, 148 filters of side 9, 8, 1 and maxout groups of size $g = 2, 4, 4$, resulting in feature maps with 64, 128, 37 channels and size $8 \times 8, 1 \times 1, 1 \times 1$ respectively. The last 37 channels are fed into a soft-max to convert them into character probabilities. In practice we use 48 channels in the final classification layer rather than 37 as the software we use, based on `cuda-convnet` [25], is optimized for multiples of 16 convolutional filters – we do however use the additional 12 classes as extra no-text classes, abstracting this to 37 output classes.

We train using stochastic gradient descent and back-propagation, and also use dropout [22] in all layers except the first convolutional layer to help prevent overfitting. Dropout simply involves randomly zeroing a proportion of the parameters; the proportion we keep for each layer is 1, 0.5, 0.5, 0.5. The training data is augmented by random rotations and noise injection. By omitting any downsampling in our network and ensuring the output for each class is one pixel in size, it is immediate to apply the learnt filters on a full image in a convolutional manner to obtain a per-pixel output without a loss of resolution, as shown

in the second image of Fig 4. Fig. 3 illustrates the learned CNN by using the visualization technique of [43].

Stage 2: Learning the other character classifiers. Training on a large amount of annotated data, and also including a no-text class in our alphabet, means the hidden layers of the network produce feature maps highly adept at discriminating characters, and can be adapted for other classification tasks related to text. We use the outputs of the second convolutional layer as our set of discriminative features, $\Phi(x) = z_2$. From these features, we train a 2-way text/no-text classifier[1], a 63-way case-sensitive character classifier, and a bigram classifier, each one using a two-layer CNN acting on $\Phi(x)$ (Fig. 2). The last two layers of each of these three CNNs result in feature maps with 128-2, 128-63, and 128-604 channels respectively, all resulting from maxout grouping of size $g = 4$. These are all trained with $\Phi(x)$ as input, with dropout of 0.5 on all layers, and fine-tuned by adaptively reducing the learning rate. The bigram classifier recognises instances of two adjacent characters, *e.g.* Fig 6.

These CNNs could have been learned independently. However, sharing the first two layers has two key advantages. First, the low-level features learned from case-insensitive character classification allows *sharing training data* among tasks, reducing overfitting and improving performance in classification tasks with less informative labels (text/no-text classification), or tasks with fewer training examples (case-sensitive character classification, bigram classification). Second, it allows *sharing computations*, significantly increasing the efficiency.

4 End-to-End Pipeline

This section describes the various stages of the proposed end-to-end text spotting system, making use of the features learnt in Sect. 3. The pipeline starts with a detection phase (Sect. 4.1) that takes a raw image and generates candidate bounding boxes of words, making use of the text/no-text classifer. The words contained within these bounding boxes are then recognized against a fixed lexicon of words (Sect. 4.2), driven by the character classifiers, bigram classifier, and other geometric cues.

4.1 Text Detection

The aim of the detection phase is to start from a large, raw pixel input image and generate a set of rectangular bounding boxes, each of which should contain the image of a word. This detection process (Fig. 4) is tuned for high recall, and generates a set of candidate word bounding boxes.

The process starts by computing a **text saliency map** by evaluating the character/background CNN classifier in a sliding window fashion across the image, which has been appropriately zero-padded so that the resulting text saliency

[1] Training a dedicated classifier was found to yield superior performance to using the background class in the 37-way case-sensitive character classifier.

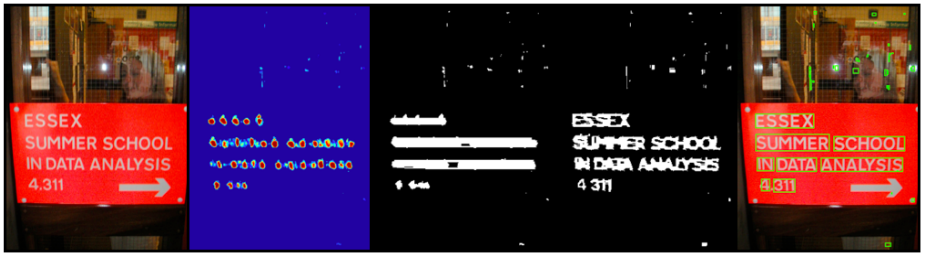


Fig. 4. The detector phase for a single scale. From left to right: input image, CNN generated text saliency map using that text/no-text classifier, after the run length smoothing phase, after the word splitting phase, the implied bounding boxes. Subsequently, the bounding boxes will be combined at multiple scales and undergo filtering and non-maximal suppression.

map is the same resolution as the original image. As the CNN is trained to detect text at a single canonical height, this process is repeated for 16 different scales to target text heights between 16 and 260 pixels by resizing the input image.

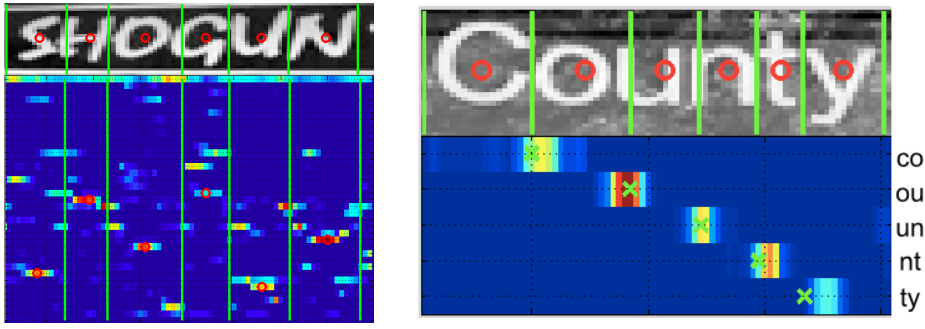
Given these saliency maps, word bounding boxes are generated independently at each scale in two steps. The first step is to **identify lines of text**. To this end, the probability map is first thresholded to find local regions of high probability. Then these regions are connected in text lines by using the *run length smoothing algorithm* (RLSA): for each row of pixels the mean μ and standard deviation σ of the spacings between probability peaks are computed and neighboring regions are connected if the space between them is less than $3\mu - 0.5\sigma$. Finding connected components of the linked regions results in candidate text lines.

The next step is to **split text lines into words**. For this, the image is cropped to just that of a text line and Otsu thresholding [37] is applied to roughly segment foreground characters from background. Adjacent connected components (which are hopefully segmented characters) are then connected if their horizontal spacings are less than the mean horizontal spacing for the text line, again using RLSA. The resulting connected components give candidate bounding boxes for individual words, which are then added to the global set of bounding boxes at all scales. Finally, these bounding boxes are filtered based on geometric constraints (box height, aspect ratio, *etc.*) and undergo non-maximal suppression sorting them by decreasing average per-pixel text saliency score.

4.2 Word Recognition

The aim of the word recognition stage is to take the candidate cropped word images $I \in \mathbb{R}^{W \times H}$ of width W and height H and estimate the text contained in them. In order to recognize a word from a fixed lexicon, each word hypothesis is scored using a generative model that combines multiple visual cues. The computational complexity is therefore linear in the lexicon size.

The input to the word recognition are the 2D character probability maps (case sensitive and insensitive) and bigram probability maps generated using the CNN classifiers. Restricted to the cropped word region, this results in a $W \times H$



(a) (b)

Fig. 5. (a) The optimal placing of breakpoints for the word "SHOGUN" in the image, with the 1D character response map for 37 character classes below. Each row of the response map is the horizontal CNN response for a particular character, with classes in row order from top to bottom: no-text, 0-9, a-z (*i.e.* first row is the no-text class response, last row is the "z" class response). (b) The optimal breakpoint placing for "County" with the bigram responses of only the ground-truth text bigrams shown below. Green lines show the placed breakpoints b^w with red circles showing the implied character center.

map for each character hypothesis. These $W \times H$ maps are reduced to $W \times 1$ responses by averaging along the columns (averaging uses a Gaussian weight centered on the middle row). Grouping the 1D responses per classifier type, this result in matrices $P \in \mathbb{R}^{37\times W}$, $Q \in \mathbb{R}^{63\times W}$, $R \in \mathbb{R}^{604\times W}$ for the case-insensitive, case-sensitive, and bigram classifier classifiers respectively (Fig. 5).

Given matrices P, Q, R, the next step is to score each word hypothesis $w = (c_1, c_2, \ldots, c_{L_w})$. Let $b^w = (b^w_1, b^w_2, \ldots, b^w_{L_w+1})$ denote the breakpoints between characters (where b^w_1 marks the beginning of the first character and the $b^w_{L_w}$ the end of the last one). The word-breakpoints hypothesis (w, b^w) receives score

$$s(w, b^w, P, Q, R) = \frac{1}{|b^w|}\left(\sum_{i=1}^{|b^w|} m_i(b^w_i, R) + \sum_{i=2}^{|b^w|} \phi(b^w_i, b^w_{i-1}, P, Q, R)\right). \quad (1)$$

For each word hypothesis w the optimal location of breakpoints are determined using dynamic programming and the word with best score $s(w, I) = \max_{b^w} s(w, b^w, P, Q, R)$ is recognized.

The unary scores $m_i(b^w_i, R)$ combine the following cues: distance from expected breakpoint placement, distance to out of image bounds, no-text class score, the bigram score, and, for the first and last breakpoint, the distance from the edge of the image. The pairwise score $\phi(b^w_i, b^w_{i-1}, P, Q, R)$ combines: the character score at the midpoint between breakpoints, the no-text score at the character center, the deviation from the average width of the character, and a dynamic contribution from the left and right bigram responses relative to the character score – this allows bigram responses to take control when it is difficult to classify the character in focus, but easy to classify characters on either side. Also it is ensured that there is no violation of the sequence of characters in the

Label	Description	Lex. size	# im.	# words
IC03	ICDAR 2003 [1] test dataset.	–	251	860
IC03-50	ICDAR 2003 [1] test dataset with fixed lexicon.	50	251	860
IC03-Full	ICDAR 2003 [1] test dataset with fixed lexicon.	860	251	860
SVT	SVT [46] test dataset.	–	250	647
SVT-50	SVT [46] test dataset with fixed lexicon.	50	250	647

Table 1. A description of the various datasets used to evaluate on. *# im.* denotes the total number of images in the dataset, and *# words* the total number of word occurrences.

word and that the character centers are all in the region of the word image. Each score is weighted and linearly combined, with the parameters found by grid search on a validation set.

Given the recognized word, the bounding box is adjusted to match the estimated breakpoints and added to a list of candidate recognized word regions. The final step is to perform non-maximal suppression on this set of bounding boxes in order to eliminate duplicate detections.

5 Experiments

This section evaluates our method on a number of standard text-spotting benchmarks. Data and technical details are discussed next, with results in Sect. 5.1.

Datasets. We train and evaluate on a number of datasets. The **four ICDAR datasets** – ICDAR 2003 [1], 2005 [28], 2011 [42], and 2013 [24] – containing a varied array of photos of the world that contain scene text. ICDAR 2003, 2005, and 2013 have word and character bounding box annotations, whereas the ICDAR 2011 dataset contains only word bounding box annotations. The **Street View Text dataset** (SVT) [46] contains images downloaded from Google Street View of road-side scenes, and only has case-insensitive word bounding box annotations. The labelled text can be very challenging with a wide variety of fonts, orientations, and lighting conditions. **KAIST** provide a scene text dataset [5] consisting of 3000 images of indoor and outdoor scenes containing text, with both a mixture of photos from a high-resolution digital camera and a low-resolution mobile phone camera. Word and character bounding boxes are provided as well as segmentation maps of characters, and the words are a mixture of English and Korean. The **Oxford Cornmarket Scene Text** dataset [39] provides high resolution images of a busy street scene, with case-insensitive, word level bounding box annotations. Not all text is labelled, but there are some difficult samples. The **Chars74k** [10] and the **StanfordSynth** [47]. Both datasets contain small single-character images of all 62 characters (0-9, a-z, A-Z). Chars74k comprises a set of characters from natural images, and a set of synthetically generated characters. The StanfordSynth characters are all synthetically generated, but are very representative of natural scene characters, whereas the Chars74k synthetic characters are not.

Fig. 6. Some training samples used for bigram classification as seen by the CNN. From left to right, the top row labels are "de", "sf", "aw", "lp", "oa", and "ad".

Classifier Training Data. The case-insensitive character classifier is learned on 163k cropped 24×24 pixel characters from ICDAR 2003, 2005, 2011, 2011, 2013 training sets, KAIST, the natural images from Chars74k (we do not use the synthetically generated images), StanfordSynth, and FlickrType. After this training, the characters in the SVT training set and Oxford Cornmarket dataset are automatically annotated, and training continues including those samples, giving a total of 186k characters. No-text data is generated from all four ICDAR training datasets (all other datasets do not annotate all text occurrences). The case-sensitive character classifier is trained on the same data, excluding FlickrType and automatically annotated characters, giving 107k training samples. Wherever possible, the characters were cropped and resized to maintain their original aspect ratio.

The bigram classifier performs 604-way classification – the number of unique bigrams present in the ICDAR 2003 and SVT lexicons. We train on 24×24 pixel samples, generated by centering a window with width 1.5 times that of the height at the breakpoint between two characters, cropping the image, and resizing to 24×24 pixels, thus squashing the aspect ratio. We use the character annotations from the ICDAR 2003, 2005, 2011, 2011, 2013 training sets, KAIST dataset, FlickrType and the automatically annotated datasets giving a total of 92k samples. However, due to the relative distribution of bigrams in natural text, some bigram classes have no training data, while others have thousands of samples – on average there are 152 image samples per bigram class.

Weak Baseline System. The Flickr mining process described in Sect. 2 uses a weak baseline end-to-end text spotting system based on the Stroke Width Transform (SWT) [16]. First, the SWT word detection algorithm is run as described in [16]. The SWT labels each pixel with a value of the width of the stroke it is estimated to belong to. Regions of similar stroke width are combined into connected components and are character candidates. Using simple heuristics [16], these character candidates are grouped together to form words, generating word bounding boxes, character bounding boxes, as well as rough character segmentations. This process is run with multiple sets of parameters, as the SWT is very sensitive to changes in them, producing a large number of word detection candidates. A random forest classifier based on [7] is then used to produce a text saliency map for each candidate bounding box, rejecting false positive SWT word detections. For each remaining word detection candidate, the rough SWT character segmentations are used to generate a color model to fully segment characters using Graph Cut [9], after which the word detections are filtered based on profile features described in [15]. Finally, the segmented word detec-

	Character Classifier (%)		Case-sensitive Character Classifier(%)	Text/No-text Classifier (%)		Bigram Classifier (%)
Method	IC03	SVT	IC03	IC03	SVT	IC03
Wang & Wu [47]	-	-	83.9	97.8*	-	-
Alsharif [6]	89.8	-	86.0	-	-	-
Proposed	**91.0**	**80.3**	**86.8**	**98.2**	**97.1**	**72.5**

Table 2. The accuracy of classifiers for 36-way character classification, 62-way case-sensitive character classification, 2-way text detection, and 604-way bigram classification on ground-truth cropped patches. For the SVT dataset the character-level annotation is automatically generated by our system. *Value read from graph.

tion is fed in to an off-the-shelf OCR package, Tesseract [2], for case-insensitive character recognition. When tested on the standard benchmarks, this baseline system achieves an unconstrained end-to-end word recognition f-measure of 0.50 on ICDAR 2003 ([33] get 0.41, higher is better) and 0.42 on ICDAR 2011 ([35] get 0.45).

5.1 Results

This section compares the performance of our system against the standard benchmarks and state-of-the-art. It also reports the performance of the individual components of the system. The evaluation datasets are given in Table 1.

CNN classifiers. Table 2 shows the results of the trained classifiers on ICDAR 2003 cropped characters (IC03) and SVT automatically annotated cropped characters, as well as ICDAR 2003 cropped bigrams. To make results comparable with published ones, the background class is ignored in this case. The 37-way case-insensitive character classifier and case-sensitive classifier both achieve state-of-the-art performance, as does the text/no-text classifier used for detection. Our bigram classifier gives a recognition accuracy of 72.5%, a good result for a problem with 604 classes.

Although the CNNs are large (2.6 million parameters for the case-insensitive classifier), the method does not require any unsupervised pre-training as is done in [47], and incorporating the unsupervised approach described in [47, 13] gives no improvement. Empirically, maxout and dropout were found to be essential to achieve this performance. For example, replacing maxout with ReLU nonlinearities (this equates to reducing the number of filters to give the same layer output dimensionality) causes slight overfitting hence worse accuracy (-3.3% accuracy for case-insensitive character classification). We also found experimentally that pooling and downsampling have no effect on classifier accuracy.

Sharing feature maps between tasks improves results compared to learning independent models: $+3\%$ accuracy for text/no-text, $+1\%$ accuracy for the case-sensitive character classifier, and $+2.5\%$ accuracy for the bigram classifier.

Fig. 7. Five randomly chosen cropped groundtruth cropped words out of only 33 that were recognized incorrectly in the IC03-50 cropped word benchmark.

Method	Cropped Words		
	IC03-50	IC03-Full	SVT-50
Wang [45]	76.0	62.0	57.0
Mishra [31]	81.8	67.8	73.2
Novikova [36]	82.8	-	72.9
Wang & Wu [47]	90.0	84.0	70.0
Alsharif [6]	93.1	88.6	74.3
Goel [19]	-	-	77.3
PhotoOCR [8]	-	-	**90.4**
Proposed	**96.2**	**91.5**	86.1

Method	End-to-End		
	IC03-50	IC03-Full	SVT-50
Wang [45]	68	51	38
Weak Baseline	-	55	41
Wang & Wu [47]	72	67	46
Alsharif [6]	77	70	48
Proposed	**80**	**75**	**56**

Table 3. Left: Ground-truth cropped word recognition accuracy (%) on different datasets. **Right**: End-to-end word recognition F-measure results (%). These methods report PASCAL VOC style 50% overlap match measure for the detection.

Including the FlickrType mined data also gives an extra 0.8% accuracy for the case-insensitive character classifier, illustrating the importance of more training data. On the contrary, learning on more synthetic data from Chars74k dataset (black and white renderings of different fonts) and Wang *et al.* [45] harmed recognition accuracy by causing the CNN to overfit to the synthetic data.

Fixed Lexicon Cropped Word Recognition. The cropped word recognition accuracy of the recognition sub-system (Tab. 3) is evaluated following the protocol of [45] (in particular, words smaller than two characters are ignored). For each word, a set of hypothesis is formed adding to the ground-truth text a small number of distractors. These distractors are: in IC03-full the full lexicon, in IC03-50 the 50 words from [45], and in SVT, 50 selected words.

Our word recognition system gives state of the art accuracy on the ICDAR 2003 benchmarks, improving on state of the art by 3.1% for the 50 word lexicon and 2.4% for the full lexicon. The total recognition accuracy of 96.2% for the 50 word lexicon makes only 33 mistakes out of 860 test images, and many of the misclassified examples can be very difficult to classify even for a human (Fig. 7). On the SVT dataset, we achieve an accuracy of 86.1%, which while 4.3% off state-of-the-art, improves on the next best result [19] by 8.8%. This is a competitive result considering our method is trained on two orders of magnitude less data, and so must use a smaller model than the state-of-the-art PhotoOCR [8] method.

End-to-End Word Recognition. The results of our end-to-end system are evaluated on the ICDAR 2003 test set with different sized lexicons and the SVT dataset (Tab. 3, Fig. 8). A recognition result is considered to be correct if the

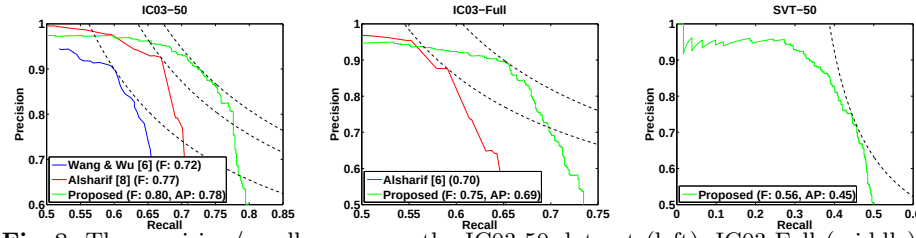


Fig. 8. The precision/recall curves on the IC03-50 dataset (left), IC03-Full (middle), and SVT-50 dataset (right) with lines of constant F-measure. The results from [47, 6] were extracted from the papers.

bounding box has at least 50% overlap with the ground truth and the text is correct. The detector described in Sect. 4.1 is tuned for high recall and generates word bounding boxes with localization P/R (precision/recall) of 0.19/0.80 (IC03) and 0.04/0.72 (SVT). After word recognition, detection results are re-ranked by word recognition score, P/R curves generated, and the P/R/F-measure at the maximum F-measure working point is reported [47, 45, 6]. Our end-to-end pipeline outperforms previous works, with P/R/F-measure of 0.90/0.73/0.80 for IC03-50, 0.89/0.66/0.75 for IC03-Full, and 0.73/0.45/0.56 for SVT-50. Interestingly, due to the fact that our pre-recognition bounding boxes are generated by a detector trained from the same data as the character recognizer, we find that the difference between the localization and recognition scores to be inline with the cropped word recognition results: at maximum recall, 95% of correctly *localized* words are subsequently correctly *recognized* in IC03-50. When removing the bigram response maps from the word recognition process, F-measure drops significantly from 0.8 to 0.76 for IC03-50 and from 0.56 to 0.50 for SVT-50.

6 Conclusions

In this paper we have presented a text spotting system using a single set of rich, learnt features, that achieve state-of-the-art performance on a number of benchmarks. These results illustrate the power of jointly learning features to build multiple strong classifiers as well as mining additional training data in publicly available resources such as Flickr. One additional potential advantage, to be explored in future work, is that by implementing sliding window detection as a byproduct of the convolutional network, our method allows the use of CNN speedup methods [23, 30] to dramatically accelerate detection. In addition, it would be interesting to continue classifier learning by self-supervision using the system to continually and automatically mine more data, and find other sources for this. Finally, this framework is not only limited to fixed lexicon recognition, as the major contributions are agnostic as to whether a lexicon is used or not.

Acknowledgements. Funding for this research is provided by the EPSRC and ERC grant VisRec no. 228180, and thanks to Prof. Ingmar Posner for his insights in discussions.

References

1. http://algoval.essex.ac.uk/icdar/datasets.html
2. https://code.google.com/p/tesseract-ocr/
3. http://www.flickr.com/
4. http://www.flickr.com/groups/type/
5. http://www.iapr-tc11.org/mediawiki/index.php/kaist_scene_text_database
6. Alsharif, O., Pineau, J.: End-to-End Text Recognition with Hybrid HMM Maxout Models. ICLR (2014)
7. Anthimopoulos, M., Gatos, B., Pratikakis, I.: Detection of artificial and scene text in images and video frames. Pattern Analysis and Applications pp. 1–16 (2011)
8. Bissacco, A., Cummins, M., Netzer, Y., Neven, H.: PhotoOCR: Reading text in uncontrolled conditions. ICCV (2013)
9. Boykov, Y., Jolly, M.P.: Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images. In: Proc. ICCV. vol. 2, pp. 105–112 (2001)
10. de Campos, T., Babu, B.R., Varma, M.: Character recognition in natural images pp. 591–604 (2009)
11. Chen, H., Tsai, S., Schroth, G., Chen, D., Grzeszczuk, R., Girod, B.: Robust text detection in natural images with edge-enhanced maximally stable extremal regions. In: Proc. International Conference on Image Processing (ICIP). pp. 2609–2612 (2011)
12. Chen, X., Yuille, A.L.: Detecting and reading text in natural scenes. In: Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on. vol. 2, pp. II–366. IEEE (2004)
13. Coates, A., Carpenter, B., Case, C., Satheesh, S., Suresh, B., Wang, T., Wu, D.J., Ng, A.Y.: Text detection and character recognition in scene images with unsupervised feature learning. In: Document Analysis and Recognition (ICDAR), 2011 International Conference on. pp. 440–445. IEEE (2011)
14. Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., Darrell, T.: Decaf: A deep convolutional activation feature for generic visual recognition. arXiv preprint arXiv:1310.1531 (2013)
15. Dutta, S., Sankaran, N., Sankar, K., Jawahar, C.: Robust recognition of degraded documents using character n-grams. In: Document Analysis Systems (DAS), International Workshop on. pp. 130–134. IEEE (2012)
16. Epshtein, B., Ofek, E., Wexler, Y.: Detecting text in natural scenes with stroke width transform. In: Proc. CVPR. pp. 2963–2970. IEEE (2010)
17. Erhan, D., Bengio, Y., Courville, A., Vincent, P.: Visualizing higher-layer features of a deep network. Tech. rep., University of Montreal (2009)
18. Felzenszwalb, P., Huttenlocher, D.: Pictorial structures for object recognition. IJCV 61(1) (2005)
19. Goel, V., Mishra, A., Alahari, K., Jawahar, C.: Whole is greater than sum of parts: Recognizing scene text words. In: Document Analysis and Recognition (ICDAR), 2013 12th International Conference on. pp. 398–402. IEEE (2013)
20. Goodfellow, I.J., Bulatov, Y., Ibarz, J., Arnoud, S., Shet, V.: Multi-digit number recognition from street view imagery using deep convolutional neural networks. ICLR (2014)
21. Goodfellow, I.J., Warde-Farley, D., Mirza, M., Courville, A., Bengio, Y.: Maxout networks. arXiv preprint arXiv:1302.4389 (2013)
22. Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.R.: Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580 (2012)

23. Jaderberg, M., Vedaldi, A., Zisserman, A.: Speeding up convolutional neural networks with low rank expansions. arXiv preprint arXiv:1405.3866 (2014)
24. Karatzas, D., Shafait, F., Uchida, S., Iwamura, M., Mestre, S.R., Mas, J., Mota, D.F., Almazan, J.A., de las Heras, L.P., et al.: Icdar 2013 robust reading competition. In: Document Analysis and Recognition (ICDAR), 2013 12th International Conference on. pp. 1484–1493. IEEE (2013)
25. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NIPS. vol. 1, p. 4 (2012)
26. Lalonde, M., Gagnon, L.: Key-text spotting in documentary videos using adaboost. In: Electronic Imaging 2006. p. 60641N. International Society for Optics and Photonics (2006)
27. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE 86(11), 2278–2324 (1998)
28. Lucas, S.M.: Icdar 2005 text locating competition results. In: Document Analysis and Recognition, 2005. Proceedings. Eighth International Conference on. pp. 80–84. IEEE (2005)
29. Matas, J., Chum, O., Urban, M., Pajdla, T.: Robust wide baseline stereo from maximally stable extremal regions. In: Proc. BMVC. pp. 384–393 (2002)
30. Mathieu, M., Henaff, M., LeCun, Y.: Fast training of convolutional networks through FFTs. CoRR abs/1312.5851 (2013)
31. Mishra, A., Alahari, K., Jawahar, C., et al.: Scene text recognition using higher order language priors. In: BMVC 2012-23rd British Machine Vision Conference (2012)
32. Neumann, L., Matas, J.: A method for text localization and recognition in real-world images. In: Proc. Asian Conf. on Computer Vision, pp. 770–783. Springer (2010)
33. Neumann, L., Matas, J.: Text localization in real-world images using efficiently pruned exhaustive search. In: Proc. ICDAR. pp. 687–691. IEEE (2011)
34. Neumann, L., Matas, J.: Real-time scene text localization and recognition. In: Proc. CVPR. vol. 3, pp. 1187–1190. IEEE (2012)
35. Neumann, L., Matas, J.: Scene text localization and recognition with oriented stroke detection. In: 2013 IEEE International Conference on Computer Vision (ICCV 2013). pp. 97–104. IEEE, California, US (December 2013)
36. Novikova, T., Barinova, O., Kohli, P., Lempitsky, V.: Large-lexicon attribute-consistent text recognition in natural images. In: Proc. ECCV, pp. 752–765. Springer (2012)
37. Otsu, N.: A threshold selection method from gray-level histograms. IEEE Transactions on Systems, Man, and Cybernetics 9(1), 62–66 (1979)
38. Ozuysal, M., Fua, P., Lepetit, V.: Fast keypoint recognition in ten lines of code. In: Proc. CVPR (2007)
39. Posner, I., Corke, P., Newman, P.: Using text-spotting to query the world. In: Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS) (2010)
40. Quack, T.: Large scale mining and retrieval of visual data in a multimodal context. Ph.D. thesis, ETH Zurich (2009)
41. Rath, T., Manmatha, R.: Word spotting for historical documents. IJDAR 9(2-4), 139–152 (2007)
42. Shahab, A., Shafait, F., Dengel, A.: Icdar 2011 robust reading competition challenge 2: Reading text in scene images. In: Proc. ICDAR. pp. 1491–1496. IEEE (2011)

43. Simonyan, K., Vedaldi, A., Zisserman, A.: Deep inside convolutional networks: Visualising image classification models and saliency maps. In: Workshop at International Conference on Learning Representations (2014)
44. Torralba, A., Murphy, K.P., Freeman, W.T.: Sharing features: efficient boosting procedures for multiclass object detection. In: Proc. CVPR. pp. 762–769 (2004)
45. Wang, K., Babenko, B., Belongie, S.: End-to-end scene text recognition. In: Proc. ICCV. pp. 1457–1464. IEEE (2011)
46. Wang, K., Belongie, S.: Word spotting in the wild. In: ECCV (2010)
47. Wang, T., Wu, D.J., Coates, A., Ng, A.Y.: End-to-end text recognition with convolutional neural networks. In: Pattern Recognition (ICPR), 2012 21st International Conference on. pp. 3304–3308. IEEE (2012)
48. Weinman, J.J., Butler, Z., Knoll, D., Feild, J.: Toward integrated scene text reading. IEEE Trans. Pattern Anal. Mach. Intell. 36(2), 375–387 (Feb 2014)
49. Yang, H., Quehl, B., Sack, H.: A framework for improved video text detection and recognition. In: Int. Journal of Multimedia Tools and Applications (MTAP) (2012)
50. Yi, C., Tian, Y.: Text string detection from natural scenes by structure-based partition and grouping. Image Processing, IEEE Transactions on 20(9), 2594–2605 (2011)
51. Yin, X.C., Yin, X., Huang, K.: Robust text detection in natural scene images. CoRR abs/1301.2628 (2013)