Detection and Recognition of Text Embedded in Online Images via Neural Context Models

Chulmoo Kang, Gunhee Kim and **Suk I. Yoo**
Department of Computer Science and Engineering
Seoul National University, Seoul, Korea
Email: { (nkarma, gunhee, sukinyoo)@snu.ac.kr }
Project page: https://github.com/cmkang/CTSN

Abstract

We address the problem of detecting and recognizing the text embedded in online images that are circulated over the Web. Our idea is to leverage context information for both text detection and recognition. For detection, we use local image context around the text region, based on that the text often sequentially appear in online images. For recognition, we exploit the metadata associated with the input online image, including tags, comments, and title, which are used as a topic prior for the word candidates in the image. To infuse such two sets of context information, we propose a *contextual text spotting network* (CTSN). We perform comparative evaluation with five state-of-the-art text spotting methods on newly collected Instagram and Flickr datasets. We show that our approach that benefits from context information is more successful for text spotting in online images.

Introduction

The use of photos in social networks is growing recently, because the photo posts often generate more engagement than text-only posts. Interestingly, a large portion of images that are circulated over the Web embed the text, as shown in Fig.1. Such text embedding on online photos becomes popular for several reasons. First, it can accompany important information about photos such as authors, location, and time. Second, text can be used as a caption if the image is a video frame captured from news clips, movies, or TV episodes. Finally, text often magnifies the message of the photo by making it funny, sarcastic, inspiring, or hilarious, which can lead higher engagement from other users. However, the text embedded in Web images has been largely ignored as a digital dark matter in the current information retrieval system, although the text often conveys the key messages of such visual memes. For example, in the *reddit*, almost identical smiley Obama picture can be used in completely opposite ways by embedding different text.

The objective of this work is to detect and recognize the text embedded in online images. That is, we aim at localizing the regions of text in the image, and recognize the words in them, as shown in Fig.1. This task is generally referred to as *text spotting* (Bissacco et al. 2013; Epshtein, Ofek, and Wexler 2010; Chen and Yuille 2004; Jaderberg, Vedaldi, and Zissermann 2014; Neumann and Matas 2012; Wang, Babenko, and Belongie 2011; Wang, Wu, and Ng 2012). To address this problem, we propose a *contextual text spotting network* (CTSN), whose major novelty is to take advantage of context information for both text detection and recognition tasks. For detection of text regions, we use local image context around the region of interest, based on that the text often sequentially appear (*i.e*. a region near to text is also likely to be text). Such local image neighbors have been popularly used as a contextual cue in object detection literature such as (Divvala et al. 2009; Gkioxari, Girshick, and Malik 2015). For recognition, we leverage metadata associated with the input online image, including tags, comments, and title, which are used as contextual topic prior for the word candidates in the image. To the best of our knowledge, there has been no convolutional recurrent neural network model for text spotting that leverages such two sets of context information.

It is worth noting that spotting the text superimposed by users in online images is different from that in natural images (*e.g*. house numbers or signs in street views). In the former task, text is often denser and shows much variation of font sizes and styles as shown in Fig.1, whereas the text in the latter task may bear severe perspective or illumination challenges. Preferably, the two tasks need to be tackled with different approaches, although both are referred to as the same *text spotting*. Targeting at the former task, we design our algorithm to take advantage of additional metadata context, and regional context from dense text regions, to enhance the performance.

For evaluation, we newly collect Instagram and Flickr datasets, consisting of about $2.6K$ images with $26K$ words with a large variety of locations, scales, and fonts. We compare our approach with five state-of-the-art text spotting methods, and show that our approach is particularly successful for the online images to perform the three basic text spotting tasks: text detection, cropped-word recognition, and end-to-end recognition.

Finally, we highlight main contributions of this work as follows. (1) We design the *contextual text spotting network* (CTSN) that boosts the text detection and recognition performance by taking advantage of context information. Our method is motivated by the fact that general users' photos often embed sequential text, and associate with multiple infor-

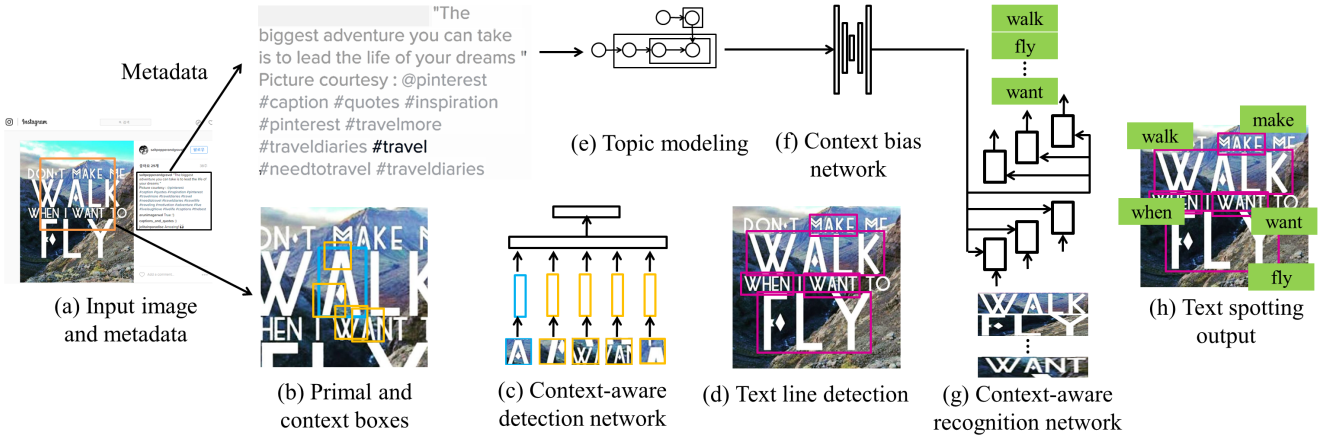

Figure 1: Overview of our *contextual text spotting network* (CTSN) model to detect and recognize the text embedded in online images by taking advantage of context information. To improve text region detection, we use the contextual cues of local neighborhood around the region of interest as shown in (b)–(d). To enhance word recognition in the image, we extract the contextual topics as prior for likely words that appear in the image, from the metadata associated with the input image (*e.g.* tags, comments, and titles) as shown in (e)–(g).

mative metadata to be used as a topic prior of likely words in the images. To the best of our knowledge, our work is unique not only in proposing the idea of leveraging context information, but also in developing neural network models that incorporate such two sets of context information into both text detection and word recognition models. (2) We evaluate our approach on novel Instagram and Flickr datasets. With comprehensive empirical comparison with five state-of-the-art text detection and recognition algorithms, we demonstrate that our approach is more successful for text spotting in the online images of Instagram and Flickr.

Related work

The text detection and recognition in natural images have been studied much in computer vision research. While some recent survey papers (Ye and Doermann 2015; Yin et al. 2016) present more comprehensive literature survey on the text spotting research, we here review a representative selection of previous papers that are closely related to this work. As a major problem domain, city and street images have been popularly studied because they include much informative text in the scene (Chen and Yuille 2004; Wang, Babenko, and Belongie 2011; Mishra, Alahari, and Jawahar 2012b; Zhang et al. 2015; Tian et al. 2015). Some previous work focuses on practicality with low latency; for examples, Epshtein *et al*. (Epshtein, Ofek, and Wexler 2010) propose a fast local image operator called the *stroke width transform* (SWT), leveraging the assumption that text tends to maintain the fixed stroke width in natural images. Another example is (Neumann and Matas 2012) that proposes to use the MSER detector as an end-to-end real-time text localization and recognition system, which is implemented as a part of openCV 3.0. In their later work (Busta, Neumann, and Matas 2015), more accurate and faster stroke-specific keypoints are proposed to replace the MSER features. The system of Google (Bissacco et al. 2013) focuses on a text extraction from smart phone imagery, for which it achieves a fast processing time of less than 1 second with help of datacenter-scale distributed language modelling. There have been several efforts to explicitly identify and tackle on the key difficulties of text spotting. Some notable examples include arbitrary orientations in natural images (Yao et al. 2012), perspective distortion (Phan et al. 2013), and different orientations, languages, and fonts (Kang, Li, and Doermann 2014). Another interesting work is spatial transformer networks (Jaderberg et al. 2015) that leverage CNNs to correct the distortions of translation, scale, rotation and clutter of characters.

With the emergence of deep neural networks, several approaches have been proposed to leverage convolutional neural networks (CNN) for robust text spotting (*e.g.* (Wang, Wu, and Ng 2012; Huang, Qiao, and Tang 2014; Jaderberg, Vedaldi, and Zissermann 2014; He et al. 2016; Zhang et al. 2016; Zhu and Zanibbi 2016)). Our method is closely related to this group of work because we also take advantage of CNNs and RNNs. Wang *et al*. (Wang, Wu, and Ng 2012) propose one of the earliest LeNet-based text detector and character recognizer. Huang *et al*. (Huang, Qiao, and Tang 2014) address a text detection method in which they first generate text candidate regions using the low-level MSERs operator, and then apply high-level sliding-window style CNN classifiers. Jaderberg *et al*. (Jaderberg, Vedaldi, and Zissermann 2014) propose an end-to-end CNN model that integrates the whole pipeline of character recognition, text detection, and word recognition. He *et al*. (He et al. 2016) propose deep-text recurrent networks (DTRN) that consist of CNNs for recognizing the words, and RNNs for decoding the CNN output sequence into a word string. Compared to a large group of previous deep learning based text recognition methods, our work is unique in that it proposes *contextual* text spotting approach for the first time, and shows that the context information indeed helps improve text detection and recognition accuracies for online images.

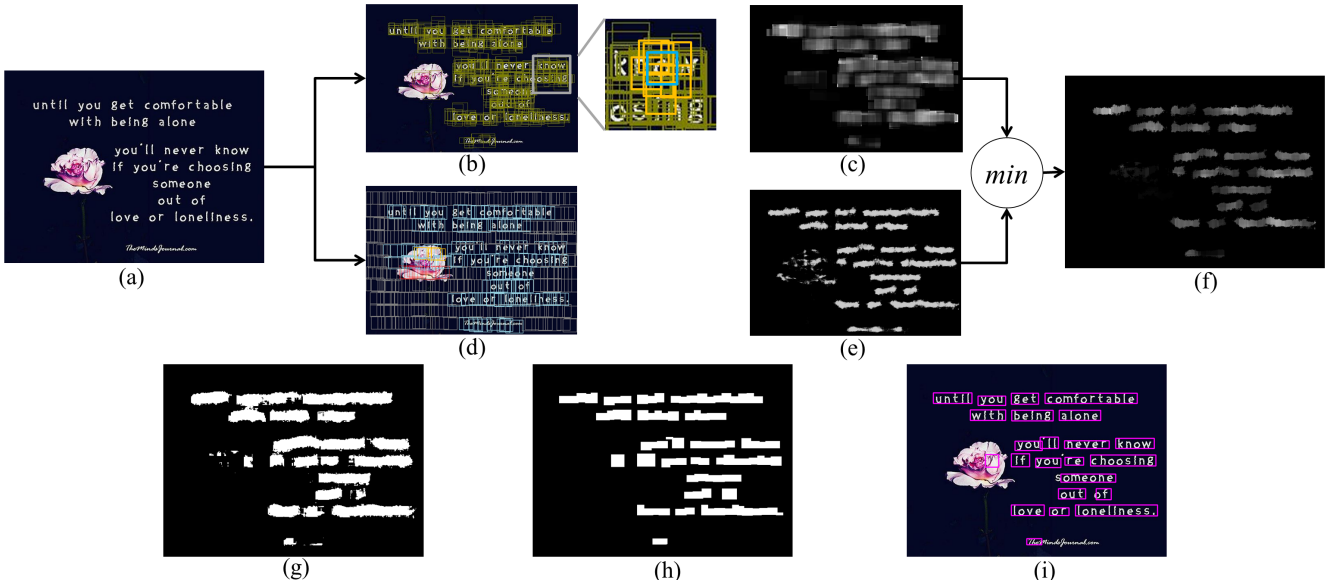

Figure 2: The pipeline of the proposed context-aware detection. (a) An input image. (b) Extracted Edge Box proposals as yellow boxes. In the right, one selected primal box (cyan) and its context boxes (green) are shown. (c) A contextual text response map after processing all the edge box proposals. (d) Context-free text detection using the slide-window approach. (e) A context-free text response map. (f) The final text response map by pixelwise min-pooling on the two maps of (c) and (e). (g) A binarized image of (f) via Otsu thresholding (Otsu 1979). (h) Refine textline boxes via MSER detection (Nistér and Stewénius 2008) followed by geometric property filtering (Dillencourt, Samet, and Tamminen 1992). (i) A final textline detection result.

Approach for Contextual Text Spotting

The text spotting aims at localizing the embedded text with bounding boxes for an input image, and then recognizing it. We use the context information in different ways for detection and recognition, which are presented below in details.

Context-Aware Text Detection

For text detection, we first enumerate a large number of candidates of text-likely regions, and then detect bounding boxes called *textline boxes* that are probable to contain individual lines of text. Ideally, each textline box corresponds to a valid word consisting of multiple characters. Fig.2 shows the pipeline of this stage.

Text proposals. Our first step is to generate a set of primitive text proposals through a single pass of Edge Boxes (Zitnick and Dollár 2014). We extract about $2K$ boxes per image. As shown in Fig.2(b), each edge box proposal mostly corresponds to a region for a single character.

Text detection with local context. Our next step is to decide whether each edge box proposal includes text or not. As shown in Fig.2(b), each edge box proposal is overlapped with many other proposals; thus it is advantageous to consider together local context (*i.e.* neighbor edge boxes) for the decision, because text is likely to sequentially appear horizontally or vertically (*i.e.* a region near to text is also likely to be text). Based on this intuition, we design the *context-aware text detection network* in Fig.3(a). We call the edge box of interest as a *primal box* and the proposals overlapped with the primal box as *context boxes*. We limit the maximum number of context boxes to K (*e.g.* $K = 10$). We randomly sample if the overlapped edge boxes are more than this limit. As local context, we use overlapped proposals instead of simply enlarged regions around the primal box, because the overlapped proposals are often text-likely regions, whereas the enlarged regions tend to contain uninformative regions as well.

The context-aware text detection network in Fig.3(a) consists of $K + 1$ *text feature extractors* (TFE), one for the primal box, and the other K for the context boxes. The text feature extractor is a 10-layer CNN designed based on VGGNet (Simonyan and Zisserman 2015). Its structure is shown in Fig.3(e), comprising six convolutional layers, three max-pool layers, and two fully-connected layers. The output dimension is (256×1). The K context box features $\mathbf{t}_{c,1}, \ldots, \mathbf{t}_{c,K}$ are then concatenated into a single vector $\mathbf{t}_c \in \mathbb{R}^{K\times 256}$, which is fed into a fully-connected layer with rectified linear units (ReLU) (Nair and Hinton 2010). Its output $\mathbf{t}_{c,o}$ is concatenated with the primal box feature $\mathbf{t}_p$, and then inputs to the softmax layer:

$$\mathbf{t}_{c,o} = \text{ReLU}(\mathbf{W}_c\mathbf{t}_c + \mathbf{b}_c), \quad (1)$$

$$\mathbf{d}_o = \text{softmax}(\mathbf{W}_d[\mathbf{t}_p \parallel \mathbf{t}_{c,o}] + \mathbf{b}_d), \quad (2)$$

where the parameters include $\mathbf{W}_c \in \mathbb{R}^{256\times(K*256)}$, $\mathbf{b}_c \in \mathbb{R}^{256\times 1}$, $\mathbf{W}_d \in \mathbb{R}^{2\times 512}$, and $\mathbf{b}_d \in \mathbb{R}^{2\times 1}$. The output of the binary softmax classifier is a distribution over text or nontext labels: $\mathbf{d}_o = [s_o, 1-s_o]^\top$, from which we obtain the score s_o of text likelihood for the primal box. We compute the score s_o for all the edge box proposals.

The contextual text response map. From the output of text detection, we construct the *contextual text response map* M_c that has the same size with the input image and assigns

the likelihood of text at every pixel in a range of [0, 1] (See an example in Fig.2(c)). We superimpose the scores s_o of all edge boxes, and normalize the score at every pixel by dividing the number of edge boxes in which the pixel involves.

The context-free text response map. In addition to contextual response map M_c, we generate another map M_f named as the *context-free text response map* that has the same size and the same range of values with M_c, as shown in Fig.2(e). We apply the *context-free text detector* in Fig.3(f), using a multi-scale sliding window approach with a size of 36×36 and a stride of 1 pixel. We vary the scale from 0.5 to 1.0 at every interval of 0.1. The context-free text detector (CFD) has the almost similar structure to the text feature extractor (TFE) in Fig.3(e), only except that the final layer is a two-way softmax layer so that it outputs the text likelihood score. We also normalize the score at every pixel by dividing the number of windows in which the pixel involves.

Extraction of textline boxes. The intuition behind using both contextual M_c and context-free M_f is that the two maps synergistically help each other to achieve better detection performance. From our observation, the context-aware detection attains a higher accuracy for text region detection; yet it has a relatively poorer spatial resolution since it has the same response values for all the pixels inside an edge box proposal. On the other hand, the context-free detection has a lower accuracy for text region detection, but it can achieve a better spatial resolution.

To take the advantage of the two text response maps with or without context, we apply the pixelwise min-pooling between the two maps as illustrated in Fig.2(f); $M(i,j) = \min(M_c(i,j), M_f(i,j))$ for $i = 1, \cdots, h$ and $j = 1, \cdots, v$ where h and v are the height and width of the image, respectively. That is, via a min-pooing with a high-resolution of context-free M_f, we can reduce the scores of the regions that are incorrectly assigned to high scores due to a coarse resolution of contextual M_c.

For refinement of detection, we perform the following post-processing steps. We first binarize M by Otsu thresholding (Fig.2(g)), and then refine positive text response regions (*i.e*. displayed as white pixels) applying the MSER algorithm (Nistér and Stewénius 2008). Next we filter out less text-likely regions using the REGIONPROPS function of Matlab, which verifies whether the detected regions pass all the geometric criteria to be text-likely, in terms of areas, aspect ratios and *Euler numbers*. After the filtering, we binarize the image again (Fig.2(h)), from which we obtain the textline detection result (Fig.2(i)) by applying the connected component analysis (Dillencourt, Samet, and Tamminen 1992), which calculates the locations of each text region.

Context-Aware Word Recognition

For each textline box in the image, we perform the word recognition to find out what words it contains. We first build a lexicon dictionary using all the words in our dataset, excluding single-character words (*e.g*. I, a, A) and the words longer than 15 characters. The dictionary size is $V = 3,202$. Since most online images have metadata in the form of text, such as title, tags, and comments, we leverage their topic information as a context prior on the likely word candidates for the image. In other words, the topics extracted from metadata can scope down the possible words that are likely to appear in the images. For instance, if the metadata deals with the *travel* topic, then the words in the image are likely to be too. As a result, given that our word dictionary size is 3,202, the word recognition reduces from 3,202-way classification to fewer-than-it-way classification.

We design a context-aware convolutional recurrent network model for word recognition as shown in Fig.3(b). It consists of three core components: bias networks for context, character recognition networks, and bidirectional recurrent networks for word recognition.

Bias networks for context. We design a compact neural network model that can infuse the context bias to the word recognition network. Since there is no guarantee that the embedded text in the image includes the exact words in the metadata, we leverage a topic model that represents the distribution of semantic topics of the metadata in a low-dimensional space. We first represent the metadata using the bag-of-words model, which counts the number of occurrences of each distinct word in the dictionary. We build a word dictionary using all the words in our metadata, excluding the stop words listed by the NLTK(*Natural Language Toolkit*)[1], single-character words, and online informal words (*e.g*. like4like, follow4follow).

We use the *Latent Dirichlet Allocation* (LDA) (Blei, Ng, and Jordan 2003; Mikolov and Zweig 2012) to encode the metadata by a topic vector, denoted by $\mathbf{v}_m \in \mathbb{R}^D$, where D is the topic dimension (*e.g*. $D = 128$). Thus, according to the topic distribution of metadata, we assign weights on the candidate words. We input $\mathbf{v}_m$ into the context bias network, consisting of three fully-connected layers with ReLU activation, and one softmax layer, as shown in Fig.3(c):

$$\mathbf{b}_1 = \text{ReLU}(\mathbf{W}_{v1}\mathbf{v}_m), \tag{3}$$

$$\mathbf{b}_2 = \text{ReLU}(\mathbf{W}_{v2}\mathbf{b}_1), \tag{4}$$

$$\mathbf{b}_3 = \text{ReLU}(\mathbf{W}_{v3}\mathbf{b}_2), \tag{5}$$

$$\mathbf{b} = \text{softmax}(\mathbf{W}_{v4}\mathbf{b}_3), \tag{6}$$

where the parameters are $\mathbf{W}_{v1} \in \mathbb{R}^{D/2\times D}$, $\mathbf{W}_{v2} \in \mathbb{R}^{D/4\times D/2}$, $\mathbf{W}_{v3} \in \mathbb{R}^{D/2\times D/4}$, and $\mathbf{W}_{v4} \in \mathbb{R}^{D\times D/2}$. The bias network is designed as a structure of autoencoder (Hinton and Salakhutdinov 2006). We apply the dropout regularization with a rate of 0.5 to the three fully-connected layers. Later the output $\mathbf{b} \in \mathbb{R}^D$ is provided to the recurrent word recognition network as a bias term. For the image with no metadata, we use a zero vector for $\mathbf{b}$.

We perform a simple correlation test between image text and corresponding metadata text in our datasets. We compute LDA vectors from the extracted TF-IDF vectors, and measure average cosine similarity, which turns out to be 0.125 in a range from -1 to 1. It means that the image and metadata text are correlated up to the level where predicting one is helped by the other without severe overfitting.

Recurrent word recognition network. Fig.3(b) illustrates the proposed context-aware word recognition network. We first resize the input textline box to have its minimum

[1] http://www.nltk.org/

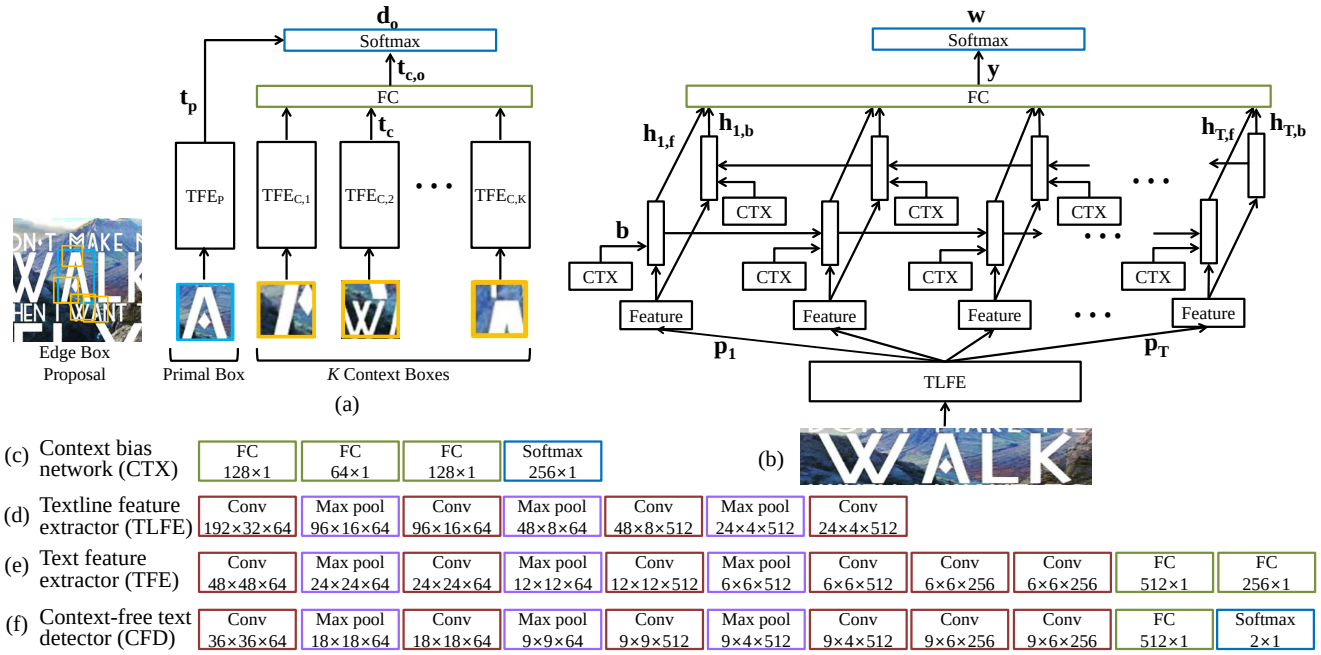

Figure 3: The illustration of the proposed context-aware detection and recognition networks in (a) and (b), respectively. Several key network components are shown below, including (c) context bias network (CTX in the model), (d) textline feature extractor (TLFE), (e) text feature extractor (TFE), and (f) context-free text detector (CFD).

dimension to be 32 pixels while preserving its aspect ratio. If the maximum dimension is smaller than h_{max} (*e.g.* $h_{max} = 192$), we zeropad to be h_{max}. Otherwise, we resize the image so that the maximum dimension to be h_{max} with ignoring the aspect ratio this time. We then input the resized textline box into the *textline feature extractor* (TLFE), which is a CNN shown in Fig.3(d). Its output is a sequence of T feature maps with a size of (4×512): $\mathbf{p} = (\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_T) \in \mathbb{R}^{T \times 4 \times 512}$. We set $T = 24$, and thus we can detect the word up to this length. Then the bidirectional RNN learns the mapping from the sequence of feature maps to a sequence of likely characters as a word. We use the bidirectional RNNs, because it can exploit both past and future feature maps to recognize a whole sequence of characters at once as a word. We exploit normal RNNs instead of LSTMs, due to its better performance in our evaluation.

Given an input sequence of feature maps $\mathbf{p}$, the BRNN updates the forward and backward hidden states denoted by $\mathbf{h}_f$ and $\mathbf{h}_b \in \mathbb{R}^D$ respectively. The output $\mathbf{y} \in \mathbb{R}^{D \times 2T}$ is a concatenation of T forward and backward hidden states, which is fed into a softmax layer over all the words in the dictionary to predict the index of the most likely word. This formulation is represented by

$$\mathbf{h}_{t,f} = \text{ReLU}(\mathbf{W}_{ph,f}\mathbf{p}_t + \mathbf{W}_{hh,f}\mathbf{h}_{t-1,f} + \mathbf{b}), \quad (7)$$

$$\mathbf{h}_{t,b} = \text{ReLU}(\mathbf{W}_{ph,b}\mathbf{p}_t + \mathbf{W}_{hh,b}\mathbf{h}_{t+1,b} + \mathbf{b}) \quad (8)$$

$$\mathbf{y} = [\mathbf{h}_{1,f}, \mathbf{h}_{2,f}, ..., \mathbf{h}_{T,f} \parallel \mathbf{h}_{1,b}, h_{2,b}, ..., \mathbf{h}_{T,b}], \quad (9)$$

$$\mathbf{w} = \text{softmax}(\mathbf{W}_w\mathbf{y} + \mathbf{b}_w) \quad (10)$$

where $\mathbf{b}$ is the topic bias computed in Eq.(3). As shown in Fig.3(b), we input the contextual topic bias $\mathbf{b}$ at every step of the BRNN to mitigate the vanishing gradient problem. The parameters include $\mathbf{W}_{ph,f}, \mathbf{W}_{ph,b} \in \mathbb{R}^{D \times 4 \times 512}$, $\mathbf{W}_{hh,f}, \mathbf{W}_{hh,b} \in \mathbb{R}^{D \times D}$, $\mathbf{W}_w \in \mathbb{R}^{V \times D \times 2T}$, and $\mathbf{b}_w \in \mathbb{R}^{V \times 1}$. The output of the softmax output layer $\mathbf{w} \in \mathbb{R}^V$ is the likelihood over all the words in the dictionary. Finally, we select the word of maximum likelihood by $\text{argmax}_{v \in V}\, \mathbf{w}(v)$.

Training

The two key components of our model are context-aware detector and recognizer, each of which is trained in an end-to-end manner. Before learning the two models, we pretrain various network components as follows.

Context bias network. For each training image, we compute the LDA topic distributions of embedded text and associated metadata. The former is used as groundtruth (GT) of the bias network, and the latter is used as the input to the network. The loss function is defined by the cosine distance:

$$d = 1 - \mathbf{t}_{gt}^T \mathbf{t}_p / \|\mathbf{t}_{gt}\| \|\mathbf{t}_p\| \quad (11)$$

where $\mathbf{t}_p$ and $\mathbf{t}_{gt}$ denote the topic vectors of the prediction and the GT, respectively. The learned weights are used as an initialization, and updated in an end-to-end way during the training of the recognition network.

Text feature extraction networks. The three text feature networks (*i.e.* TLFE, TFE, CFD in Fig.3(d)(e)(f)) share not only similar structure based on VGGNet, but also the similar objective as character recognizers. We pretrain them using standard character datasets of Chars74K (de Campos, Babu, and Varma 2009) and IIIT5K (Mishra, Alahari, and Jawahar 2012a). We insert a batch normalization layer (Ioffe and Szegedy 2015) to every convolutional layer except the first

one. The pretrained weights are used as initialization for following end-to-end training of the text detection network.

Text detection network. We obtain positive examples from the edge boxes that have an overlap area ratio larger than 0.9 with the groundtruth (GT) textline boxes, and negative samples from the ones that have no intersection. We randomly sample $N(= 10)$ positive and negative examples per training image. During training, we minimize the negative log-likelihood of conditional probability of prediction. We use Nesterov momentum (Sutskever et al. 2013) as an optimizer with a learning rate of $1e^{-4}$, a momentum of 0.9, and a decrease ratio of 0.9 every 20 epochs.

Word recognition network. We first pretrain the recognition network of Fig.3(b) using the synthetic word data of (Jaderberg et al. 2014), consisting of $8M$ training images for $90K$ lexicons. We minimize the negative log-likelihood as the loss function to use the Nesterov momentum with a learning rate of $3e^{-3}$ and a momentum of 0.9. We apply three batch normalization to all the convolutional layers except the first one. We then finetune the recognition network for our dataset using the same optimizer with a learning rate of $5e^{-4}$ and a decrease ratio of 0.9 every epoch.

Experiments

We present more details of dataset collection and experimental setting in our project page[2], where we also make public our source code and datasets.

Experimental Setup

Instagram and Flickr Datasets. We collect two datasets of text embedded images from Instagram and Flickr. The images are challenging for both detection and recognition, due to a large variation of fonts, sizes, numbers of words even in a single image.

The Instagram dataset consists of 2,233 images with about $22K$ words in total(*i.e*. about 10 words per image on average), and is randomly split into 1,786 training and 447 test images. We use the Instagram build-in search engine for image crawling, and then let human labelers to annotate words that can be recognizable and are longer than a single character, using the bounding box annotation toolbox[3]

The Flickr dataset comprises 424 images with about 4,262 words, and is randomly split into 213 training and 211 test images. We also crawling the metadata associated with images if available for both Instagram and Flickr datasets.

Evaluation. To report the performance, we follow the protocol of ICDAR 2013 competition (Karatzas et al. 2013). For detection evaluation, we first decide whether the detection is *true positive* (TF) or *false positive* (FP). The TF is a detection whose intersection-over-union (IoU) metric with groundtruth (GT) is larger than 0.5; otherwise it is a FP. We then compute the F1-score as a balanced average metric between precision and recall: $2(1/\text{precision}+1/\text{recall})^{-1}$. For recognition evaluation, we count the success if the estimated word of a TP detection is identical to the GT word. Even with a single wrong character, we count it as a failure. We then compute the recognition accuracy by (# successes)/(# TP detections).

Baselines. We compare our approach with five state-of-the-art methods, which are based on object detection of pictorial structure (Wang, Babenko, and Belongie 2011) (`TPS`), extremal regions detection (Neumann and Matas 2012) (`TER`), stroke-specific keypoints (Busta, Neumann, and Matas 2015) (`TFAST`), convolutional neural networks (Wang, Wu, and Ng 2012)(`TCNN`), and convolutional recurrent networks (Shi, Bai, and Yao 2015) (`TCRNN`). For all the baselines, we use publicly available codes provided by the authors. Some baselines do not deal with both detection and recognition; for example, (`TFAST`) for detection only and (`TCRNN`) for cropped word recognition only.

Results

Table 1 summarizes the results of three tasks of text detection, cropped-word recognition, and end-to-end recognition. We test two variants of our CTSN with or without the context to quantize the performance gain by the use of context. In the table, we use N/A (not available) for some baselines that are only applicable to detection or recognition. We observe that the context indeed helps enhance the performance.

For text detection, our approach with context significantly outperforms other state-of-the-art baselines with large margins (*e.g*. a higher F1 score by 9.2%p than the best baseline (`TER`) for Instagram). The improvement by our CTSN comes from using the local context, which suppresses false positives significantly with a little loss of detection sensitivity. This property coincides with the result that the CTSN relatively attains a higher score for precision than recall.

The goal of cropped-word recognition is to recognize the word for a given clearly cropped word image. This evaluation quantifies the recognition ability of each method. Our approach achieves better performance than all the baselines, as shown in Table 1. The context information improves performance by excluding out-of-topic words from the algorithm's consideration. In other words, the metadata of an online image provide topic-level information, which is useful in predicting the words embedded in the image. The performance increase of recognition by the context is less significant (3.1% for Instagram) than that of detection (26.3%). It is partly due to that our word recognizer already achieves high accuracy (87.3%), and thus the context has less room for the improvement of recognition than detection.

The end-to-end task involves both word localization and recognition from input images. That is, algorithms should accomplish both tasks, to be counted as a success. Table 1 assures that our method outperforms all the baselines for both datasets (*e.g*. higher by 17.1%p than the best (`TER`) for Instagram).

Qualitative Results. Fig.4 illustrates some selected examples of text spotting. We depict successful detections with green boxes, recognition failures with blue boxes, and detection failures with yellow boxes. For true positive detections, we also present the recognition output by our algorithm. As shown in the examples, online images often contain a long sequence of words embedded by users with a high variation

[2] `http://github.com/cmkang/CTSN`.

[3] `http://www.ipb.uni-bonn.de/html/software/annotation_tool/`.

of locations, fonts, and sizes of text. The failure cases include word splitting errors, part-only detections, missed detections of low-contrast fonts, and wrong word recognition.

Conclusion

We proposed the contextual text spotting network (CTSN) model to detect and recognize text embedded in online images. We designed a neural network model that takes advantage of context information: local neighbor patches for detection and metadata associated with images for recognition. With quantitative evaluation on newly collected Instagram and Flickr dataset, we showed that our CTSN achieved better performance than other state-of-the-art methods. One important future direction that go beyond this work is to improve detection accuracies for natural scene images that include fewer and less dense words.

Acknowledgement. Gunhee Kim is partially supported by Basic Science Research Program through National Research Foundation of Korea (2015R1C1A1A02036562).

References

Bissacco, A.; Cummins, M.; Netzer, Y.; and Neven, H. 2013. PhotoOCR: Reading Text in Uncontrolled Conditions. In *ICCV*.

Blei, D. M.; Ng, A. Y.; and Jordan, M. I. 2003. Latent Dirichlet Allocation. *JMLR* 3:993–1022.

Busta, M.; Neumann, L.; and Matas, J. 2015. FASText: Efficient Unconstrained Scene Text Detector. In *ICCV*.

Chen, X., and Yuille, A. L. 2004. Detecting and Reading Text in Natural Scenes. In *CVPR*.

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

Dillencourt, M. B.; Samet, H.; and Tamminen, M. 1992. A general approach to connected-component labeling for arbitrary image representations. *Journal of ACM* 39(2):253–280.

Divvala, S. K.; Hoiem, D.; Hays, J. H.; Efros, A. A.; and Hebert, M. 2009. An Empirical Study of Context in Object Detection. In *CVPR*.

Epshtein, B.; Ofek, E.; and Wexler, Y. 2010. Detecting Text in Natural Scenes with Stroke Width Transform. In *CVPR*.

Gkioxari, G.; Girshick, R.; and Malik, J. 2015. Contextual Action Recognition with RCNN. In *ICCV*.

He, P.; Huang, W.; Qiao, Y.; Loy, C. C.; and Tang, X. 2016. Reading Scene Text in Deep Convolutional Sequences. In *AAAI*.

Hinton, G. E., and Salakhutdinov, R. R. 2006. Reducing the Dimensionality of Data with Neural Networks. *Science* 313(5786):504–507.

Huang, W.; Qiao, Y.; and Tang, X. 2014. Robust Scene Text Detection with Convolution Neural Network Induced MSER Trees. In *ECCV*.

Ioffe, S., and Szegedy, C. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *ICML*.

Jaderberg, M.; Simonyan, K.; Vedaldi, A.; and Zisserman, A. 2014. Synthetic Data and Artificial Neural Networks for Natural Scene Text Recognition. In *arXiv*.

Jaderberg, M.; Simonyan, K.; Zisserman, A.; and Kavukcuoglu, K. 2015. Spatial Transformer Networks. In *NIPS*.

Jaderberg, M.; Vedaldi, A.; and Zissermann, A. 2014. Deep features for Text Spotting. In *ECCV*.

Kang, L.; Li, Y.; and Doermann, D. 2014. Orientation Robus Text Line Detection in Natural Images. In *CVPR*.

Karatzas, D.; Shafait, F.; Uchida, S.; Iwamura, M.; i. Bigorda, L. G.; Mestre, S. R.; Mas, J.; Mota, D. F.; Almazan, J. A.; and de las Heras, L. P. 2013. ICDAR 2013 Robust Reading Competitions. In *ICDAR*.

Mikolov, T., and Zweig, G. 2012. Context Dependent Recurrent Neural Network Language Model. In *IEEE SLT*.

Mishra, A.; Alahari, K.; and Jawahar, C. V. 2012a. Scene Text Recognition using Higher Order Language Priors. In *BMVC*.

Mishra, A.; Alahari, K.; and Jawahar, C. V. 2012b. Top-Down and Bottom-up Cues for Scene Text Recognition. In *CVPR*.

Nair, V., and Hinton, G. E. 2010. Rectified linear units improve restricted boltzmann machines. In *ICML*.

Neumann, L., and Matas, J. 2012. Real-Time Scene Text Localization and Recognition. In *CVPR*.

Nistér, D., and Stewénius, H. 2008. Linear Time Maximally Stable Extremal Regions. In *ECCV*.

Otsu, N. 1979. threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics* 9:62–66.

Phan, T. Q.; Shivakumara, P.; Tian, S.; and Tan, C. L. 2013. Recognizing Text with Perspective Distortion in Natural Scenes. In *ICCV*.

Shi, B.; Bai, X.; and Yao, C. 2015. An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recognition. In *arXiv*.

Simonyan, K., and Zisserman, A. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *ICLR*.

Sutskever, I.; Martens, J.; Dahl, G.; and Hinton, G. E. 2013. On the importance of initialization and momentum in deep learning. In *ICML*.

Tian, S.; Pan, Y.; Huang, C.; Lu, S.; Yu, K.; and Tan, C. L. 2015. Text Flow: A Unified Text Detection System in Natural Scene Images. In *ICCV*.

Wang, K.; Babenko, B.; and Belongie, S. 2011. End-to-End Scene Text Recognition. In *ICCV*.

Wang, T.; Wu, D. J.; and Ng, A. Y. 2012. End-to-End Text Recognition with Convolutional Neural Networks. In *ICPR*.

Yao, C.; Bai, X.; Liu, W.; Ma, Y.; and Tu, Z. 2012. Detecting Texts of Arbitrary Orientations in Natural Images. In *CVPR*.

Ye, Q., and Doermann, D. 2015. Text Detection and Recognition in Imagery: A Survey. *TPAMI* 37(7):1480–1500.

Method	Instagram			Flickr		
	Detection	Recognition	End-to-end	Detection	Recognition	End-to-end
(TER) (Neumann and Matas 2012)	46.3	N/A	26.2	44.6	N/A	25.3
(TFAST) (Busta, Neumann, and Matas 2015)	14.2	N/A	NA	17.6	N/A	NA
(TPS) (Wang, Babenko, and Belongie 2011)	N/A	23.4	10.3	N/A	38.2	13.4
(TCNN) (Wang, Wu, and Ng 2012)	N/A	50.6	6.8	N/A	55.3	10.5
(TCRNN) (Shi, Bai, and Yao 2015)	N/A	87.1	N/A	N/A	88.7	NA
CTSN w/o context	29.2	87.3	18.9	34.0	90.3	20.0
CTSN w context	**55.5**	**90.4**	**43.3**	**55.5**	**93.0**	**39.7**

Table 1: The accuracies of text detection (F1 scores), cropped word recognition (accuracies), and end-to-end recognition (F1 scores) for Instagram and Flickr. Some baselines are only applicable to detection or recognition, for which we use N/A labels.

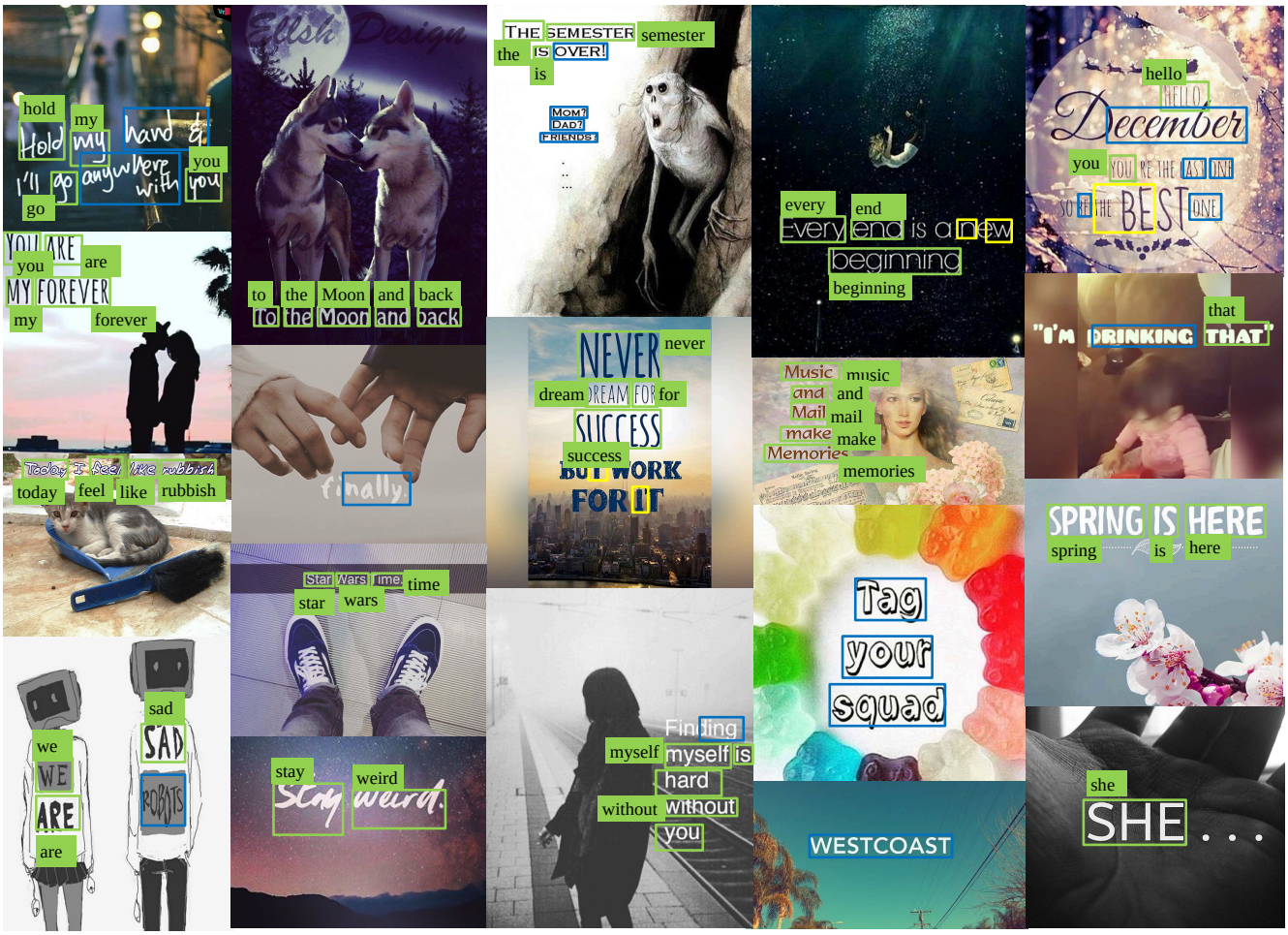

Figure 4: Examples of text detection and recognition with true positives in green, detection successes but recognition failures in blue, and detection failures in yellow.

Yin, X.-C.; Zuo, Z.-Y.; Tian, S.; and Liu, C.-L. 2016. Text Detection, Tracking and Recognition in Video: A Comprehensive Survey. *TIP* 25(6):2752–2773.

Zhang, Z.; Shen, W.; Yao, C.; and Bai, X. 2015. Symmetry-Based Text Line Detection in Natural Scenes. In *CVPR*.

Zhang, Z.; Zhang, C.; Shen, W.; Yao, C.; Liu, W.; and Bai, X. 2016. Multi-Oriented Text Detection with Fully Convolutional Networks. In *CVPR*.

Zhu, S., and Zanibbi, R. 2016. A Text Detection System for Natural Scenes with Convolutional Feature Learning and Cascaded Classification. In *CVPR*.

Zitnick, C. L., and Dollár, P. 2014. Edge Boxes: Locating Object Proposals from Edges. In *ECCV*.