공학박사학위논문

온라인 영상의 텍스트 검출 및 인식을 위한 신경망 문맥 모델

A Neural Contextual Model for Detection and Recognition of Text Embedded in Online Images

2017 년 2 월

서울대학교 대학원

전기·컴퓨터공학부

강 철 무

Abstract

A Neural Contextual Model for Detection and Recognition of Text Embedded in Online Images

Chulmoo Kang
Department of Electrical Engineering and Computer Science
College of Engineering
The Graduate School
Seoul National University

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.

Keywords: Text detection, Text recognition, Context model, Deep learning

Student Number: 2013-30218

Contents

Abstract **i**

Contents **iii**

List of Figures **vi**

List of Tables **xii**

Chapter 1 Introduction **1**

Chapter 2 RelatedWork **5**

Chapter 3 Preliminary of Neural Networks **9**

3.1 Basic of Neural Network 9

3.2 Convolutional Neural Network 12

3.3 Pooling Layer 13

3.4 Activation Function 15

3.5 Recurrent Neural Network 16

3.6 Back-Propagation Through Time 16

3.7 Bidirectional Recurrent Neural Networks 17

3.8 Long-Short Term Memory 17

3.9 Optimization 18

3.10 Training Loss 19

3.11 Training Process 20

Chapter 4 Approach for Contextual Text Spotting 25

4.1 Overview of the Proposed Framework 25

4.2 Context-Aware Text Detection 26

4.2.1 Text Proposals 27

4.2.2 Text Detection Network 28

4.2.3 Extraction of Textline Boxes 33

4.2.4 Text Detection Network Variants 37

4.3 Context-Aware Word Recognition 40

4.3.1 Bias Networks for Context 42

4.3.2 Recurrent Word Recognition Network 43

4.3.3 Recognition Network Variant 46

Chapter 5 Experiments 52

5.1 Dataset 52

5.2 Experimental Setup 55

5.3 Training 56

5.4 Hyperparameters of Contextual Model 59

5.5 Neural Network Architecture Variants 71

5.6 Results 74

Chapter 6 Conclusion **86**

요약 **102**

List of Figures

Figure 1.1 Motivation for detection and recognition of text embedded in online images. Photos that are circulated over social networks often contain embeded texts. In many case, consecutive letters are appeared in the photos along with informative metadata, such as title, comments and tags. Our objective is to propose a novel network model that can take advantage of such context information from subsequent letters and metadata in order to improve the performance of detection and recognition. 4

Figure 3.1 A fully connected neural network. 12

Figure 3.2 The pooling layer. (a) The diagram of pooling layer with kernel size 2 . (b) The input of pooling layer. (c) The output of max pooling layer. (d) The output of min pooling layer. (e) The output of average pooling layer 14

Figure 3.3 The Diagram for RNN (a) The vanilla RNN. (b) The unfolded RNN. (c) The bidirectional RNN 23

Figure 3.4 The diagram of LSTM. 24

Figure 4.1 Overview for 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 topic bias 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). 27

Figure 4.2 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 side-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 [82]. (h) Refined textline boxes via MSER detection [81] followed by geometric property filtering. (i) A final textline detection result. 28

Figure 4.3 The flowchart of the Edge Boxes. 29

Figure 4.4 The flowchart of the Edge Boxes. 30
Figure 4.5 The illustration of the proposed context-aware detection. . . 33
Figure 4.6 The Text Feature Extractor (TFE) network components. . . 34
Figure 4.7 The Context-Free Text Detector (CFD). 35
Figure 4.8 Dropout Network 38
Figure 4.9 The diagram of LeNet. 39
Figure 4.10 The illustration of the proposed maxout detection network . 40
Figure 4.11 The illustration of the proposed context-aware recognition network. 46
Figure 4.12 Context bias network (CTX in the model), ... 47
Figure 4.13 The textline feature extractor (TLFE) 47
Figure 4.14 The illustration of the proposed context-aware recognition network without context bias network. 48
Figure 4.15 The illustration of the directional recurrent neural word recognition network. 49
Figure 4.16 The illustration of the double bidirectional recurrent neural word recognition network. 50
Figure 4.17 The illustration of the context-aware recognition network for fully connected bias network 51

Figure 5.1 Examples of text embeded online images for Instagram. 54
Figure 5.2 Examples of text embeded online images for Flickr. . . 55
Figure 5.3 Examples of text embeded online images for Pinterest. 56
Figure 5.4 The trained weights for context-aware detection network. 60

Figure 5.5 The trained weight for context free detection network . 60
Figure 5.6 The trained weight for textline feature extractor 61
Figure 5.7 The training loss for various number of K(Instagram) . 62
Figure 5.8 The validataion loss for various number of K(Instagram) 62
Figure 5.9 The training loss for various number of K(Flickr). . . . 63
Figure 5.10 The validataion loss for various number of K(Flickr). . 63
Figure 5.11 The training loss for various number of K(Pinterest). . 64
Figure 5.12 The validataion loss for various number of K(Pinterest). 64
Figure 5.13 The training loss for various activation functions(Instagram). 65
Figure 5.14 The validataion loss for various activation functions(Instagram). 65
Figure 5.15 The training loss for various activation functions(Flickr). 66
Figure 5.16 The validation loss for various activation functions(Flickr). 66
Figure 5.17 The training loss for various activation functions(Pinterest). 67
Figure 5.18 The validation loss for various activation functions(Pinterest). 67
Figure 5.19 The training loss of according to optimization method(Flickr) 68
Figure 5.20 The validation loss of according to optimization method(Flickr) 68

Figure 5.21 The training loss of according to optimization method(Pinterest) 69
Figure 5.22 The validation loss of according to optimization method(Pinterest) 69
Figure 5.23 The training loss for Glorot Uniform Initialization vs. Glorot Normal Initialization(Instagram) 70
Figure 5.24 The validation loss for Glorot Uniform Initialization vs. Glorot Normal Initialization(Instagram) 71
Figure 5.25 The validation accuracy for Glorot Uniform Initialization vs. Glorot Normal Initialization(Instagram) 72
Figure 5.26 The training loss for Glorot Uniform Initialization vs. Glorot Normal Initialization(Flickr) 72
Figure 5.27 The validation loss for Glorot Uniform Initialization vs. Glorot Normal Initialization(Flickr) 73
Figure 5.28 The validation accuracy for Glorot Uniform Initialization vs. Glorot Normal Initialization(Flickr) 73
Figure 5.29 The validation score difference between VGG and LeNet for TFE(Instagram) 74
Figure 5.30 The comparison of training loss for TFE(Instagram) . 75
Figure 5.31 The comparison of validation loss for TFE(Instagram) 76
Figure 5.32 The validation score difference between VGG and LeNet for the TFE(Flickr) 77
Figure 5.33 The comparison of training loss for TFE(Flickr) 78
Figure 5.34 The comparison of validation loss for TFE(Flickr) . . . 79

Figure 5.35 The validation score difference between VGG and LeNet for the TFE(Pinterest) 80
Figure 5.36 The comparison of training loss for the TFE(Pinterest) 80
Figure 5.37 The comparison of validation loss for the TFE(Pinterest) 81
Figure 5.38 Examples of text detection and recognition for Instagram with true positives in green, detection successes but recognition failures in blue, and detection failures in yellow. 82
Figure 5.39 Examples of text detection and recognition for Flickr with true positives in green, detection successes but recognition failures in blue, and detection failures in yellow. 83
Figure 5.40 Examples of text detection and recognition for Pinterest with true positives in green, detection successes but recognition failures in blue, and detection failures in yellow. 83

List of Tables

Table 5.1 The results of text detection with precision, recall, F1 scores for Instagram datasets. 77

Table 5.2 The results of text detection with precision, recall, F1 scores for Flickr datasets. 78

Table 5.3 The results of text detection with precision, recall, F1 scores for Pinterest datasets. 79

Table 5.4 The accuracy of cropped word recognition for Instagram datasets. 79

Table 5.5 The accuracy of cropped word recognition for Flickr datasets. 79

Table 5.6 The accuracy of cropped word recognition for Pinterest datasets. 80

Table 5.7 The results of end-to-end recognition with precision, recall, F1 scores for Instagram datasets. 81

Table 5.8 The results of end-to-end recognition with precision, recall, F1 scores for Flickr datasets. 81

Table 5.9 The results of end-to-end recognition with precision, recall, F1 scores for Pinterest datasets. 82

Table 5.10 The repeatability of text detection with precision, recall, F1 scores for Instagram datasets. 84

Table 5.11 The repeatability of cropped word recognition for Instagram datasets. 84

Table 5.12 The repeatability of end to end with precision, recall, F1 scores for Instagram datasets. 85

Chapter 1

Introduction

The use of photos in social network sites is growing recently, since 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.1.1. 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. For example, in the *reddit*, almost identical smiley Obama picture can be used in a completely opposite way by embedding different text. Moreover, the embedded text often conveys the key messages of such visual memes. However, it still remains as a great challenge to automatically understand such valuable information in ever-growing online images.

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.1.1. This task is generally referred to as *text spotting* [5, 21, 11, 50, 80, 105, 109]. 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 [18, 26]. 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.1.1, 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 boost the performance.

For evaluation, we conduct extensive experiments on newly collected Instagram and Flickr datasets, consisting of about $2.8K$ images with $28K$ 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 text spotting tasks of 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 informative 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, Flickr and Pinterest 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, Flickr and Pinterest.

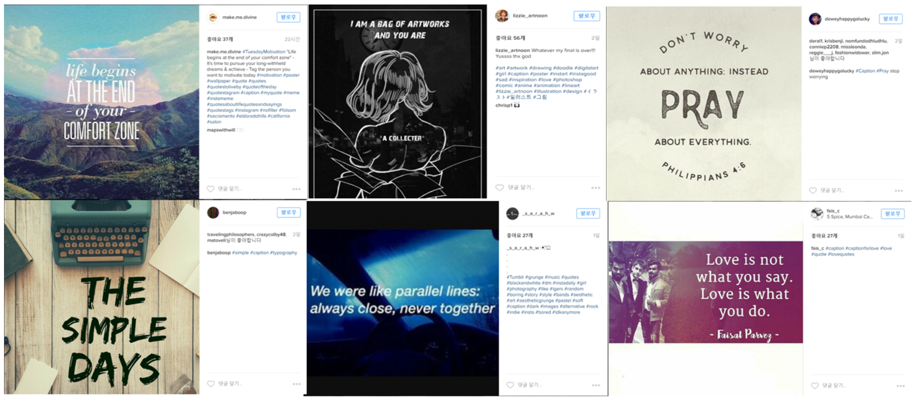


Figure 1.1 Motivation for detection and recognition of text embedded in online images. Photos that are circulated over social networks often contain embeded texts. In many case, consecutive letters are appeared in the photos along with informative metadata, such as title, comments and tags. Our objective is to propose a novel network model that can take advantage of such context information from subsequent letters and meta-data in order to improve the performance of detection and recognition.

Chapter 2

RelatedWork

The text detection and recognition in natural images have been studied much in computer vision research. While some recent survey papers [115, 116] 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.

The problem of text detection and recognition in natural images have been studied much in computer vision research. We review a representative selection of previous papers that are closely related to this work. As a major problem domain, city and street views have been popularly studied because they include much informative text in them [11, 105, 75, 118, 101]. Mishra *et al.* [75] aims to recognize text from street images by exploiting both bottom-up cues (*i.e.* *Conditional Random Field* (CRF) model on individual character detections) and top-down cues (*i.e.* a lexicon-based prior of language statistics). Tian *et al.* [101] also uses the CRF model for text line detection, but the cost of that CRF model is only for text/non-text score. Zhang *et al.* [118] uses the symmetric properties of text lines to localize the text position.

Low Level Vision Approach Some previous work focuses on practicality with low latency; for examples, Epshtein *et al.* [21] propose a fast local image operators called the *Stroke Width Transform* (SWT), leveraging the assumption that text tends to maintain fixed stroke width in natural images. Some modified method by using color image is propoesed by [44] Another example is [80] that proposes to use the *Maximally Stable Extremal Region* (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 later work [7], more accurate and faster stroke-specific keypoints are proposed to replace the MSER features. The other examples of MSER are [45, 79] The system of Google [5] 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 [114], perspective distortion [84], and different orientations, languages, and fonts [52]. Another interesting work is spatial transformer networks [49] that leverage CNNs to correct the distortions of translation, scale, rotation and clutter of characters.

Neural Network A lot of research is being done by the development of deep learning [38, 39, 63, 62]. Object Detection [25, 33, 24, 87, 4], Object Tracking [108, 107, 72, 121], Semantic Segmentation [120, 2, 67, 58, 66, 42, 76, 68, 32, 91, 3, 91, 9, 10, 83, 8], Classification [95, 59, 35, 100], Image Captioning [54, 19, 104, 12, 22, 112, 61, 85, 86, 60] , Neural Language Model [73, 51, 99, 57], Machine Translation[71, 90, 70, 64, 110], Speech Recognition [30, 37, 14, 31, 88, 1, 111] and Image Reconstruction [20, 13, 49, 97, 113, 43, 55, 56, 41]. Object detection and neural language model are closely related to this study.

Neural Network Approach for Text Spotting With the emergence of deep neural networks, several approaches have been recently proposed to leverage *Convolutional Neural Networks* (CNN) for robust text spotting (*e.g.* [109, 45, 50, 36, 119, 122, 93, 102]). Our method is closely related to this group of work because we also take advantage of CNNs and *Recurrent Neural Networks* (RNN)s. Wang *et al.* [109] propose one of the earliest LeNet-based text detector and character recognizer modules. Huang *et al.* [45] 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.* [50] propose an end-to-end CNN model that integrates the whole pipeline of character recognition, text detection, and word recognition. He *et al.* [36] propose deep-text recurrent networks (DTRN) for text recognition, which consist of deep CNNs for recognizing the words and the RNNs for decoding the CNN output sequence into a word string. Zhang *et al.* [119] use modified *Fully Convolutional Neural Network* (FCN)[68] to detect multi-oriented textlines. Zhu *et al.* [122] also use CNN to detect text region pixels followed by connected component analysis. Shi *et al.* [93] especially aim irregular shape text caused by perspective distortion or curved placement for graphical apprearance. Tian *et al*. [102] suggest a neural network architecture that adapt connectionist concept. Liao *et al*. [65] suggests fast detection neural network.

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.

Chapter 3

Preliminary of Neural Networks

The proposed framework is based on the results of the neural network research. To understand the proposed model, this chapter introduces the basic knowledge of *Neural Networks*. Further details can be found in [28].

3.1 Basic of Neural Network

One area of machine learning involves the representation of data as vectors or graphs that can be processed by a computer and building models to learn them. For specific learning objectives, such as recognizing facial or facial expressions, Deep Learning focuses on better representation and better model building for learning. Many of the methods of expressing deep running are inspired by neuroscience and are based on the information processing and communication patterns of the nervous system. Deep Learing succeeded in recognizing Handwritten Postal Code in [63] by applying *Back Propagation* algorithm to neural network. After that, computer performance is improved and Geoffrey Hinton successfully training Multi Layer Neural network by adding pre-training process using Unsupervised learning to Multilayer Neural Network[38], and applying the improved technique to object recognition shown in [59]. Deep learing has begun to attract attention again, while it has improved by about 10% compared with the existing method. After that, it started to be used in various fields. Professor Fei-Fei devised a way to explain images automatically, and many people are doing research with good results in Natural Language Processing. *Neural Neworks* (NNs) are typically organized in layers. Layers are made up of a number of interconnected *nodes* which contain an *activation function*. Patterns are presented to the network via the *input layer*, which is connected to one or more *hidden layers*. The hidden layers then link to an *output layer*. Most NNs contain some form of *learning rule* which modifies the weights of the connections according to the input patterns that it is presented with. Although there are many different kinds of learning rules used by neural networks, the most popular rule is *back propagations*. Back propagation is an abbreviation for the backwards propagation of error. With the backpropagation rule, *learning* is a supervised process that occurs with each cycle or *epoch* (i.e. the cycle of the whole data set is used in training) through a forward activation flow of outputs, and the backwards error propagation of weight adjustments. More simply, when a neural network is initially presented with a pattern it makes a random *initialization* as to what it might be. It then sees how far its answer was from the actual one and makes an appropriate adjustment to its connection weights.

Backpropagation performs a gradient descent within the solution's vector space towards a *global minimum* along the steepest vector of the error surface. The global minimum is that theoretical solution with the lowest possible error. The error surface itself is a hyperparaboloid and *smooth*, there are numerous *pits* and *hills* in the solution space, though. They may cause the network to settle down in a *local minum* which is not the best overall solution.

Since the nature of the error space can not be known a prioi, neural network analysis often requires a large number of individual runs to determine the best solution. Most learning rules have built-in mathematical terms to assist in this process which control the *rate* and the *momentum* of the learning. The rate of learning is actually the one of convergence between the current solution and the global minimum. Momentum helps the network to overcome obstacles (local minima) in the error surface and settle down at or near the global miniumum.

Once a neural network is *trained* to a satisfactory level it may be used as an analytical tool on other data. To do this, the user no longer specifies any training runs and instead allows the network to work in forward propagation mode only. New inputs are presented to the input pattern where they filter into and are processed by the middle layers as though training were taking place, however, at this point the output is retained and no backpropagation occurs. The output of a forward propagation run is the predicted model for the data which can then be used for further analysis and interpretation. It is also possible to *overfit* a neural network, which means that the network has been trained over to respond to training data.

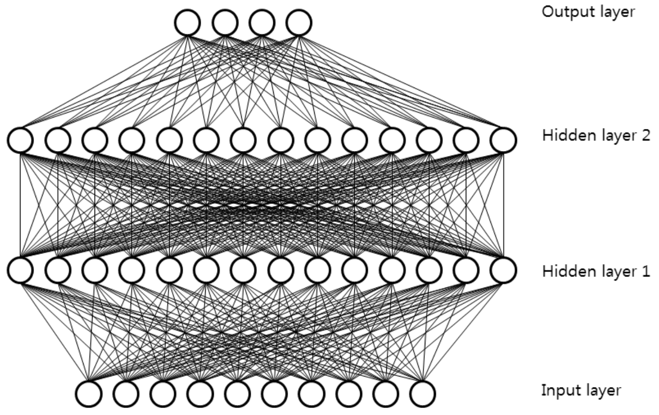


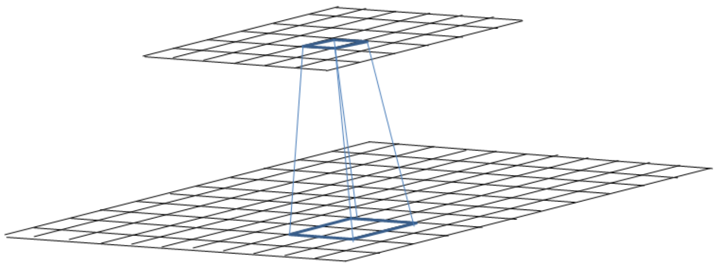
Figure 3.1 A fully connected neural network.

3.2 Convolutional Neural Network

As we saw in the previous section, neural networks receive an input, and transform it through a series of hidden layers. Each hidden layer is made up of a set of neurons, where each neuron is *fully connected* to all neurons in the previous layer, and where neurons in a single layer function completely independently and do not share any connections. The fully connected neural nets don't scale well to full images. If an image has the size of $32 \times 32 \times 3$, so a single fully-connected neuron in a first hidden layer of a fully connected neural network would have $3072 (= 32 \times 32 \times 3)$ weights. This amount still seems manageable, but clearly this fully-connected structure does not scale to larger images. For example, an image of more bigger size, e.g. $640 \times 480 \times 3$, would lead to neurons that have $640 \times 480 \times 3 = 921600$ weights. Moreover, we would almost certainly want to build several layers, so the parameters would increase. Clearly, this full connectivity is wasteful and the huge number of parameters would quickly lead to overfitting. *Convolutional Neural Networks* (CNN) take advantage of the fact that the input consists of images and they constrain the architecture in a more sensible way. In particular, unlike a fully connected neural network, the layers of a CNN have neurons arranged in 3 dimensions: width, height, depth. (Note that the word depth here refers to the third dimension of an activation volume, *the number of feature* to be learned automatically to represent the given dataset). For example, the input images as same as previous example are an input volume of activations, and the volume has dimensions $640 \times 480 \times 3$ (width, height, depth respectively). The neurons in a layer will only be connected to the region of *convolutional kernel size* of the layer before it. The total paramter number is $kernelsize \times kernelsize \times numberoffeatures$. The typical number of kernel size and number of feature are 7 and 128. The total parameter in this case is $7 \times 7 \times 128$. It seems manageable, and moreover is scalable.

3.3 Pooling Layer

The pooling layer, is used to reduce the *spatial* dimensions on a convolution neural network. By having less spatial information you gain computation performance. Less spatial information also means less parameters, so less chance to overfit. You get some translation invariance. Some projects don't use pooling, specially when they want to learn some object specific position. In Fig. 3.2, we show the most common type of pooling, *the maxpooling layer, the minpooling layer, the average pooling layer*, which tiles a window and get the



(a)

1	3	5	7
2	4	6	8
9	11	13	14
10	12	16	15

(b)

4	8
12	16

(c)

1	5
9	13

(d)

2.5	6.5
10.5	14.5

(e)

Figure 3.2 The pooling layer. (a) The diagram of pooling layer with kernel size 2 . (b) The input of pooling layer. (c) The output of max pooling layer. (d) The output of min pooling layer. (e) The output of average pooling layer

biggest value on the window as the output. The meaning of the pooling layer is as follows. Features obtained from the convolutional layer can be recognized differently when the position of the input image is changed. A pooling layer is used to reduce the effect of this position change. A feature is recognized as the same feature in a spatial support. As the layers are increased, then more wide spatial support make the feature as the same even though the position of the image is changed. As a result influence of the position movement is reduced accordingly.

3.4 Activation Function

A function used to transform the activation level of a unit (neuron) into an output signal. Typically, activation functions have a ”squashing” effect. Together with the PSP function (which is applied first) this defines the unit type. Neural Networks supports a wide range of activation functions. Only a few of these are used by default; the others are available for customization.

Identity. The activation level is passed on directly as the output. Used in a variety of network types, including linear networks, and the output layer of radial basis function networks.

Logistic(as known as Sigmoid). This is an S-shaped (sigmoid) curve, with output in the range (0,1). **sigmoid** is defined as:

$$sigmoid(x) = \frac{1}{1 + \exp(-x)} \tag{3.1}$$

where x is the input of the neural network.

Hyperbolic. The hyperbolic tangent function (tanh): a sigmoid curve, like the logistic function, except that output lies in the range (-1,+1). Often performs better than the logistic function because of its symmetry. Ideal for customization of multilayer perceptrons, particularly the hidden layers.

Rectified Linear Unit (ReLU)[77]. ReLU is defined as:

$$ReLU(x) = max(0, x) \tag{3.2}$$

where x is the input of the neural network.

Exponential Linear Units (ELU) ELU is defined as:

$$ELU(x) = \begin{cases} x & \text{if} \quad x > 0 \\ \alpha(\exp(x) - 1) & \text{if} \quad x <= 0 \end{cases}$$

where x is the input of the neural network and α is hyperparameter.

Softmax. Exponential function, with results normalized so that the sum of activations across the layer is 1.0. Can be used in the output layer of multilayer perceptrons for classification problems, so that the outputs can be interpreted as probabilities of class membership.

3.5 Recurrent Neural Network

(RNN) The elementary building blocks of a RNN are neurons (as known as *units*) connected to *sequentially* next neurons. One typically distinguishes input units, internal (or hidden) units, and output units. At a given time, t, a unit has an activation. We denote the input units by $u(t)$, of internal units by $x(t)$, of output units by $y(t)$, for discrete time steps t = 1, 2, 3, ..., shown in Fig. 3.3(a).

3.6 Back-Propagation Through Time

(BPTT) The feedforward backpropagation algorithm cannot be directly transferred to RNNs because the error backpropagation pass presupposes that the connections between units induce a cycle-free ordering. The solution of the backpropagation approach for RNN is to *unfold* the recurrent network in time as shown in Fig. 3.3(b). This gives a feedforward network, which is amenable to the backpropagation algorithm.

3.7 Bidirectional Recurrent Neural Networks

(BRNN) Bidirectional Recurrent Neural Networks extend the unidirectional RNN by introducing a second hidden layer, where the hidden to hidden connections flow in opposite temporal order. The model is therefore able to exploit information both from the past and the future as shown in 3.3(c)

3.8 Long-Short Term Memory

Long Short Term Memory networks(LSTM) are a special kind of RNN, capable of learning long-term dependencies. They were introduced in [40, 29], and were refined and popularized by many people. They work tremendously well on a large variety of problems, and are now widely used. LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behavior, not something they struggle to learn. All recurrent neural networks have the form of a chain of repeating modules of neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer. LSTMs also have this chain like structure, but the repeating module has a different structure. Instead of having a single neural network layer, there are four, interacting in a very special way. The key to LSTMs is the cell state. The cell state is kind of like a conveyor belt. It runs straight down the entire chain, with only some minor linear interactions. The LSTM does have the ability to remove or add information to the cell state, carefully regulated by structures called gates. Gates are a way to optionally let information through. They are composed out of a sigmoid neural net layer and a pointwise multiplication operation. The sigmoid layer outputs numbers between zero and one, describing how much of each component should be let through. The equation of the LSTM is shown below.

$$I_t = tanh(W_{xI}x_t + W_{hI}h_{t-1} + b_I) \tag{3.3}$$

$$J_t = tanh(W_{xJ}x_t + W_{hJ}h_{t-1} + b_J) \tag{3.4}$$

$$F_t = tanh(W_{xF}x_t + W_{hF}h_{t-1} + b_F) \tag{3.5}$$

$$O_t = tanh(W_{xO}x_t + W_{hO}h_{t-1} + b_O) \tag{3.6}$$

$$Cell_t = Cell_{t-1} \odot F_t + I_t \odot J_t \tag{3.7}$$

$$h_t = tanh(Cell_t) \odot O_t \tag{3.8}$$

where $\odot$ means element-wise product, h means hidden state.

3.9 Optimization

Momentum The Stochastic Gradient Descent method suffers when searching for a problem space with ravine shapes. In this case, the SGD scheme oscillates repeatedly in the ravine region, and it is difficult to proceed toward the local optima. This phenomenon is relieved by the momentum method. The momentum method gives the force to maintain in the previous update direction. This increases the probability of going to the local optima in the ravine area. Since momentum keeps the speed at which to proceed, it also speeds up the problem space search. As a result, we gain faster convergence and reduced oscillation.

Nesterov accelerated gradient Sutskever[98] propose an algorithm using the Nesterov momentum method[78]. The conventional method updates the weight in the direction of momentum in the existing direction and the gradient direction in the current position. However, Sutskever separated the existing method into two steps. The first step moves in the direction of the previous momentum. The second step is to update the weight in the direction of the gradient descent at the moved position.

Adagrad Adagrad[16] is an algorithm for gradient-based optimization that does just this: It adapts the learning rate to the parameters, performing larger updates for infrequent and smaller updates for frequent parameters. For this reason, it is well-suited for dealing with sparse data.

Adadelta Adadelta[117] is an extension of Adagrad that seeks to reduce its aggressive, monotonically decreasing learning rate. Instead of accumulating all past squared gradients, Adadelta restricts the window of accumulated past gradients to some fixed size w.

3.10 Training Loss

The training loss is a cost fuction in a neural network, also called objectives. The weight is updated to decrease the training loss. Training losses include square error, binary cross entropy, and categorical cross entropy.

square error The square error is obtained by squaring the difference between the target value and the prediction value.

cross entropy Entropy is used to describe the randomness of probability distributions in information theory. If the probability distribution is uniform, it has a low value and vice versa. Cross entropy represents the similarity of two probability distributions. Cross entropy also has a large value when the difference is large, and a lower value when the difference between the two is small. Because of this characteristic, it is suitable as a cost function. Two distributions represent the distribution of the prediction and the distribution of the target. Use binary cross entropy if the target and prediction are binomial distributions. Use categorical cross entropy for mulitnomial distributions.

3.11 Training Process

The process of training Neural Network should consider weight initialization, learning rate, momentum, overfitting, and weight update method.

Weight Initialization There are many methods of weight initialization, and many studies have been done on this. Basically, there is a way to initialize a uniform or gaussian distribution. Glort's[27] methods is proposed to indicate initialization methods assuming that input data and weights are independent and identically distributed, and there are other methods recently proposed[34, 89]. Among these initialization methods, it is reasonable to choose a method with an experimentally better performance.

Learing Rate The learing rate is a factor that affects the performance of weight update through backpropagation. If the learning rate is too large, it is difficult to fine tune to find the optimum solution. If the learning rate is too small, it will take a long time to converge.

Learning Rate Decay The strategy for speeding up training is to set a high learning rate at the beginning of the traing and lower it to a lower value.

The way to enable this is to use learning rate decay.

Momentum The problem space to solve in deep learing is a very high dimensional space. In order to find the optimal solution in this problem space, places which has gentle slope must pass quickly. The way to do this is to use momentum. The momentum method is a method of multiplying the amount of weight update by a predetermined value and add it to the current weight update.

Overfittng An important problem in the machine learing approach is the overfitting problem. It is important to check if the solution obtained is confined to the data I have, and if it is difficult to generalize, it is difficult to practically use the solution. However, there is often an error in training a higher-level problem space for a longer time to achieve higher performance only for the data I have. A typical way to avoid overfitting is to divide the data into training and validation set. Only training data is used during training, and validation data is used when verifying results. In practice, training loss is monitored for validation data at training time, and when training loss increases for validation data, the training process must be stopped. When dividing data, it is often used to divide it several times, to check the method mentioned above, and to average the values.

Weigh Update There are two ways to update the weight. There is a way to update the weight for one datum each and a method to update the data set, called batch, at once. The former is called the stochastic method and the latter is called the batch update method. The stochastic method is generally known to find better solutions. However, there is a drawback that it takes a longer time. The batch method updates the weight for batch set at a time, so it can be processed more efficiently by hardware such as the GPU. At present, the batch method is widely used.

Weight Regularization When training a neural network, the size of the weight can be larger than the appropriate size of the given problem. The solution to this problem can be overfitting. One way to prevent this is the weight regularization method. This has the same purpose and effect as the regularization of the regression problem. Weight regularization mainly uses the weight norm. Obtain L1 norm or L2 norm of weight and add a weighted of it to training loss. This prevents the L1 norm or L2 norm of the weight from becoming too large. One of the regularizations with the magnitude of the weight is regularization, which limits the maximum size. This method sets the weight value to a predetermined reference value when the weight value exceeds a predetermined reference value. The advantage of this method is that if the learning rate is set higher than the appropriate value, the weight value will be restricted automatically, thereby alleviating the failure of the training.

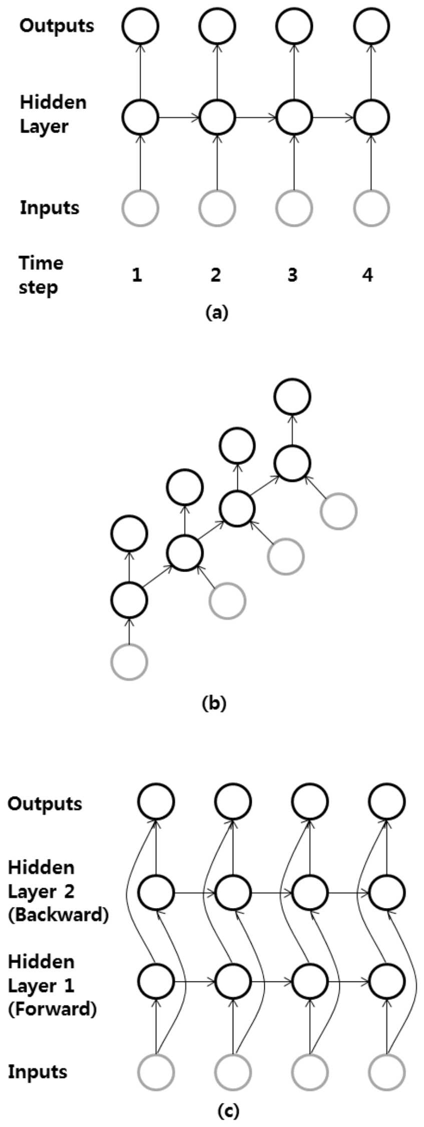


Figure 3.3 The Diagram for RNN (a) The vanilla RNN. (b) The unfolded RNN. (c) The bidirectional RNN

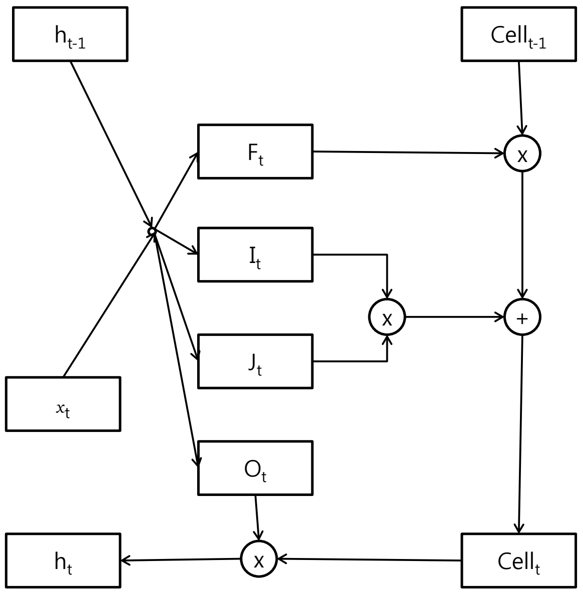


Figure 3.4 The diagram of LSTM.

Chapter 4

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 in Sec. 4.2 and in Sec. 4.3 in details.

4.1 Overview of the Proposed Framework

The method proposed in this paper will be outlined as follows. First, we use the edge box method to generate an object candidate from a given image. For each proposed box, find the primal box and context boxes, and use these to obtain the output of the context-aware detection network. Since the output of the context-aware detection network is degraded in spatial resolution, the context free detection network output is obtained to compensate for the min-pooling. The min-pooling result is converted into a binary image, and a text box detection result is obtained by combining MSER detection and connected component analysis method. Word recognition uses the text box detection result generated in this way. When word recognition is performed, metadata accompanied by image is used. The metadata is the title, comments, tags, and so on. In this metadata, except for NLTK stopwords and internet informal words, it is encoded as one-hot vector and topic modeled by LDA. This is input to the context bias network and the output of this bias network is used to improve word recognition performance. How to recognize the word is here. The image of the text box detection area is rescaled to maintain the aspect ratio, and is made zero mean. This is entered into the context-aware recognition network, where the output of context bias network is input as bias. This network informs the word through the bidirectional recurrent neural network. Let's take a closer look at this step from now on.

4.2 Context-Aware Text Detection

For text detection, we first quickly 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.4.2 shows the pipeline of this stage.

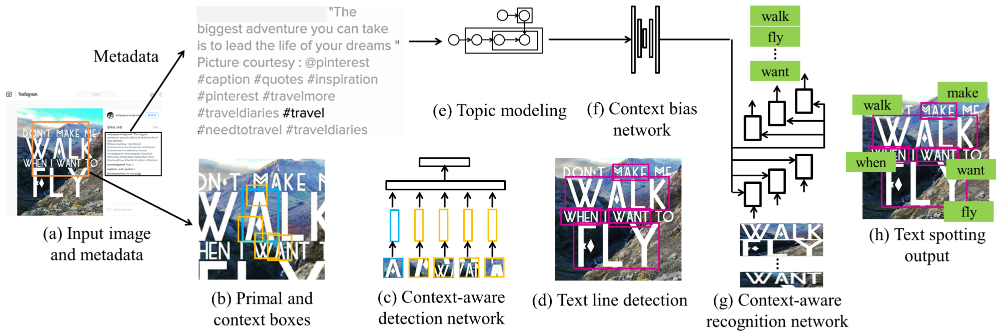

Metadata
"The biggest adventure you can take is to lead the life of your dreams " Picture courtesy : @pinterest #caption #quotes #inspiration #pinterest #travelmore #traveldiaries #travel #needtotravel #traveldiaries
(a) Input image and metadata
(b) Primal and context boxes
(c) Context-aware detection network
(d) Text line detection
(e) Topic modeling
(f) Context bias network
walk
fly
want
(g) Context-aware recognition network
walk
make
when
want
fly
(h) Text spotting output

Figure 4.1 Overview for 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 topic bias 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).

4.2.1 Text Proposals

In order to detect object candidates efficiently, Recent researches propose an efficient method to find object candidates in contrast to existing sliding window method. Among them, there are Edge Boxes[123] method and Selective Search[103] method.

Edge Boxes The edge box method starts from the assumption that the edge of the image contains a lot of information. Get Edge and Edge Grouping, and then obtains Affinities Score to determine object candidates. The flowchart of Edge Boxes is shown in Fig.4.3.

Selective search The Selective Search method uses the primitive seg-

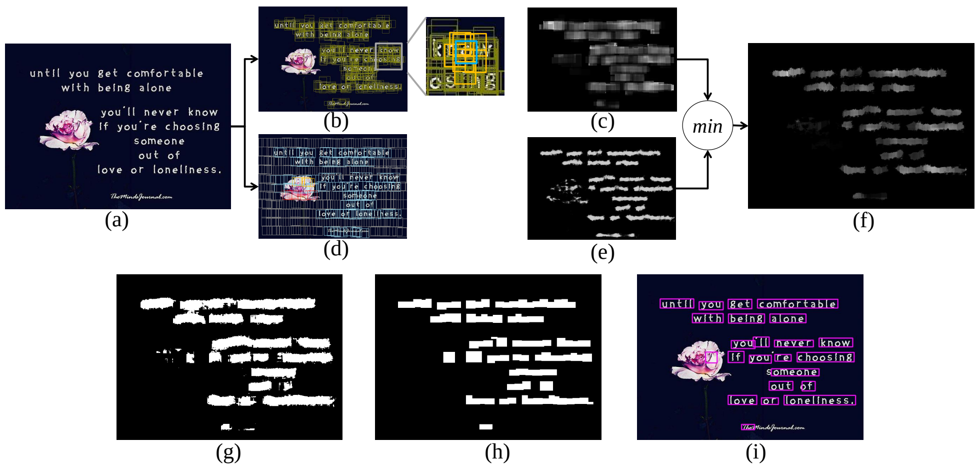


Figure 4.2 The pipeline of the proposed context-aware detection. (a) An input image. (b) Extracted Edge Box proposals as yellow boxes. In the right, one selected primal box (cyan) and its context boxes (green) are shown. (c) A contextual text response map after processing all the edge box proposals. (d) Context-free text detection using the side-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 [82]. (h) Refined textline boxes via MSER detection [81] followed by geometric property filtering. (i) A final textline detection result.

mentation regions proposed in [23] segmentation information to meage similar areas to create an object candidate. When determining similar areas, we use color, texture, size, and overlapping degree.

4.2.2 Text Detection Network

Our next step is to decide whether each edge box proposal includes text or not. To do that, we construct the detection network using context. Detection

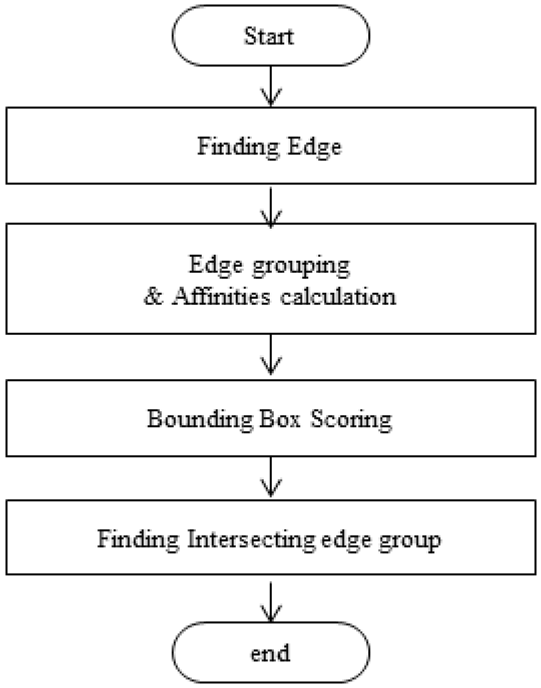


Figure 4.3 The flowchart of the Edge Boxes.

network is constructed using local context based on context categorization by [18]. The following paragraphs describe the local context detection network and describe additional network structures.

Text detection with local context. As shown in Fig.4.2(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.4.5. We call the edge box of

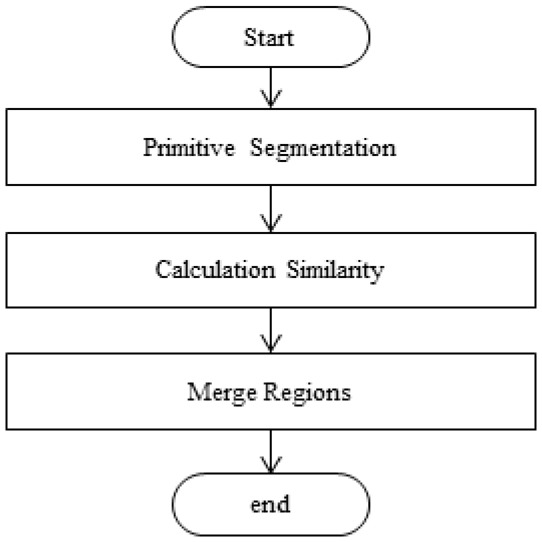


Figure 4.4 The flowchart of the Edge Boxes.

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.

The context-aware text detection network in Fig.4.5 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 [95]. Its structure is shown in Fig.4.6, comprising six convolutional layers, three max-pool layers, and two fully connected layers. The equations for the VGG is as follows:

$$\mathbf{v}gg_1 = \text{ReLU}(\mathbf{W}_{v1}\mathbf{x}), \tag{4.1}$$

$$\mathbf{v}gg_2 = \text{MaxPool}(\mathbf{W}_{v2}\mathbf{v}gg_1), \tag{4.2}$$

$$\mathbf{v}gg_3 = \text{ReLU}(\mathbf{W}_{v3}\mathbf{v}gg_2), \tag{4.3}$$

$$\mathbf{v}gg_4 = \text{MaxPool}(\mathbf{W}_{v4}\mathbf{v}gg_3), \tag{4.4}$$

$$\mathbf{v}gg_5 = \text{ReLU}(\mathbf{W}_{v5}\mathbf{v}gg_4), \tag{4.5}$$

$$\mathbf{v}gg_6 = \text{MaxPool}(\mathbf{W}_{v6}\mathbf{v}gg_5), \tag{4.6}$$

$$\mathbf{v}gg_7 = \text{ReLU}(\mathbf{W}_{v7}\mathbf{v}gg_6), \tag{4.7}$$

$$\mathbf{v}gg_8 = \text{ReLU}(\mathbf{W}_{v8}\mathbf{v}gg_7), \tag{4.8}$$

$$\mathbf{v}gg_9 = \text{ReLU}(\mathbf{W}_{v9}\mathbf{v}gg_8), \tag{4.9}$$

$$\mathbf{v}gg_{10} = \text{ReLU}(\mathbf{W}_{v10}\mathbf{v}gg_9), \tag{4.10}$$

$$\mathbf{v}gg_{out} = \text{ReLU}(\mathbf{W}_{v11}\mathbf{v}gg_{10}), \tag{4.11}$$

where we omit the bias terms for simple notation. The weights for 1,3,5,7,8,9 means the convolutional matrix.

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) [77]. Its output $\mathbf{t}_{c,o}$ is concatenated with the primal box feature $\mathbf{t}_p$, and then inputs to the softmax layer:

$$\mathbf{t}_{c,o} = \text{ReLU}(\mathbf{W}_c\mathbf{t}_c + \mathbf{b}_c), \tag{4.12}$$

$$\mathbf{d}_o = \text{softmax}(\mathbf{W}_d[\mathbf{t}_p \parallel \mathbf{t}_{c,o}] + \mathbf{b}_d), \tag{4.13}$$

where the parameters include $\mathbf{W}_c \in \mathbb{R}^{256\times(K*256)}$, $\mathbf{b}_c \in \mathbb{R}^{256\times1}$, $\mathbf{W}_d \in \mathbb{R}^{2\times512}$, and $\mathbf{b}_d \in \mathbb{R}^{2\times1}$. 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.4.2(c)). We superimpose the scores s_o of all the edge boxes, and then 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.4.2(e). We apply the *context-free text detector* (CFD) in Fig.4.7, using a multi-scale sliding window approach with a size of 36×36 and a stride of 1 pixel. The scale varies 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.4.6, 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.

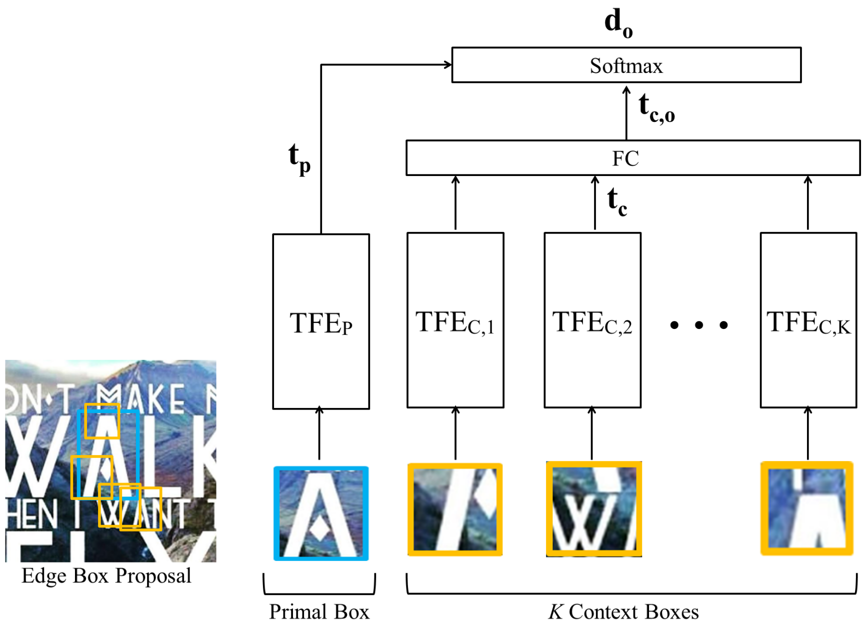


Figure 4.5 The illustration of the proposed context-aware detection.

4.2.3 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 illus-

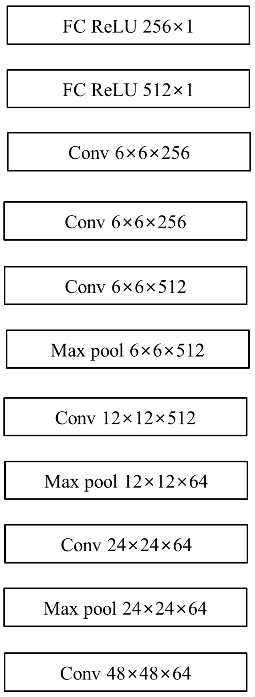


Figure 4.6 The Text Feature Extractor (TFE) network components.

trated in Fig.4.2(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. After obtaining the final text response map by pixelwise min-pooling operation (Fig.4.2(f)), we binarize the response map $M(i, j)$ by Otsu thresholding (Fig.4.2(g)). The positive text response regions (*i.e.* displayed as white pixels) are refined by *Maximally Stable Extremal Regions* (MSER) detection method [81]. And then, we check several geometric criteria to make sure that the regions are text-likely, including areas, aspect ratios and *Euler numbers* (*i.e.* the total number of objects in the image minus the total number of holes in those objects). To obtain such geometric properties of the regions, we ap-

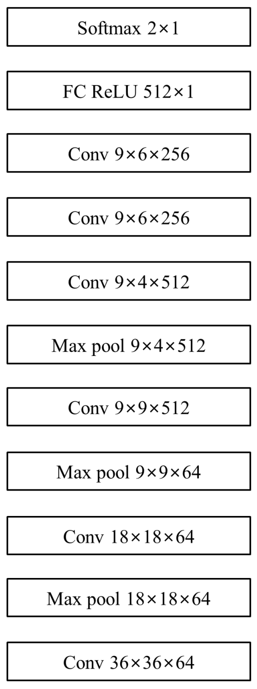


Figure 4.7 The Context-Free Text Detector (CFD).

ply the connected component analysis [17]. If the regions fail to pass all the geometric criteria, they are removed. The refined binary image is shown in Fig.4.2(h).

Otsu Thresholding The Otsu thresholding technique is based on the assumption that the variance of the object's intensity distribution is less distributed, and those of the back ground. The optimal threshold is the value that minimizes the variance within the object and back ground (within class variance) and maximizes the variance between object and back ground (between class variance).

Connected Component Analysis The information obtained through

Connected Component Analysis includes *Area*, *Blob Box*, *Euler Number*, *Extent*, and *Solidity*. Each is defined as follows.

Area : The actual number of pixels in the object.

Bounding Box : The smallest rectangle containing the object.

Euler Number : The total number of objects in the image minus the total number of holes in those objects.

Extent : The ratio of pixels in the object to pixels in the bounding box. It can be computed as the Area divided by the area of the bounding box.

Solidity : The proportion of the pixels in the convex hull that are also in the object

Perimeter : The distance around the edge of the object

Batch Normalization Layer Covariate shift [94] is a phenomenon in machine learning where the features presented to a model change in distribution. In order for learning to succeed in the presence of covariance shift, the model's parameters must be adjusted not just to learn the concept at hand but also to adapt to the changing distribution of the inputs. In deep neural networks, this problem manifests as internal covariance shift [47], where changing the parameters of a layer affects the distribution of the inputs to all layers above it. To deal this, in the training phase, the mean and variance are stored in units of batch, and the correction is made at the time of inference. In addition, training time is shortened by improving the convergence speed of training. The equation is as follows.

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i \tag{4.14}$$

$$\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2 \tag{4.15}$$

$$\hat{x_i} = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \tag{4.16}$$

$$y_i = \gamma\hat{x_i} + \beta \tag{4.17}$$

where the y_i is the output of batch normalization layer after compensation.

Dropout Layer Deep neural nets with a large number of parameters are very powerful machine learning systems. However, overfitting is a serious problem in such networks. Large networks are also slow to use, making it difficult to deal with overfitting by combining the predictions of many different large neural nets at test time. Dropout[96] is a technique for addressing this problem. The key idea is to randomly drop units from the neural network during training. This prevents units from co-adapting too much. During training, dropout samples from an exponential number of different "thinned" networks. At test time, it is easy to approximate the effect of averaging the predictions of all these thinned networks by simply using a single unthinned network that has smaller weights. This significantly reduces overfitting and gives major improvements over other regularization methods.

4.2.4 Text Detection Network Variants

Neural network configuration can be done in various ways. In this section, we will discuss how to construct a network that has seven layers like LeNet not

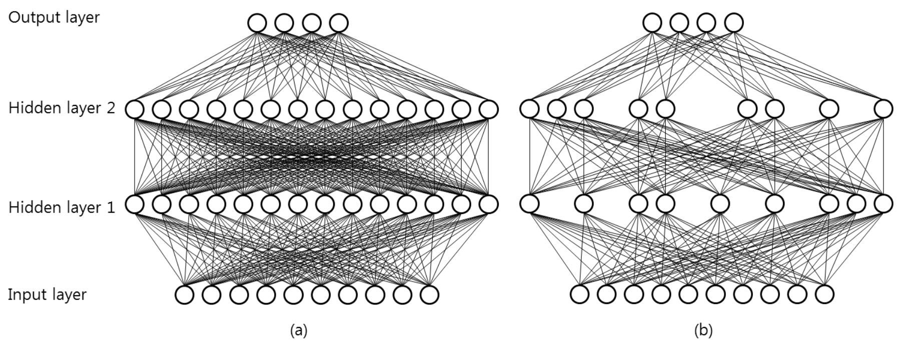


Figure 4.8 Dropout Network

VGG like network.

LeNet *LeNet* is the typical Convolutional Neural Network for *MNIST Digit Recognition* coined by Yann LeCun *et al.* [63]. LeNet evolved into LeNet1, LeNet2, ..., LeNet5. Typically it consists of Convolution, Pooling and Fully Connected layers. The convolution layer is useful for finding patterns in local pixels. You can find a global pattern by constructing multiple layers. This pattern is called a feature or feature map. When constructing multiple layers, put a pooling layer to maintain spatial invariancy. The final layer constitutes a fully connected neural network suitable for object recognition. Let the input as X. In this work, we construct 3 convolutiona layers, 2 max pooling layers

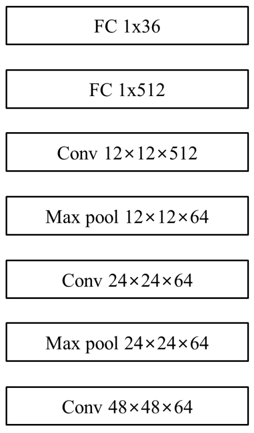


Figure 4.9 The diagram of LeNet.

and 2 fully connected layers.

$$\mathbf{le}_1 = \mathrm{ReLU}(\mathbf{W}_{le1}\mathbf{x}_1), \tag{4.18}$$

$$\mathbf{le}_2 = \mathrm{Conv}(\mathbf{W}_{le2}\mathbf{le}_1), \tag{4.19}$$

$$\mathbf{le}_3 = \mathrm{ReLU}(\mathbf{W}_{le3}\mathbf{le}_2), \tag{4.20}$$

$$\mathbf{le}_4 = \mathrm{Conv}(\mathbf{W}_{le4}\mathbf{le}_3), \tag{4.21}$$

$$\mathbf{le}_5 = \mathrm{ReLU}(\mathbf{W}_{le5}\mathbf{le}_4), \tag{4.22}$$

$$\mathbf{le}_6 = \mathrm{ReLU}(\mathbf{W}_{le6}\mathbf{le}_5), \tag{4.23}$$

$$\mathbf{le}_7 = \mathrm{ReLU}(\mathbf{W}_{le7}\mathbf{le}_6) \tag{4.24}$$

The Max Out Network. When configuring the text detection network, we try to connect the context boxes to the max out network instead of the fully connected network. The difference between these two networks is that the fully connected layer is affected by the order of the context boxes, and the max out layer has the advantage of not affecting connection order. However, only one

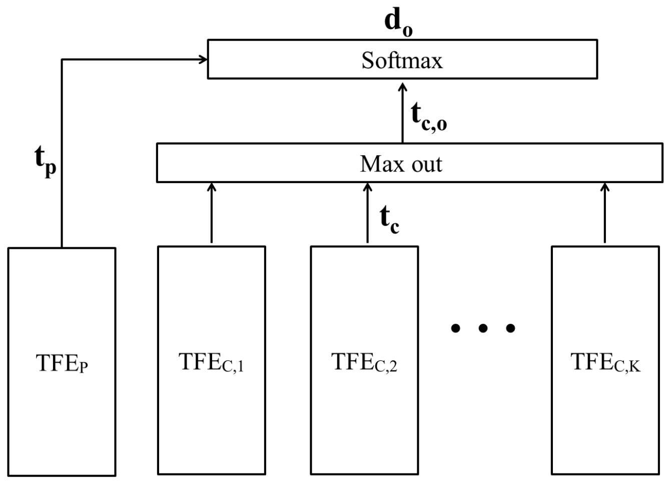


Figure 4.10 The illustration of the proposed maxout detection network

contex box feature should be used among several context boxes, which may cause performance degradation. The max out network equation is as follows:

$$\mathbf{t}_c = \mathbf{vgg}_{out} \tag{4.25}$$

$$\mathbf{t}_{c,o} = \text{ReLU}(\mathbf{W}_c \times \text{MaxOut}(\mathbf{t}_c) + \mathbf{b}_c), \tag{4.26}$$

$$\mathbf{d}_o = \text{softmax}(\mathbf{W}_d[\mathbf{t}_p \parallel \mathbf{t}_{c,o}] + \mathbf{b}_d), \tag{4.27}$$

4.3 Context-Aware Word Recognition

We 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 $V = 3,202$.

Once we obtain a set of textline boxes, we perform the word recognition to find out what words individual textline boxes contain. 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.

To this end, we design a context-aware convolutional recurrent network model for word recognition as shown in Fig.4.11. It consists of three core components of bias networks for context, character recognition networks, and bidirectional recurrent networks for word recognition.

Latent Dirichlet Allocation. Latent Dirichlet allocation (LDA), a generative probabilistic model for collections of discrete data such as text corpora. In addition, it is unsupervised topic modeling, which is a way to find a topic in a document. LDA is a three-level hierarchical model, in which each item of a collection is modeled as a finite mixture over an underlying set of topics. Each topic is, in turn, modeled as an infinite mixture over an underlying set of topic probabilities. In the context of text modeling, the topic probabilities provide an explicit representation of a document. LDA has several advantages: first, the document is a mixture of topics; the other is a probability model. This can be used in combination with other areas that utilize existing probability model models. Another characteristic of the LDA is the generative attribute. After a topic modei is created, a word can be generated by the topic model

ZCA Whitening The data has its principal axes. After whitening the data, the distribution looks round and it also looks rotated. After whitening the distribution looks round and it's oriented in the same way as originally. One can get from PCA whitened data to ZCA whitened data by rotating with principal axis. "ZCA" means "zero phase component analysis". When applied to bunch of images, the main axis looks like a Fourier component of increasing frequency. So they are very *global.* On the other hand, the ZCA transform row looks very *local.* This means that ZCA tries to convert the data as small as possible, so that each row is closer to the original underlying function. This is possible because the natural image correlations are mostly very local.

4.3.1 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 can represent the semantic meaning 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 then use the *Latent Dirichlet Allocation* (LDA) [6, 73] to encode the metadata by topic vectors. Thus, we implicitly discover the topic distribution of the image from its metadata, and assign weights on the candidate words according to topic distribution. We denote the topic distribution of metadata by $\mathbf{v}_m \in \mathbb{R}^D$, where D is the topic dimension (*e.g.* $D = 128$). Then $\mathbf{v}_m$ is fed into the context bias network, consisting of three fully-connected layers with ReLU activation, and one softmax layer, as shown in Fig.4.12.

$$\mathbf{b}_1 = \mathrm{ReLU}(\mathbf{W}_{v1}\mathbf{v}_m), \tag{4.28}$$

$$\mathbf{b}_2 = \mathrm{ReLU}(\mathbf{W}_{v2}\mathbf{b}_1), \tag{4.29}$$

$$\mathbf{b}_3 = \mathrm{ReLU}(\mathbf{W}_{v3}\mathbf{b}_2), \tag{4.30}$$

$$\mathbf{b} = \mathrm{softmax}(\mathbf{W}_{v4}\mathbf{b}_3), \tag{4.31}$$

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 [39]. We apply the dropout regularization with a rate of 0.5 to the three fully-connected layers. 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 $\mathbf{b}$.

4.3.2 Recurrent Word Recognition Network

. Fig.4.11 illustrates the proposed context-aware word recognition network. We first resize the input textline box to have its minimum 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.4.13. 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 $T = 24$. 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 instead of the conventional ones, because it is more reasonable to 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

$$\mathbf{h}_{t,f} = \mathrm{ReLU}(\mathbf{W}_{ph,f}\mathbf{p}_t + \mathbf{W}_{hh,f}\mathbf{h}_{t-1,f} + \mathbf{b}), \tag{4.32}$$

$$\mathbf{h}_{t,b} = \mathrm{ReLU}(\mathbf{W}_{ph,b}\mathbf{p}_t + \mathbf{W}_{hh,b}\mathbf{h}_{t+1,b} + \mathbf{b}) \tag{4.33}$$

$$\mathbf{y} = [\mathbf{h}_{1,f}, \mathbf{h}_{2,f}, ..., \mathbf{h}_{T,f} \parallel \mathbf{h}_{1,b}, h_{2,b}, ..., \mathbf{h}_{T,b}], \tag{4.34}$$

$$\mathbf{w} = \mathrm{softmax}(\mathbf{W}_w\mathbf{y} + \mathbf{b}_w) \tag{4.35}$$

where $\mathbf{b}$ is the topic bias computed in Eq.(4.28). 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 $\mathrm{argmax}_{v\in V}\,\mathbf{w}(v)$.

When we consider the word recognition without context bias network shown in Fig.(4.14). We replace the b in Eq.(4.32) and b_var. This formulation is represented by

$$\mathbf{h}_{t,f} = \mathrm{ReLU}(\mathbf{W}_{ph,f}\mathbf{p}_t + \mathbf{W}_{hh,f}\mathbf{h}_{t-1,f} + \mathbf{b}_{var}), \tag{4.36}$$

$$\mathbf{h}_{t,b} = \mathrm{ReLU}(\mathbf{W}_{ph,b}\mathbf{p}_t + \mathbf{W}_{hh,b}\mathbf{h}_{t+1,b} + \mathbf{b}_{var}) \tag{4.37}$$

$$\mathbf{y} = [\mathbf{h}_{1,f}, \mathbf{h}_{2,f}, ..., \mathbf{h}_{T,f} \parallel \mathbf{h}_{1,b}, h_{2,b}, ..., \mathbf{h}_{T,b}], \tag{4.38}$$

$$\mathbf{w} = \mathrm{softmax}(\mathbf{W}_w\mathbf{y} + \mathbf{b}_w) \tag{4.39}$$

where $\mathbf{b}_{var} \in \mathbb{R}^D$ is the bias for node. The rest of setting is the same as the previous recognition network.

Variable depth for Recurrent neural network We check the effect the depth for Recurrent neural network. First, directional recurrent neural network word recognition formula as follows:

$$\mathbf{h}_f = \mathrm{ReLU}(\mathbf{W}_{ph}\mathbf{p}_t + \mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{b}), \tag{4.40}$$

$$\mathbf{y} = [\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_T], \tag{4.41}$$

$$\mathbf{w} = \mathrm{softmax}(\mathbf{W}_w\mathbf{y} + \mathbf{b}_w) \tag{4.42}$$

The double bidirectional neural network formula as follows:

$$\mathbf{h1}_{t,f} = \mathrm{ReLU}(\mathbf{W}_{ph1,f}\mathbf{p}_t + \mathbf{W}_{h1h1,f}\mathbf{h}1_{t-1,f} + \mathbf{b}), \tag{4.43}$$

$$\mathbf{h1}_{t,b} = \mathrm{ReLU}(\mathbf{W}_{ph1,b}\mathbf{p}_t + \mathbf{W}_{h1h1,b}\mathbf{h}1_{t+1,b} + \mathbf{b}), \tag{4.44}$$

$$\mathbf{h2}_{t,f} = \mathrm{ReLU}(\mathbf{W}_{ph2,f}\mathbf{p}_t + \mathbf{W}_{h2h2,f}\mathbf{h}2_{t-1,f} + \mathbf{b}), \tag{4.45}$$

$$\mathbf{h2}_{t,b} = \mathrm{ReLU}(\mathbf{W}_{ph2,b}\mathbf{p}_t + \mathbf{W}_{h2h2,b}\mathbf{h}2_{t+1,b} + \mathbf{b}), \tag{4.46}$$

$$\mathbf{y} = [\mathbf{h}2_{1,f}, \mathbf{h}2_{2,f}, ..., \mathbf{h}2_{T,f} \parallel \mathbf{h}2_{1,b}, h2_{2,b}, ..., \mathbf{h}2_{T,b}], \tag{4.47}$$

$$\mathbf{w} = \mathrm{softmax}(\mathbf{W}_w\mathbf{y} + \mathbf{b}_w) \tag{4.48}$$

The double bidirectional neural network architecture is shown in Fig.(**??**)

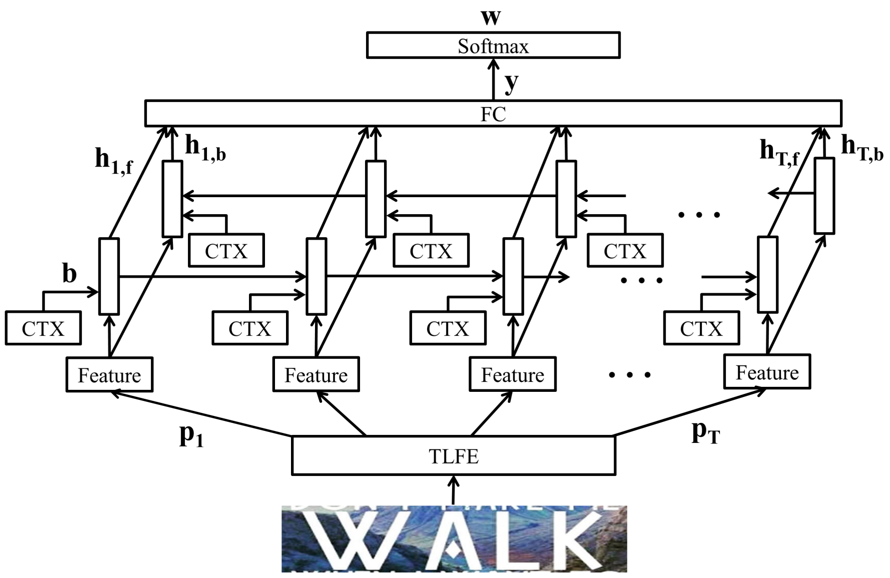


Figure 4.11 The illustration of the proposed context-aware recognition network.

Zero Padding The input image are rescaled as the height of it is 32. The warped image has the same aspect ratio as the ratio of the original image. If the width of warped image is shorter than the input width of character feature extractor, we pad the remaining buffer with zero by following the method in [46].

4.3.3 Recognition Network Variant

Fully Connected bias network. When word recognition is performed, we will discuss how to use the context bias network at every step. This method has been actively studied in the field of natural language processing. When language modeling is used to generate each sentence for given words, if the fully connected network is used, it is difficult to implement due to the vanishing

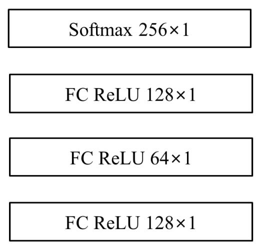


Figure 4.12 Context bias network (CTX in the model),

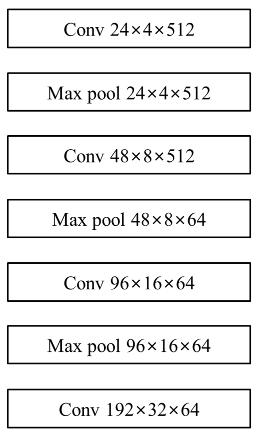


Figure 4.13 The textline feature extractor (TLFE)

gradient problem. To resolve this problem, a recurrent neural network was used. In particular, the vanishing gradient problem was solved by inputting context information to help modeling at every step. This problem is expected when constructing a fully connected network, but we will experimentally verify

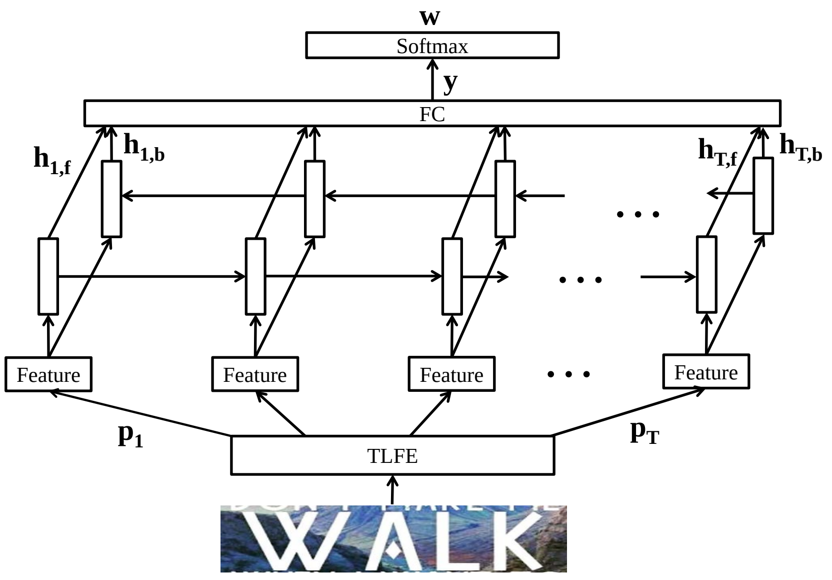


Figure 4.14 The illustration of the proposed context-aware recognition network without context bias network.

it. The formula for the network is as follows.

$$\mathbf{h}_{t,f} = \mathrm{ReLU}(\mathbf{W}_{ph,f}\mathbf{p}_t + \mathbf{W}_{hh,f}\mathbf{h}_{t-1,f}), \tag{4.49}$$

$$\mathbf{h}_{t,b} = \mathrm{ReLU}(\mathbf{W}_{ph,b}\mathbf{p}_t + \mathbf{W}_{hh,b}\mathbf{h}_{t+1,b}) \tag{4.50}$$

$$\mathbf{c}_h = [\mathbf{h}_{1,f}, \mathbf{h}_{2,f}, ..., \mathbf{h}_{T,f} \parallel \mathbf{h}_{1,b}, h_{2,b}, ..., \mathbf{h}_{T,b}], \tag{4.51}$$

$$\mathbf{y} = \mathbf{W}_y\mathbf{c}_h + \mathbf{b}_h \tag{4.52}$$

$$\mathbf{w} = \mathrm{softmax}(\mathbf{W}_w\mathbf{y} + \mathbf{b}_w) \tag{4.53}$$

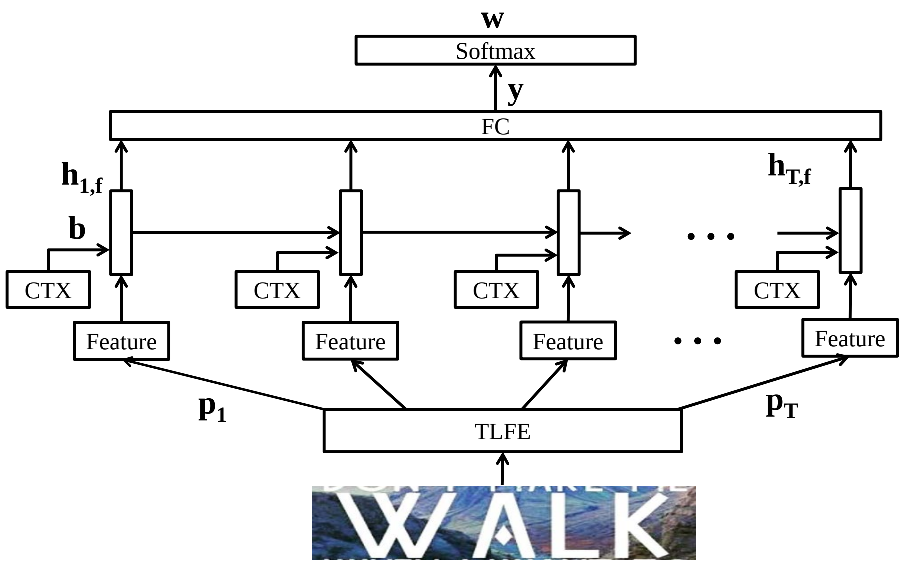


Figure 4.15 The illustration of the directional recurrent neural word recognition network.

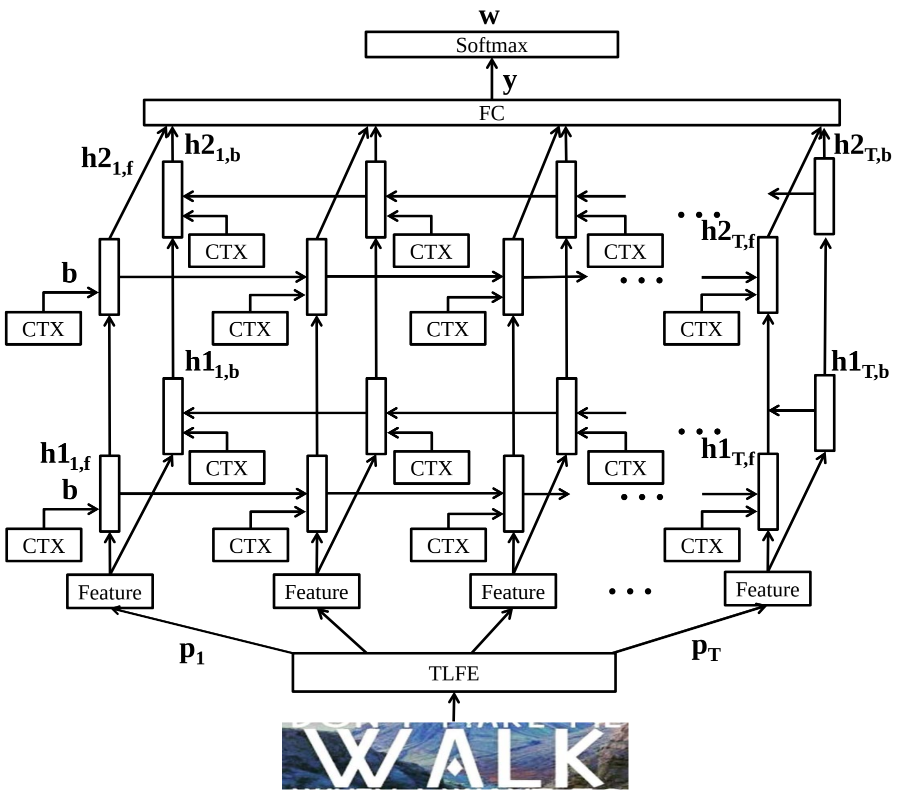


Figure 4.16 The illustration of the double bidirectional recurrent neural word recognition network.

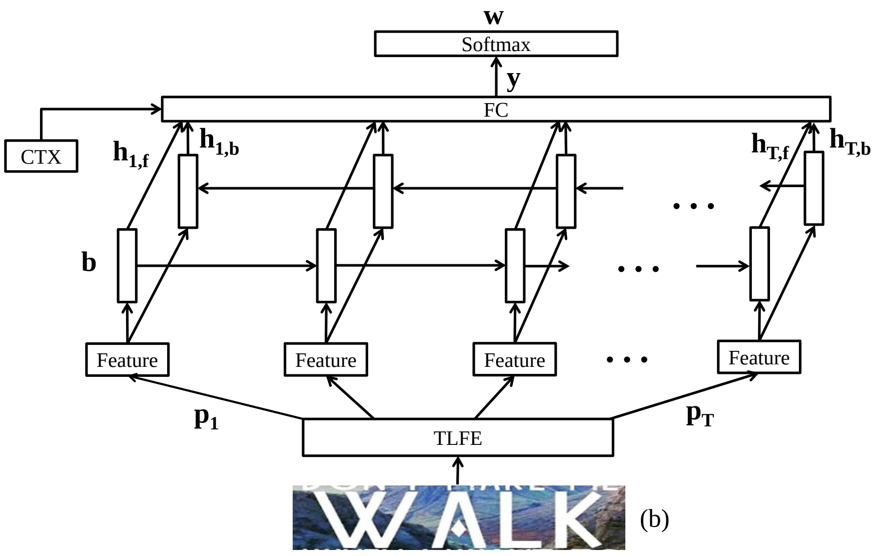


Figure 4.17 The illustration of the context-aware recognition network for fully connected bias network

Chapter 5

Experiments

We defer more results and details to the supplementary, including dataset collection, experimental setting, and a demo code. We plan to make public our source code and datasets.

5.1 Dataset

We collect three datasets of text embedded images from Instagram, Flickr and Pinterest. The Instagram dataset consists of 2,233 images with about $22K$ words in total, the Flickr dataset comprises 424 images with about 4,262 words, and Pinterest data set consists of 206 images We also crawling the metadata associated with images if available, including tags, comments and title. 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.

Instagram Dataset. We collect general users' text embedded images from

Instagram as follows. We query the keyword *caption* from the Instagram build-in search engine, and manually select the 2,233 images, which contain about $22K$ words in total (*i.e.* about 10 words per image on average). Then, we let human labelers to annotate words that can be recognizable and are longer than a single characters, using the bounding box annotation toolbox[1]. The Instagram images are challenging that they show a large variation of fonts and sizes of words even in a single image.We also crawl the metadata associated with images, including tags, comments and title if available.We randomly split the Instagram dataset into 1,786 training and 447 test images. We show the selected examples in Fig.5.1

Flickr Dataset. We create ohter new dataset by crawling text embedded images from Flickr. We query the keyword *funny* and *caption* from the Flickr search engine. We collect 424 images containing 4,262 words, and their associated metadata. We split the image set into 213 training and 211 test images. We also use the same annotation toolbox. We show the selected examples in Fig.5.2

Pinterest Dataset. We create the other new dataset by crawling text embedded images from Pinterest. We query the keyword *captions* and *quotes* from the Pinterest search engine. We collect 206 images, and their associated metadata. We split the image set into 103 training and 103 test images. We also use the same annotation toolbox. We show the selected examples in Fig.5.3

Metadata Pruning We mentioned earlier that one-hot encoding of metadata is used as input to the context bias network via the LDA topic model.

[1] http://www.ipb.uni-bonn.de/html_pages_software/annotation-tool/.

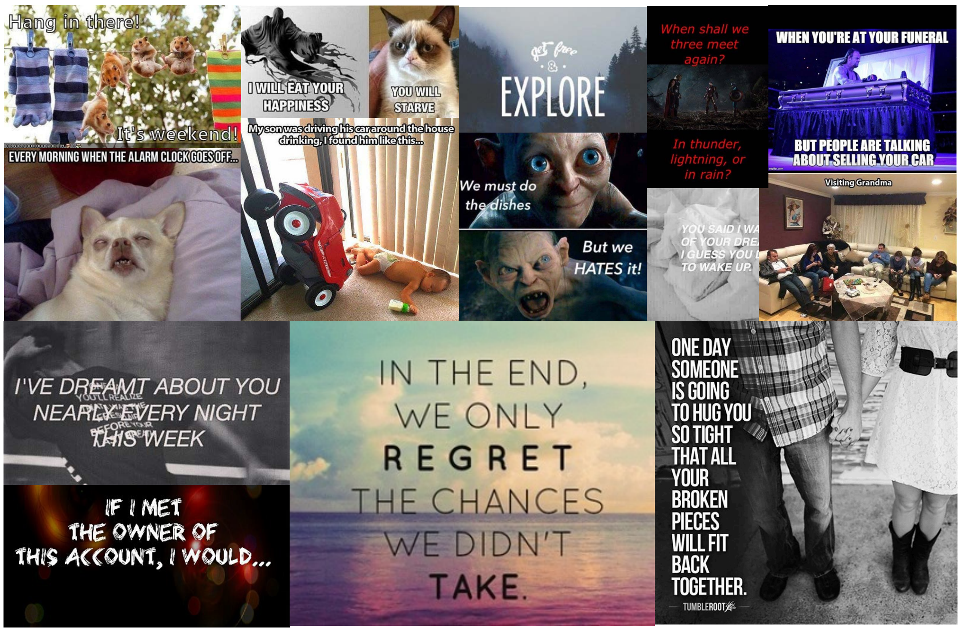

Hang in there!
It's weekend!
I WILL EAT YOUR HAPPINESS
YOU WILL STARVE
EXPLORE
When shall we three meet again?
In thunder, lightning, or in rain?
WHEN YOU'RE AT YOUR FUNERAL
BUT PEOPLE ARE TALKING ABOUT SELLING YOUR CAR
EVERY MORNING WHEN THE ALARM CLOCK GOES OFF...
My son was driving his car around the house drinking, I found him like this...
We must do the dishes
But we HATES it!
Visiting Grandma
I'VE DREAMT ABOUT YOU NEARLY EVERY NIGHT THIS WEEK
IN THE END, WE ONLY REGRET THE CHANCES WE DIDN'T TAKE.
ONE DAY SOMEONE IS GOING TO HUG YOU SO TIGHT THAT ALL YOUR BROKEN PIECES WILL FIT BACK TOGETHER.
IF I MET THE OWNER OF THIS ACCOUNT, I WOULD...

Figure 5.1 Examples of text embeded online images for Instagram.

However, when topic modeling is performed using all crawled metadata, the distribution structure of the main words is not revealed due to the word having a high frequency, or it can be misinterpreted due to the internet informal words. To prevent this, we prune the metadata if those are NLTK stop words or infomal internet words, *i.e. loveit*, *Featuremeinstagood*, *spam4spam*, *likeforfollow*, *follow4follow*, *like4like*, and *followme*.

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.

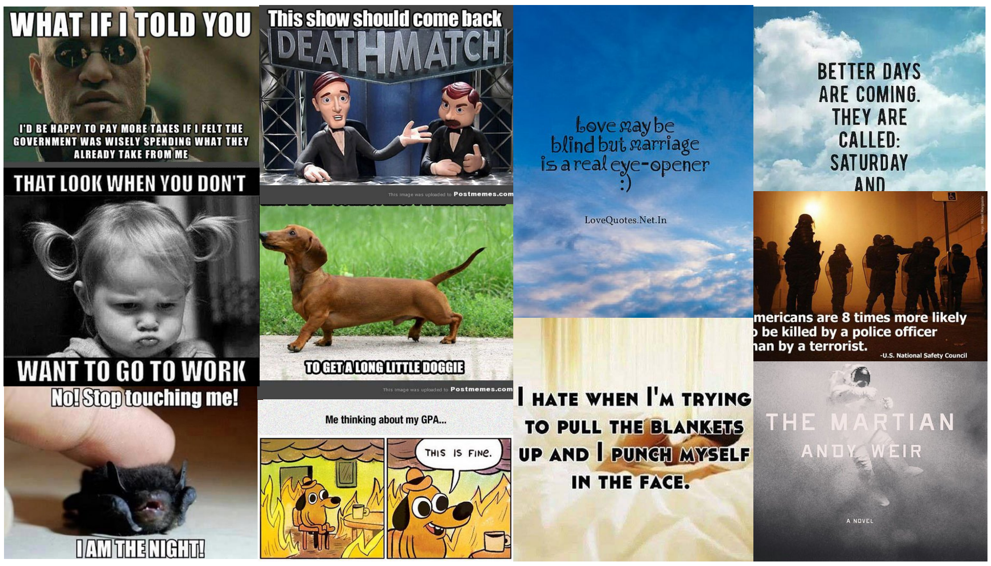

WHAT IF I TOLD YOU
I'D BE HAPPY TO PAY MORE TAXES IF I FELT THE GOVERNMENT WAS WISELY SPENDING WHAT THEY ALREADY TAKE FROM ME
This show should come back
DEATHMATCH
Love may be blind but marriage is a real eye-opener :)
LoveQuotes.Net.In
BETTER DAYS ARE COMING. THEY ARE CALLED: SATURDAY AND
THAT LOOK WHEN YOU DON'T
WANT TO GO TO WORK
TO GET A LONG LITTLE DOGGIE
mericans are 8 times more likely be killed by a police officer han by a terrorist.
-U.S. National Safety Council
No! Stop touching me!
I AM THE NIGHT!
Me thinking about my GPA...
THIS IS FINE.
I HATE WHEN I'M TRYING TO PULL THE BLANKETS UP AND I PUNCH MYSELF IN THE FACE.
THE MARTIAN
ANDY WEIR
A NOVEL

Figure 5.2 Examples of text embeded online images for Flickr.

5.2 Experimental Setup

Evaluation. To report the performance, we follow the protocol of ICDAR 2013 competition [53]. 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,

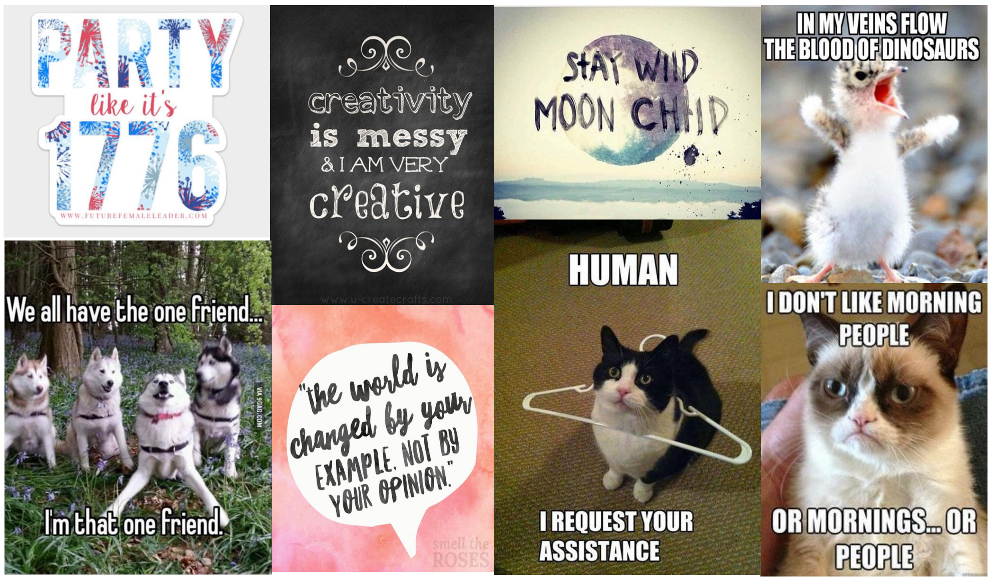

PARTY like it's 1776
WWW.FUTUREFEMALELEADER.COM
creativity is messy & I AM VERY creative
www.u-createcrafts.com
STAY WILD MOON CHILD
IN MY VEINS FLOW THE BLOOD OF DINOSAURS
We all have the one friend...
I'm that one friend.
VIA 9GAG.COM
"the world is changed by your EXAMPLE. NOT BY YOUR OPINION."
smell the ROSES
HUMAN
I REQUEST YOUR ASSISTANCE
I DON'T LIKE MORNING PEOPLE
OR MORNINGS... OR PEOPLE

Figure 5.3 Examples of text embeded online images for Pinterest.

which are based on object detection of pictorial structure [105] (TPS), extremal regions detection [80] (TER), stroke-specific keypoints [7] (TFAST), convolutional neural networks [109](TCNN), and convolutional recurrent networks [92] (TCRNN). For all the baselines, we use publicly available codes provided by the authors. Some baselines do not deal with both detection and recognition; the (TFAST) for detection only and the (TCRNN) is for cropped word recognition only.

5.3 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:

$$d = 1 - \mathbf{t}_{gt}^{T}\mathbf{t}_{p} / \|\mathbf{t}_{gt}\| \|\mathbf{t}_{p}\| \tag{5.1}$$

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.4.6, Fig. 4.7 and Fig.4.13 respectively) shares not only similar structure based on VGGNet, but also the similar objective as character recognizers. We pretrain them using the standard character datasets of Chars74K [15] and IIIT5K [74]. We insert a batch normalization layer [47] 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 [98] 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.4.11 using the synthetic word data[2] of [48], consisting of $8M$ training images for $90K$ lexicons. We use the negative log-likelihood as the cost function, and the Nesterov momentum with a learning rate of $3e^{-3}$ and a momentum of 0.9. We use three batch normalization for 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.

Synthetic Data When we train a Recurrent Word Recognition Neural Network, we can learn many different data by using synthetic data to improve training performance. Also, since many words are generated by people in a program in natural environment, we can get the same type word by program. Againg, considering that much of the text found in natural scenes needs the imaging process variations(e.g. camera, viewpoint, illumination, clutter). To know about that in detail, we summarize as follows: 1. Font rendering – a font is randomly selected from a catalogue of over 1400 fonts downloaded from Google Fonts. The kerning, weight, underline, and other properties are varied randomly from arbitrarily defined distributions. The word is rendered on to the foreground image-layer's alpha channel with either a horizontal bottom text line or following a random curve. 2. Border/shadow rendering – an inset border, outset border or shadow with a random width may be rendered from the foreground. 3. Base coloring – each of the three image-layers are filled with a different uniform color sampled from clusters over natural images. The clusters are formed by k-means clustering the three color components of each

[2]http://www.robots.ox.ac.uk/∼vgg/data/text/.

image of the training datasets. 4. Projective distortion – the foreground and border/shadow image-layers are distorted with a random, full-projective transformation, simulating the 3D world. 5. Natural data blending – each of the image-layers are blended with a randomly-sampled crop of an image from the training datasets of ICDAR 2003[69] and SVT [106]. The amount of blend and alpha blend mode (e.g. normal, add, multiply, burn, max, etc.) is dictated by a random process, and this creates an eclectic range of textures and compositions. The three image-layers are also blended together in a random manner, to give a single output image. 6. Noise – Gaussian noise, blur, and JPEG compression artefacts are introduced to the image. The word samples are generated with a fixed height of 32 pixels, but with a variable width. Since the input to our CNNs is a fixed-size image, the generated word images are rescaled so that the width equals 100 pixels. Although this does not preserve the aspect ratio, the horizontal frequency distortion of image features most likely provides the word-length cues. We also experimented with different padding regimes to preserve the aspect ratio, but found that the results are not quite as good as with resizing. To further detail, refer the paper[48]

5.4 Hyperparameters of Contextual Model

Neural networks have a wide variety of hyperparameters. In this section, several important parameters are selected and their effects are examined such as the number of K, the activation function and the effect of the optimization method.

the effect of K numbers We train the context-aware detection network

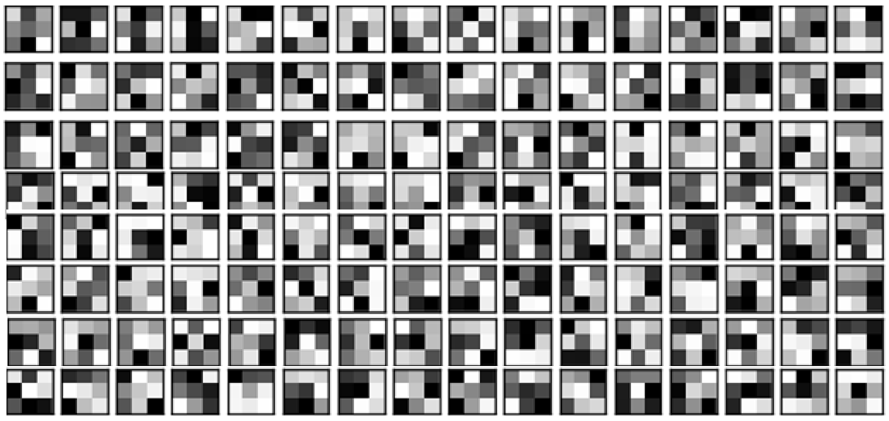


Figure 5.4 The trained weights for context-aware detection network.

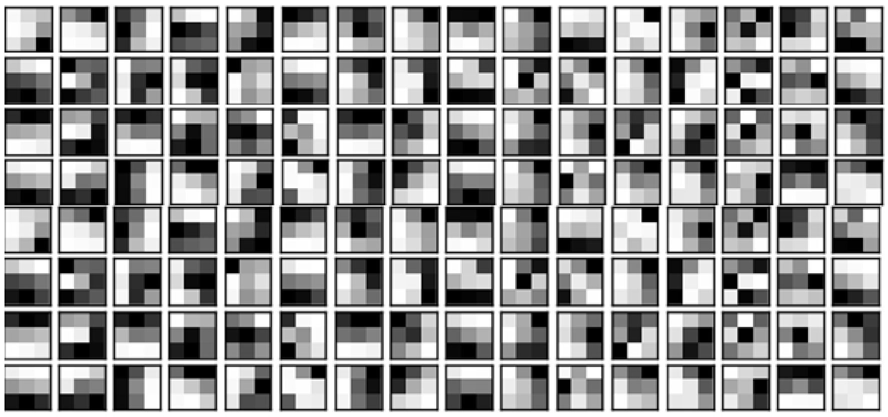


Figure 5.5 The trained weight for context free detection network

for varous number of *K*. The training loss is shown in Fig. 5.7 through Fig. 5.12. The validatiaon loss is minimum for 12, we set the number of K to 12. We set Batch Number to 200, and set learning rate to 0.001, and we use nesterov momentum method as optimization method for this experiment.

the effect of Activation Function We also examine the training loss of the context-aware detection network for varous types of *activation function.* The kinds of activation function is *Rectified Linear Unit(ReLU), sigmoid, tanh,*

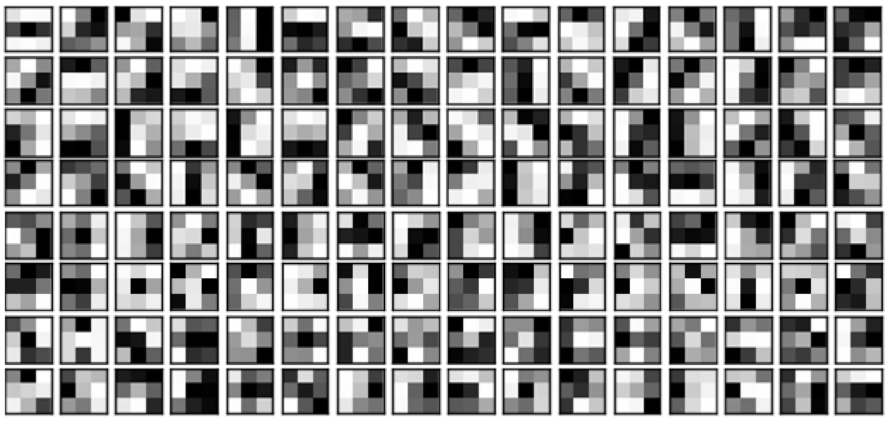


Figure 5.6 The trained weight for textline feature extractor

exponential ReLu. The training loss is shown in Fig. **??** through Fig. 5.17. The validatiaon loss is minimum for ReLU, we set the activatio type to ReLU.

the effect of Optimization We also examine the training loss of the context-aware detection network for varous types of *optimization.* The kinds of optimization is *Nesterov Momentum, Adadelta, Adagrid, Adam.* The training loss is shown in Fig. 5.19 and Fig. 5.21.

the effect of Weight Initialization We also examine the training loss for the various weight initialization. The training/validation loss and validation accuracy is shown in Fig. 5.23 through Fig. 5.28.

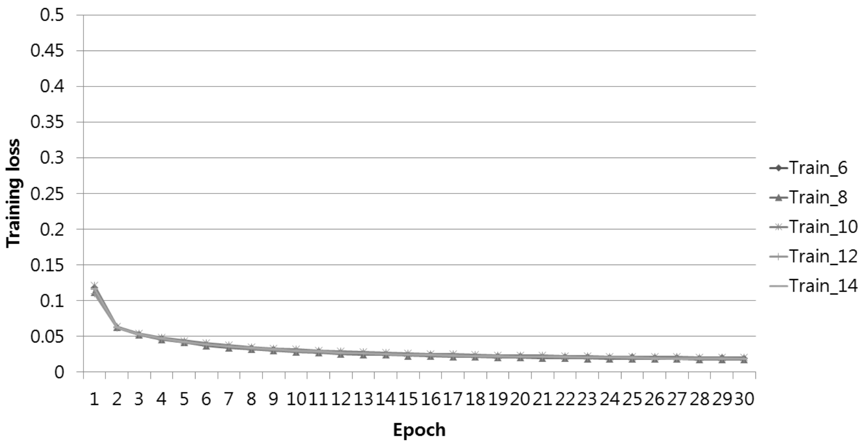


Figure 5.7 The training loss for various number of K(Instagram)

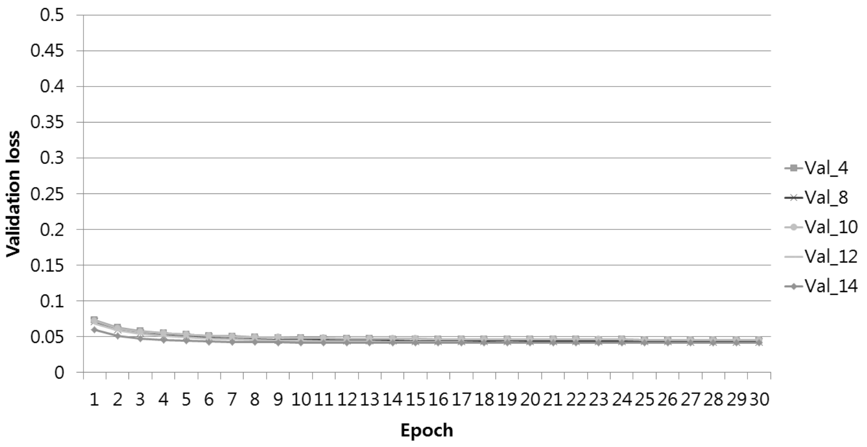


Figure 5.8 The validataion loss for various number of K(Instagram)

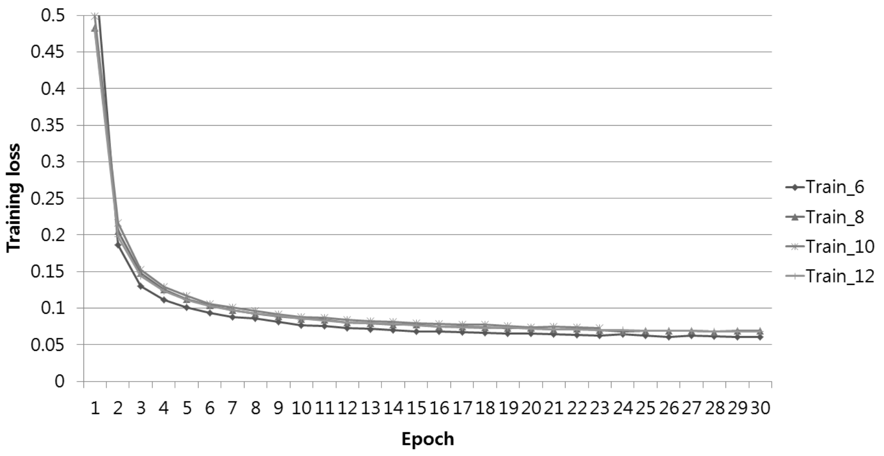


Figure 5.9 The training loss for various number of K(Flickr).

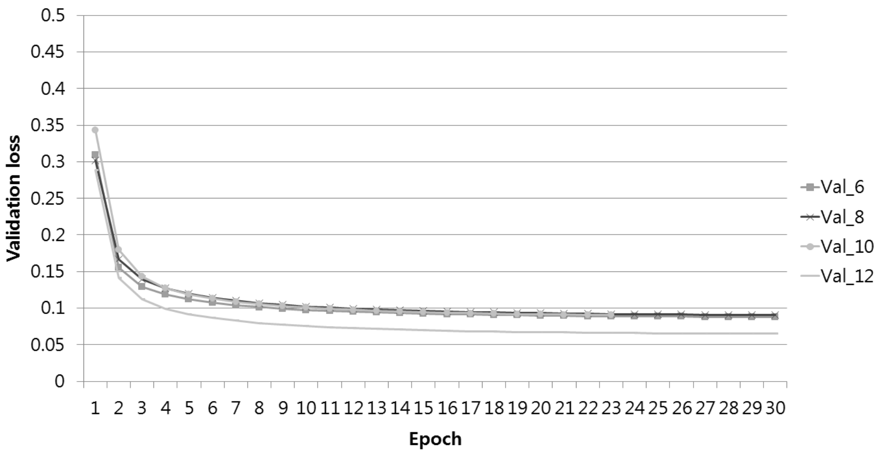


Figure 5.10 The validataion loss for various number of K(Flickr).

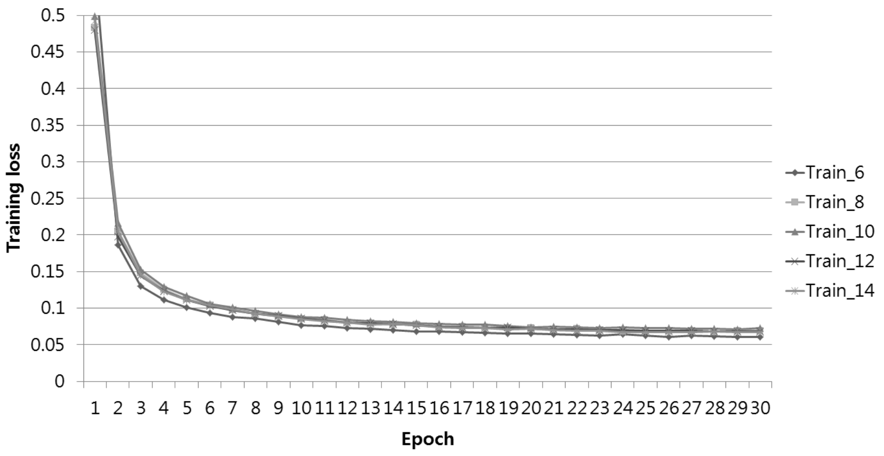


Figure 5.11 The training loss for various number of K(Pinterest).

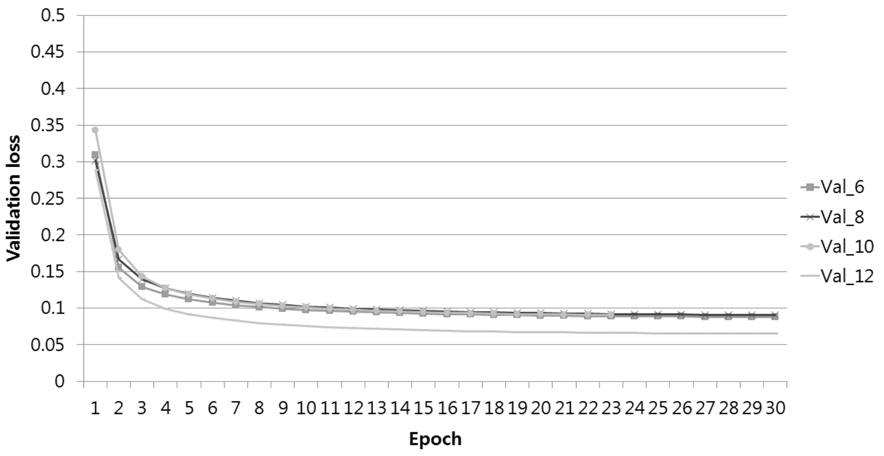


Figure 5.12 The validataion loss for various number of K(Pinterest).

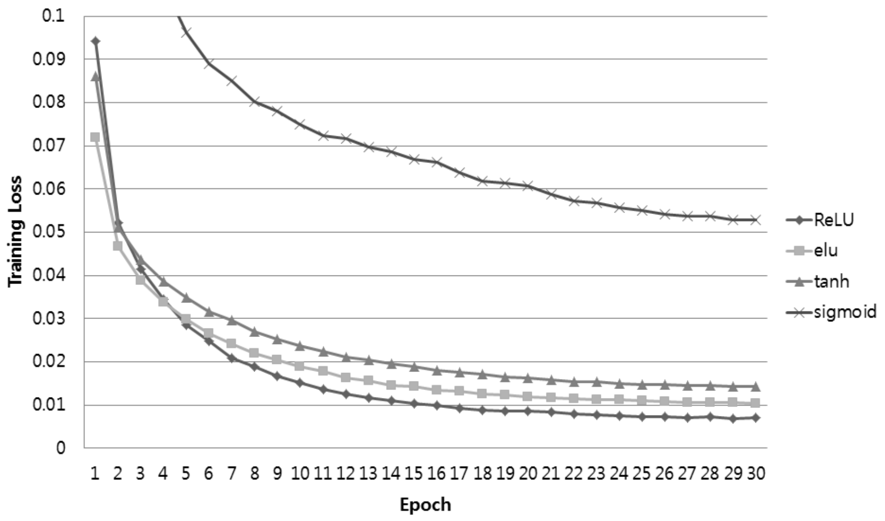


Figure 5.13 The training loss for various activation functions(Instagram).

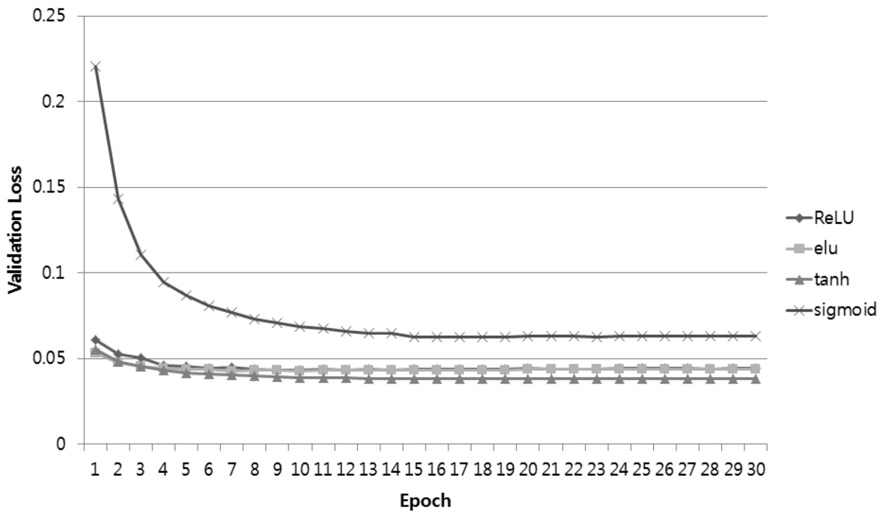


Figure 5.14 The validataion loss for various activation functions(Instagram).

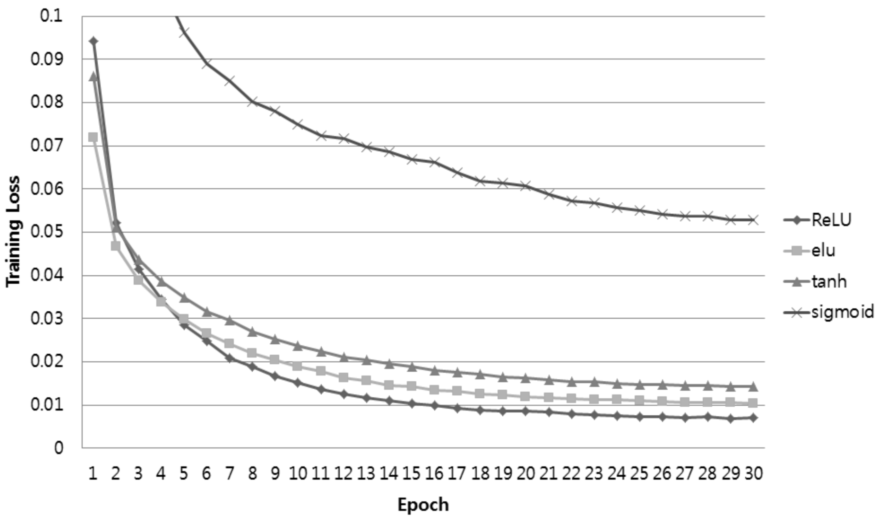


Figure 5.15 The training loss for various activation functions(Flickr).

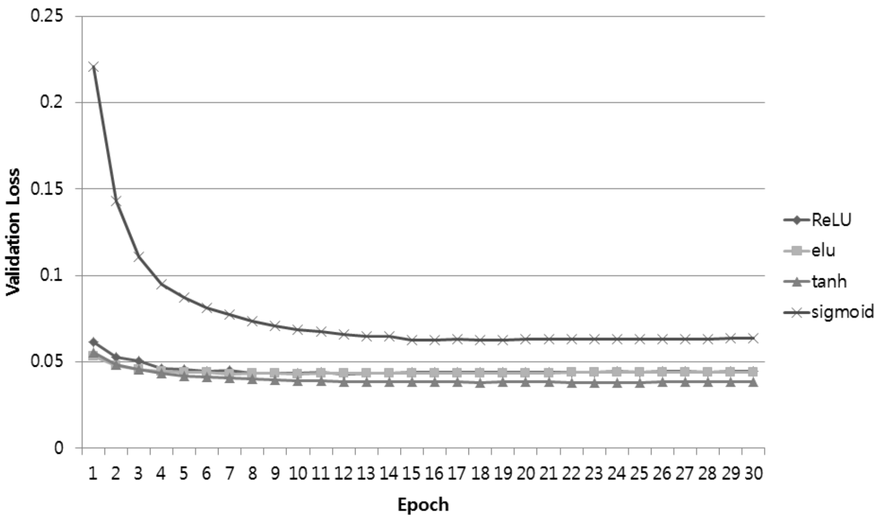


Figure 5.16 The validation loss for various activation functions(Flickr).

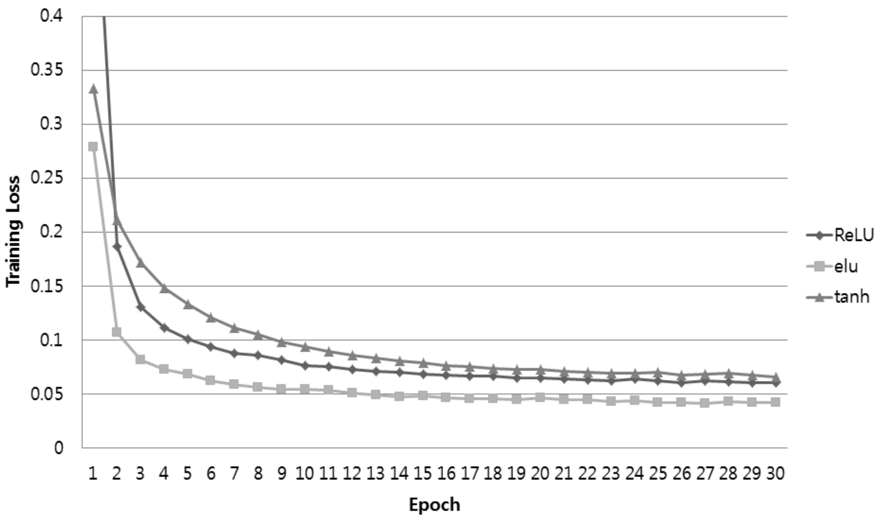


Figure 5.17 The training loss for various activation functions(Pinterest).

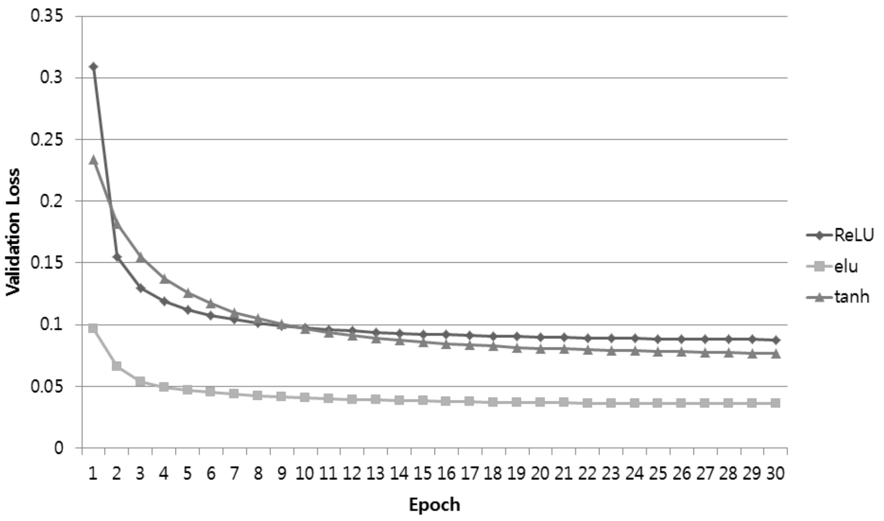


Figure 5.18 The validation loss for various activation functions(Pinterest).

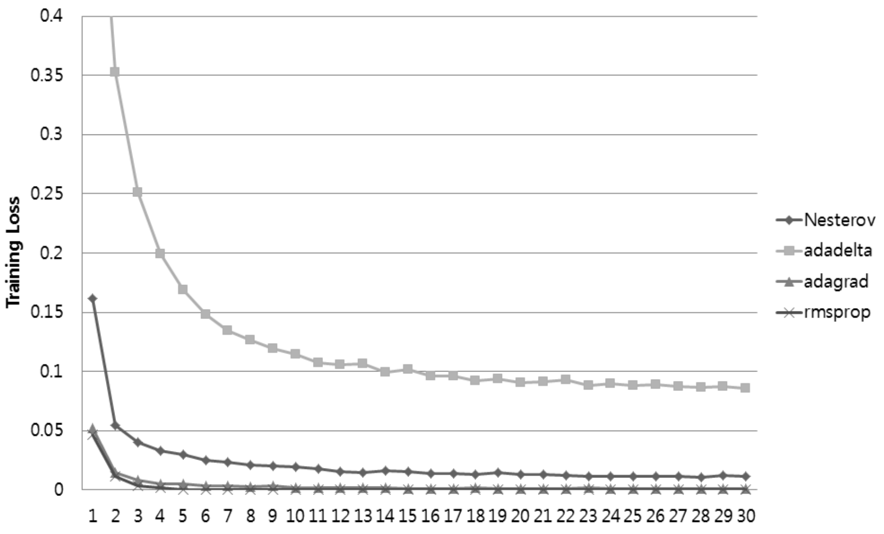


Figure 5.19 The training loss of according to optimization method(Flickr)

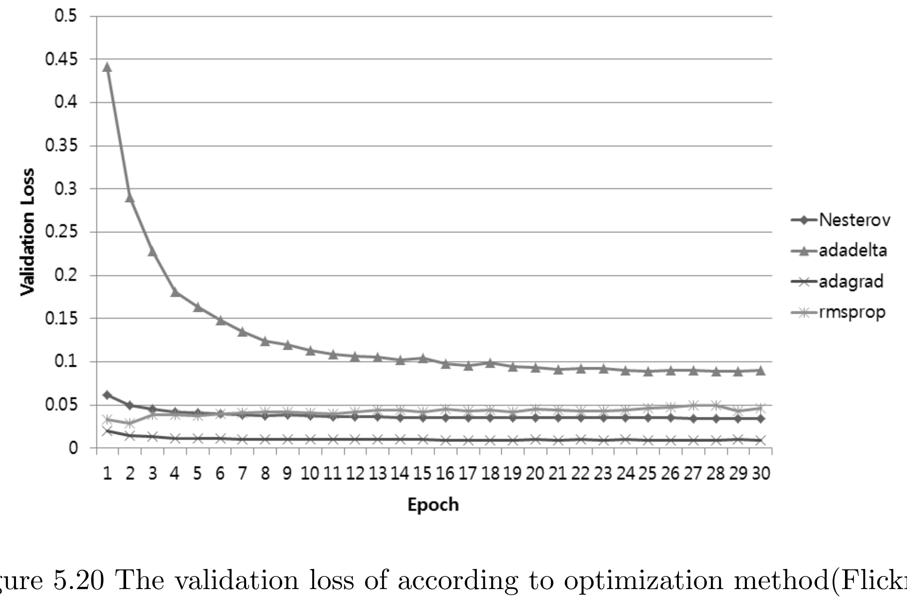


Figure 5.20 The validation loss of according to optimization method(Flickr)

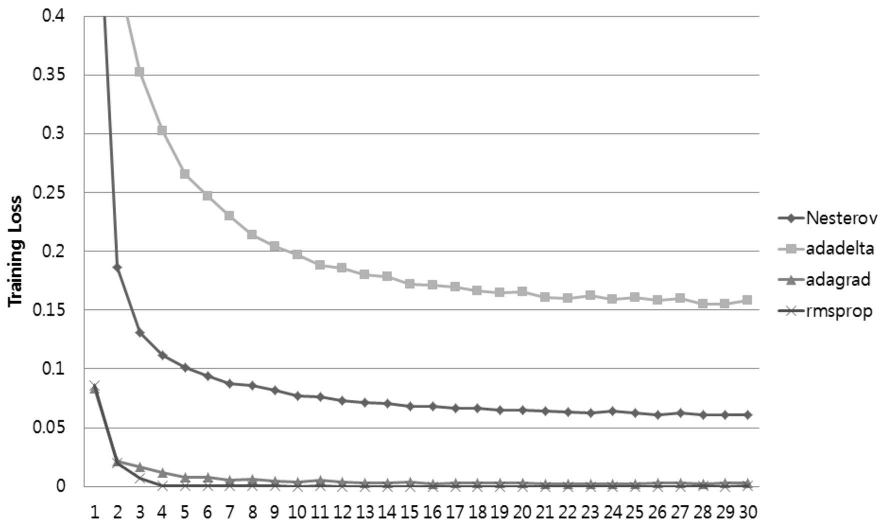


Figure 5.21 The training loss of according to optimization method(Pinterest)

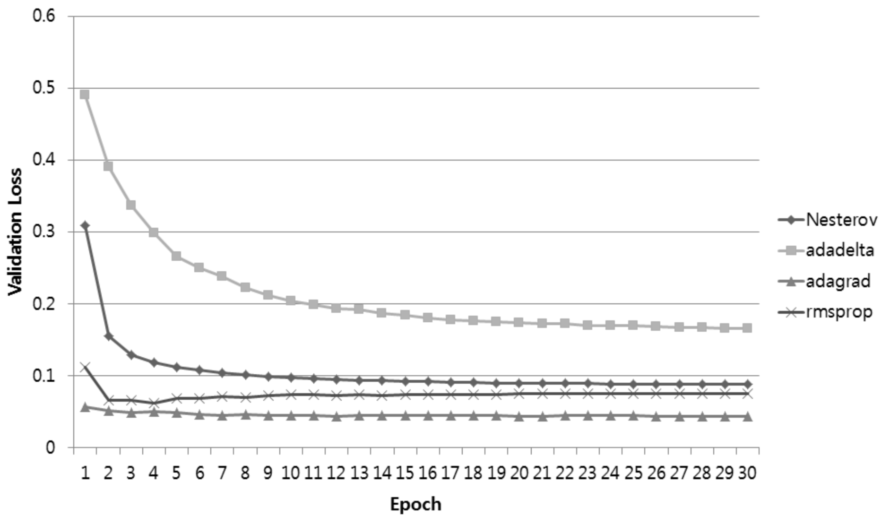


Figure 5.22 The validation loss of according to optimization method(Pinterest)

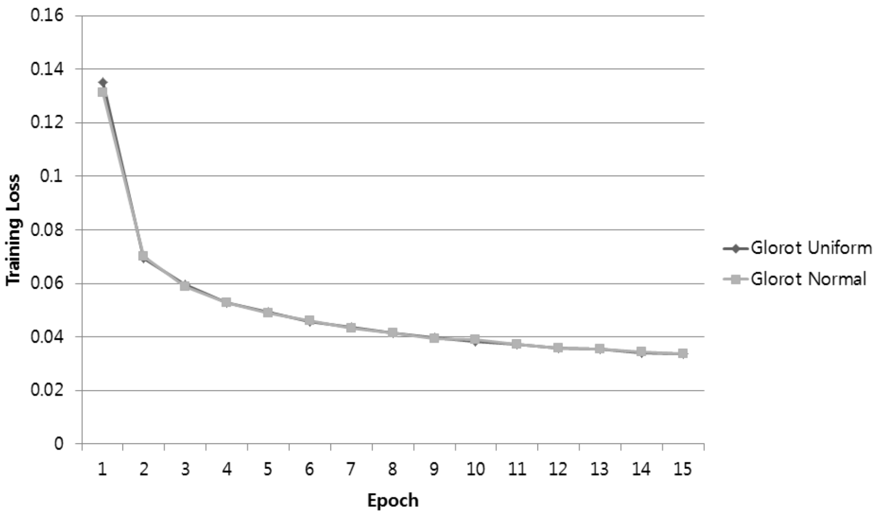


Figure 5.23 The training loss for Glorot Uniform Initialization vs. Glorot Normal Initialization(Instagram)

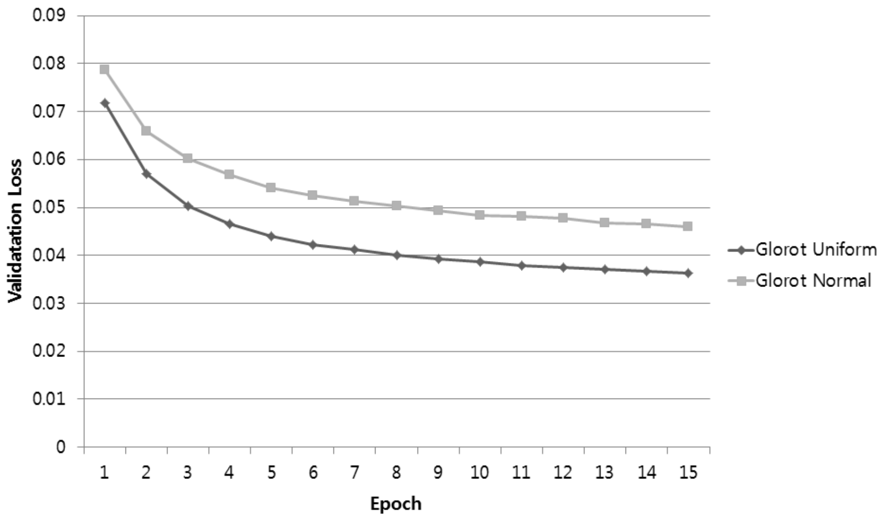


Figure 5.24 The validation loss for Glorot Uniform Initialization vs. Glorot Normal Initialization(Instagram)

5.5 Neural Network Architecture Variants

Comparison VGG and LeNet for TFE In this section, we will examine the results when the Text Feature Extractor part of detection module is composed of LeNet and when it is composed of VGG Net. As shown in Fig.5.29, it is 2.3% higher for VGG Net when the epoch number is 30 for Instagram dataset. The training loss is shown in Fig. 5.30, Fig. 5.33 and Fig. 5.36 for Instagram, Flicrk and Pinterest, respectively.

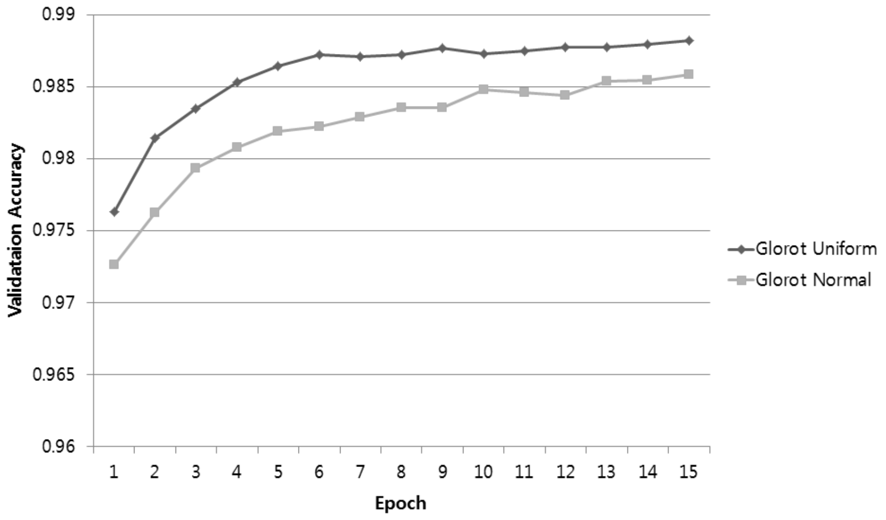


Figure 5.25 The validation accuracy for Glorot Uniform Initialization vs. Glorot Normal Initialization(Instagram)

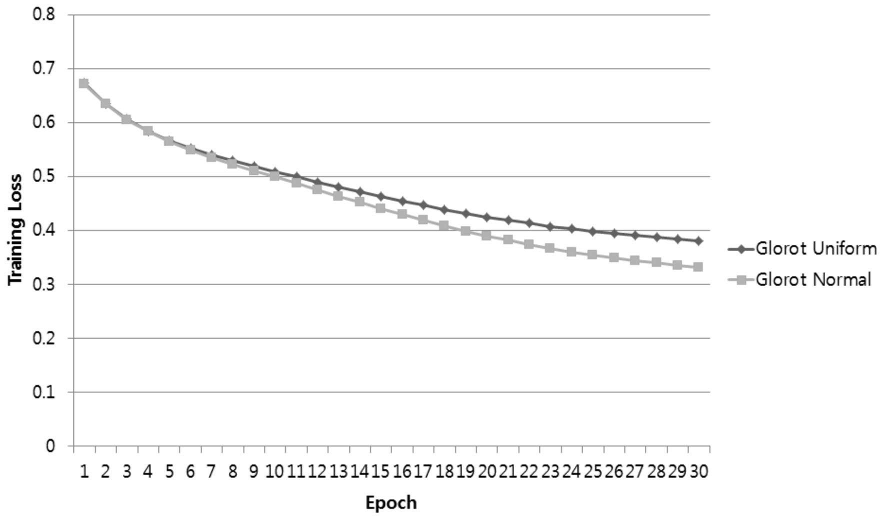


Figure 5.26 The training loss for Glorot Uniform Initialization vs. Glorot Normal Initialization(Flickr)

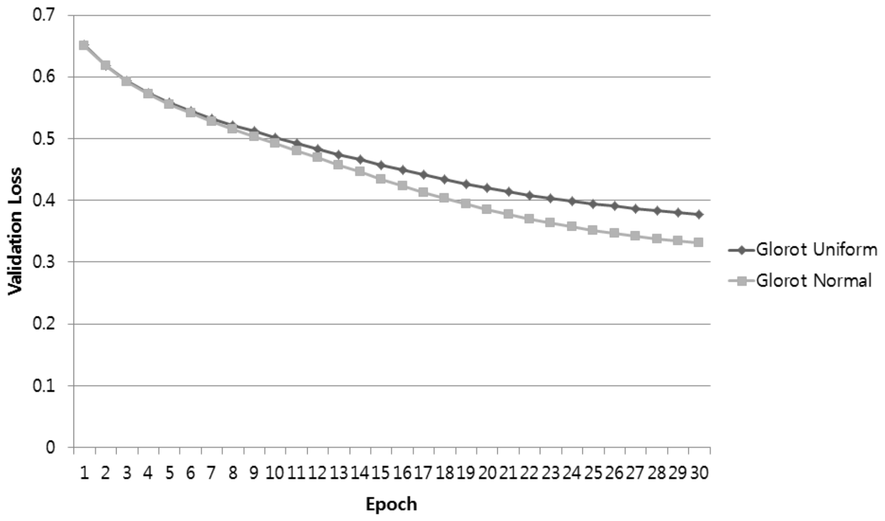


Figure 5.27 The validation loss for Glorot Uniform Initialization vs. Glorot Normal Initialization(Flickr)

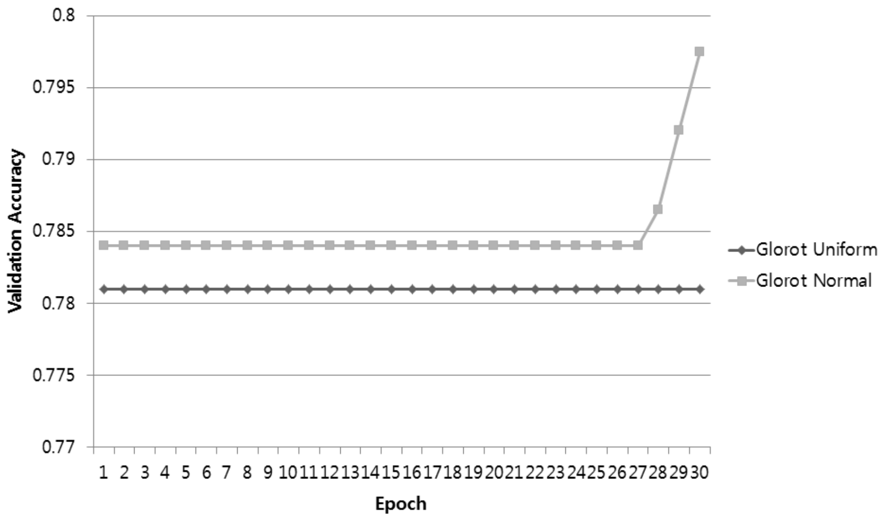


Figure 5.28 The validation accuracy for Glorot Uniform Initialization vs. Glorot Normal Initialization(Flickr)

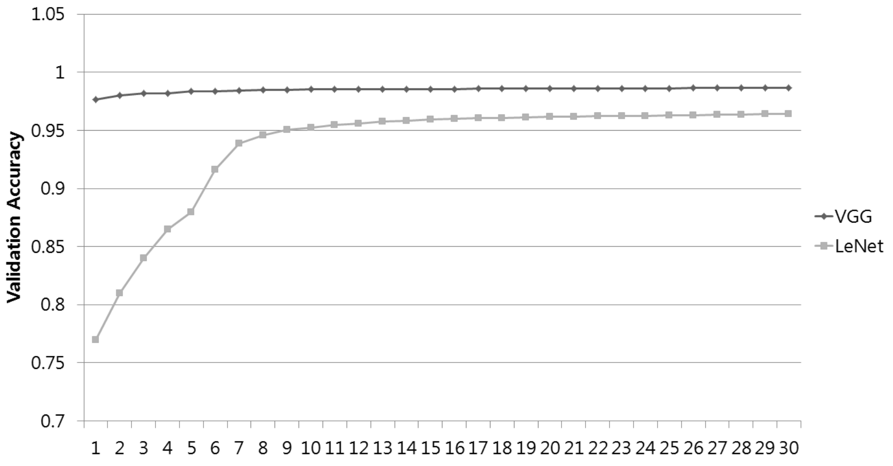


Figure 5.29 The validation score difference between VGG and LeNet for TFE(Instagram)

Comparison Infusing RNN and Fully Connected Architecture

Our RNN inputs contextual information at every step of RNN to mitigate the vanishing gradient problem. Its effectiveness was also reported in different domains; Mikolov *et al.* [73] shows that language modeling performance is improved by continuous input of real-valued vectors to every step of RNN models. We also tested a variant as the context is fed into the fully-connected layer and observed a decrease of validate accuracy by 6.7%p.

5.6 Results

We summarize the results of three tasks of text detection, cropped-word recognition, and end-to-end recognition in Table 5.1 through Table 5.9. We test two variants of our CTSN with or without the context to quantize the performance gain by the use of context. We observe that the context indeed helps enhance the performance.

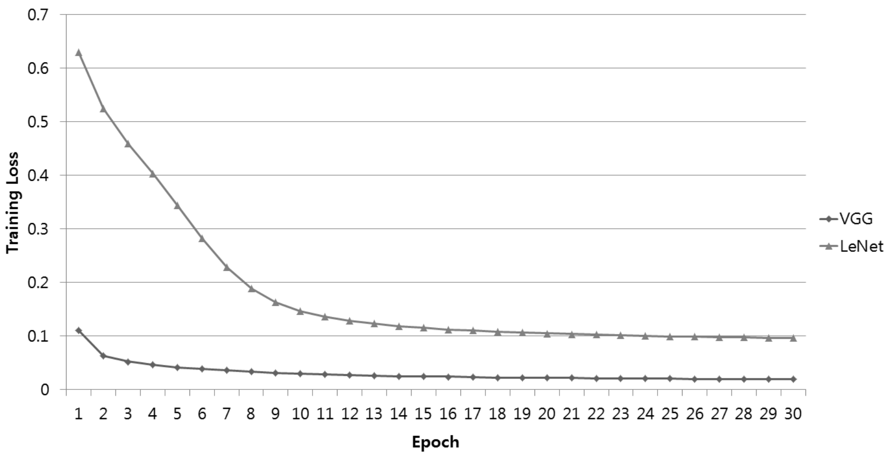


Figure 5.30 The comparison of training loss for TFE(Instagram)

Detection. 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.

Cropped Word Recognition. 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 5.4 through Table5.6. We summarizes the comparative results between our method and three state-of-the-art baselines. As the CTSN can handle variable lengths of words, the words can preserve their aspect ratio and it will help the get higher response from neural network which is trained by the standard character set. The con-

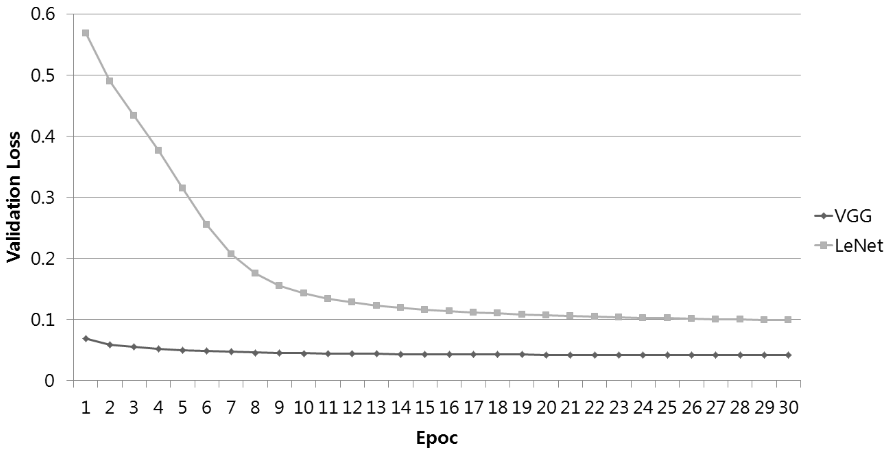


Figure 5.31 The comparison of validation loss for TFE(Instagram)

text information improves performance by excluding the 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.

End-to-End. 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 5.7 assures that our method outperforms all the baselines for both datasets (*e.g.* higher by 17.1%p than the best (`TER`) for Instagram).

Repeatability Table 5.10, Table 5.11 and Table 5.12 summarizes the results of detetion, cropped word recognition, and end-to-end repetition exper-

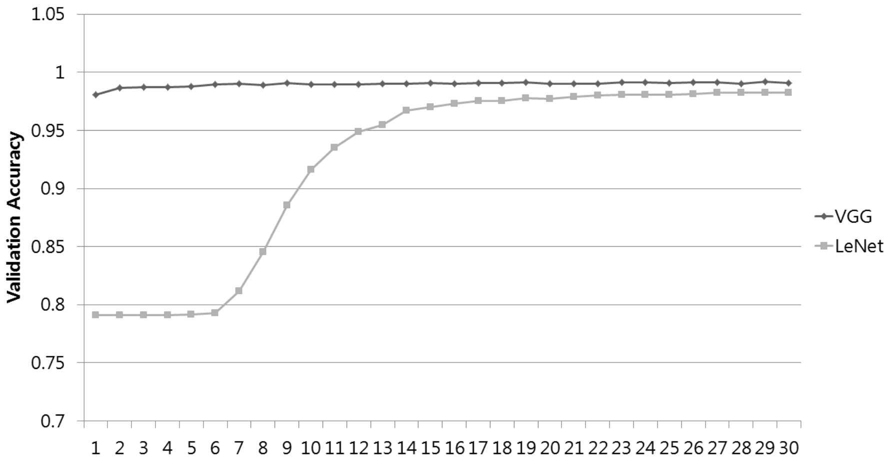


Figure 5.32 The validation score difference between VGG and LeNet for the TFE(Flickr)

Method	Instagram		
	Rec.	Prec.	F1
(`TER`)[80]	37.3	60.1	46.3
(`TFAST`)[7]	64.7	8.0	14.2
CTSN w/o context	45.0	21.6	29.2
CTSN w context	**65.2**	**48.3**	**55.5**

Table 5.1 The results of text detection with precision, recall, F1 scores for Instagram datasets.

iments. Detection is compared with TER, cropped word recognition is compared with TCRNN, and end-to-end result is compared with TER. An experiment is performed on the Instagram dataset. In the case of detection, the proposed method averages 5.1%p higher. The standard deviation is 1.5. The average cropped word recognition is 1.6%p higher. The standard deviation is 1.3. The end-to-end test result is 9.5%p higher.

Qualitative Results. Fig.5.1 illustrates some selected examples of text spotting, with a high variation of locations, fonts, and sizes of text. We depict successful detections with green boxes and failures with blue boxes. As shown

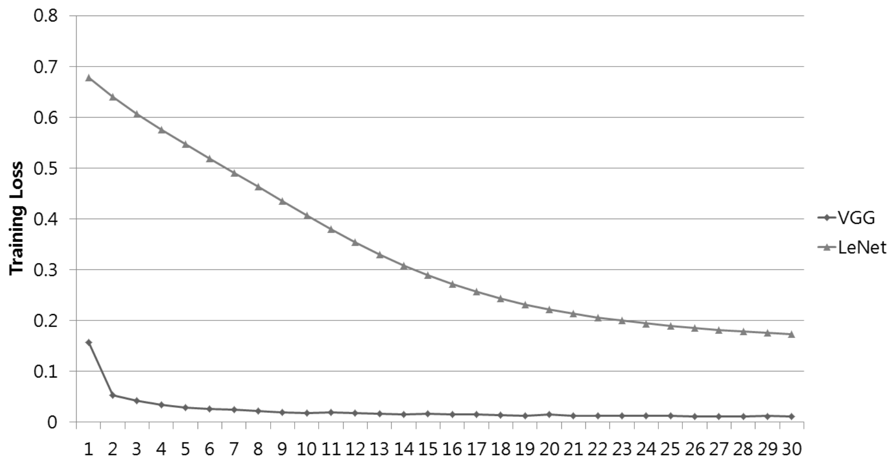


Figure 5.33 The comparison of training loss for TFE(Flickr)

Method	Flickr Rec.	Flickr Prec.	Flickr F1
(TER)[80]	54.9	37.5	44.6
(TFAST)[7]	**70.1**	10.1	17.6
CTSN w/o context	51.2	25.5	34.0
CTSN w context	55.1	**55.9**	**55.5**

Table 5.2 The results of text detection with precision, recall, F1 scores for Flickr datasets.

in the examples, online images often contain a long sequence of words embedded by users with a high variation of locations, fonts, and sizes. The failure cases include word splitting errors, part-only detections, missed detections of low-contrast fonts, and wrong word recognition.

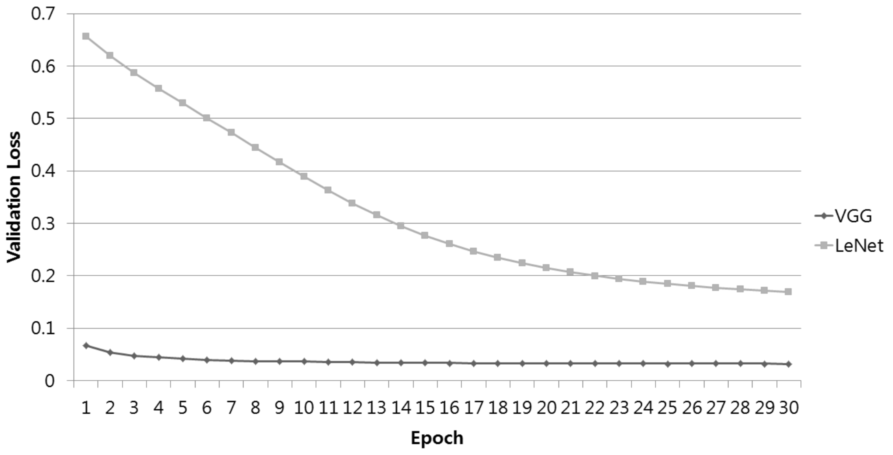


Figure 5.34 The comparison of validation loss for TFE(Flickr)

Method	Pinterest		
	Rec.	Prec.	F1
(TER)[80]	37.9	72.7	49.8
(TFAST)[7]	**67.7**	12.0	20.4
CTSN w/o context	42.2	23.2	29.9
CTSN w context	46.2	**57.7**	**51.3**

Table 5.3 The results of text detection with precision, recall, F1 scores for Pinterest datasets.

Method	Instagram
(TPS) [105]	23.4
(TCNN) [109]	50.6
(TCRNN) [92]	87.1
CTSN w/o context	87.3
CTSN w context	**90.4**

Table 5.4 The accuracy of cropped word recognition for Instagram datasets.

Method	FLICKR
(TPS) [105]	38.2
(TCNN) [109]	55.3
(TCRNN) [92]	88.7
CTSN w/o context	90.3
CTSN w context	**93.0**

Table 5.5 The accuracy of cropped word recognition for Flickr datasets.

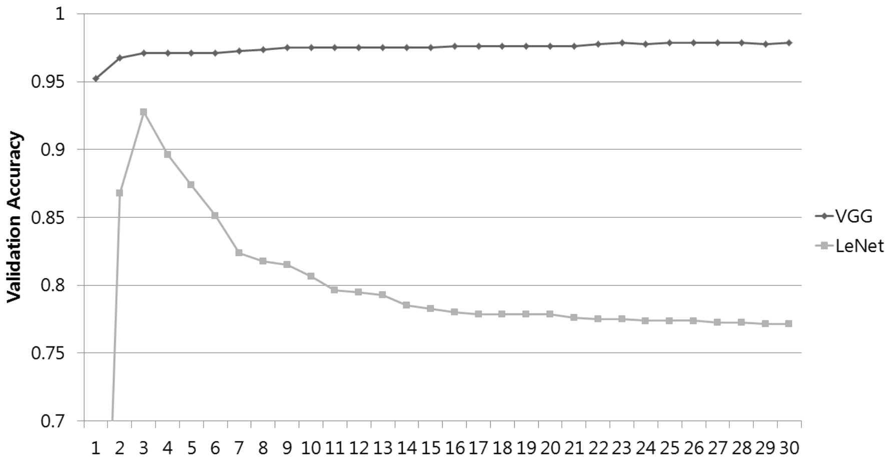


Figure 5.35 The validation score difference between VGG and LeNet for the TFE(Pinterest)

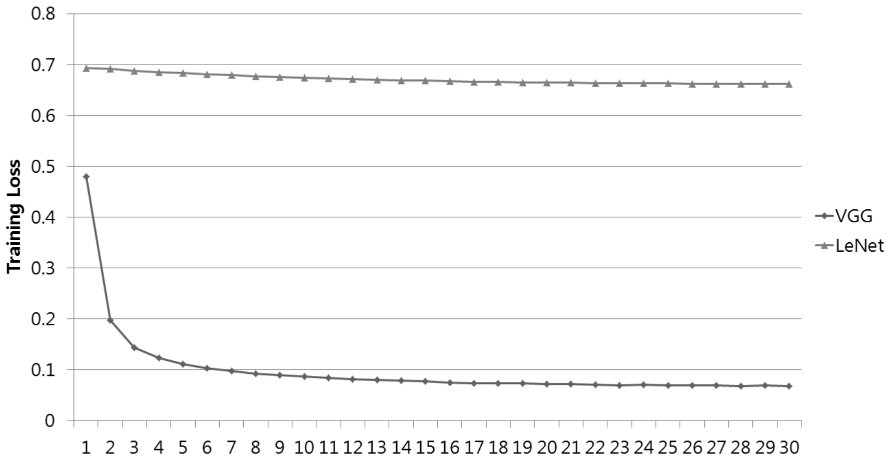


Figure 5.36 The comparison of training loss for the TFE(Pinterest)

Method	Pinterest
(TPS) [105]	20.1
(TCNN) [109]	47.4
(TCRNN) [92]	86.2
CTSN w/o context	88.5
CTSN w context	**90.3**

Table 5.6 The accuracy of cropped word recognition for Pinterest datasets.

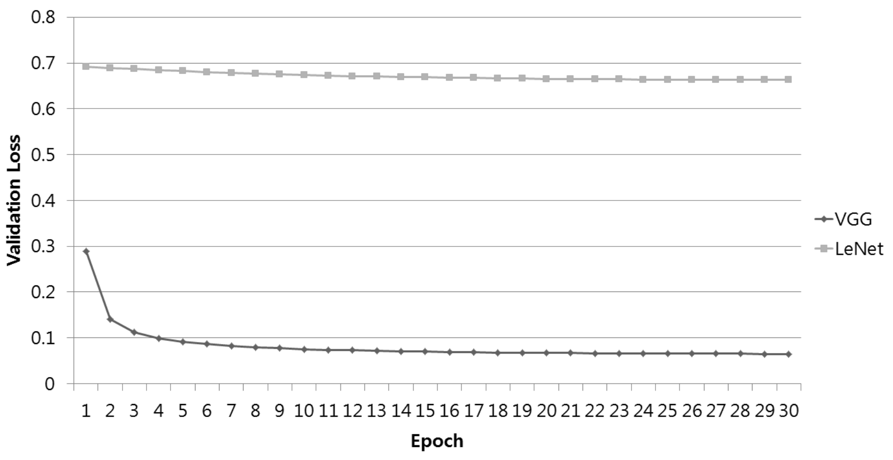


Figure 5.37 The comparison of validation loss for the TFE(Pinterest)

Method	Instagram		
	Rec.	Prec.	F1
(`TPS`)[105]	15.3	7.8	10.3
(`TER`)[80]	23.4	29.8	26.2
(`TCNN`)[109]	4.1	18.7	6.8
CTSN w/o context	**40.7**	12.3	18.9
CTSN w context	39.2	**48.3**	**43.3**

Table 5.7 The results of end-to-end recognition with precision, recall, F1 scores for Instagram datasets.

Method	FLICKR		
	Rec.	Prec.	F1
(`TPS`)[105]	19.6	10.2	13.4
(`TER`)[80]	23.6	27.2	25.3
(`TCNN`)[109]	7.7	16.7	10.5
CTSN w/o context	29.9	15.0	20.0
CTSN w context	**32.5**	**50.9**	**39.7**

Table 5.8 The results of end-to-end recognition with precision, recall, F1 scores for Flickr datasets.

Method	Pinterest		
	Rec.	Prec.	F1
(`TPS`)[105]	12.8	10.5	11.5
(`TER`)[80]	16.2	31.0	21.3
(`TCNN`)[109]	4.5	15.2	6.9
CTSN w/o context	22.3	17.9	19.9
CTSN w context	23.4	57.7	33.3

Table 5.9 The results of end-to-end recognition with precision, recall, F1 scores for Pinterest datasets.

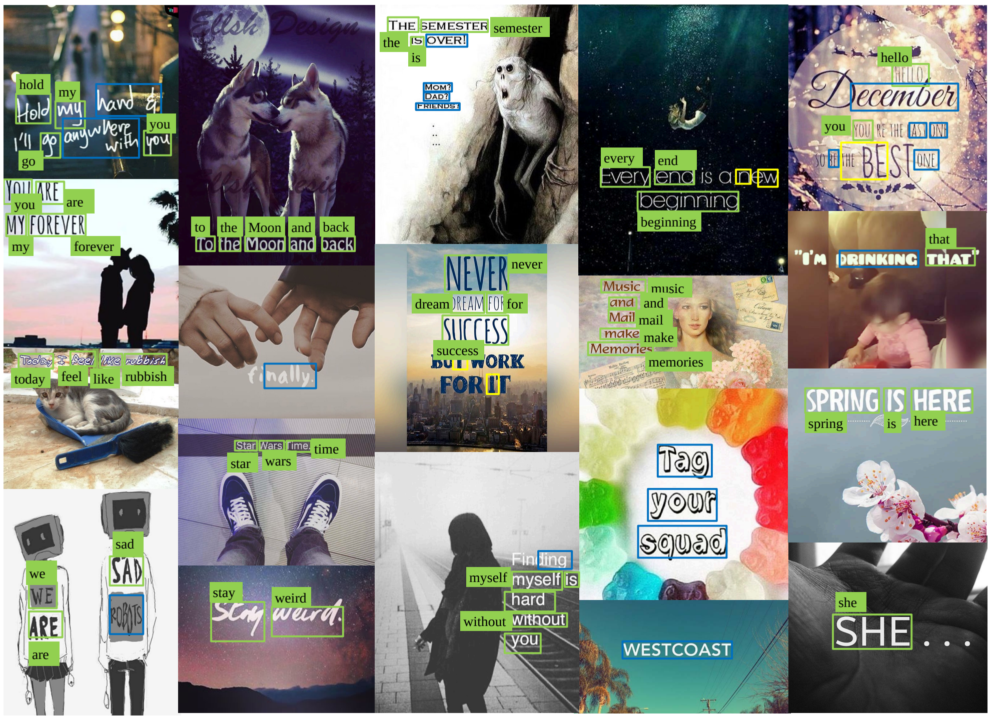


Figure 5.38 Examples of text detection and recognition for Instagram with true positives in green, detection successes but recognition failures in blue, and detection failures in yellow.

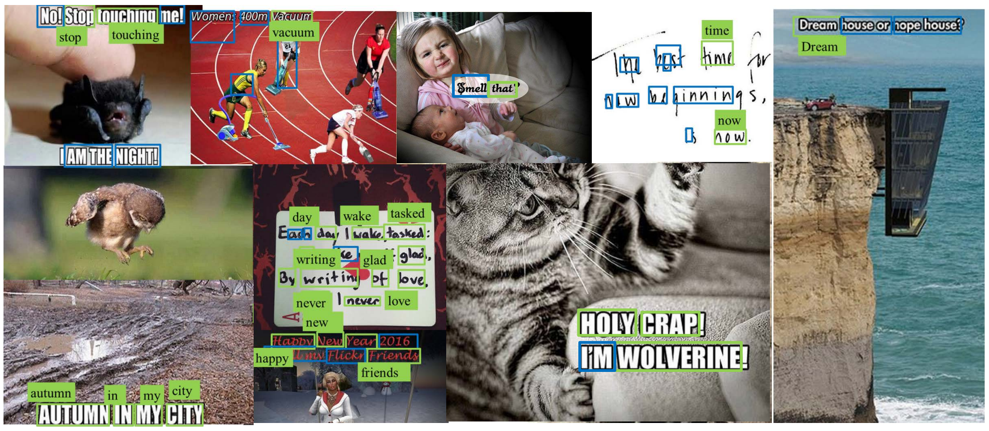


Figure 5.39 Examples of text detection and recognition for Flickr with true positives in green, detection successes but recognition failures in blue, and detection failures in yellow.

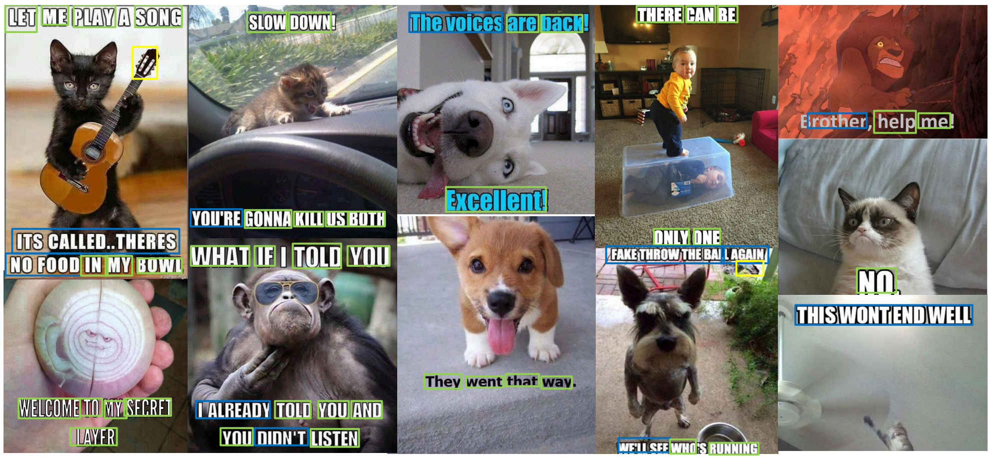


Figure 5.40 Examples of text detection and recognition for Pinterest with true positives in green, detection successes but recognition failures in blue, and detection failures in yellow.

Count	TER			CTSN		
	Rec.	Prec.	F1	Rec.	Prec.	F1
(1)	37.7	60.1	46.3	48.3	65.2	55.5
(2)	40.9	65.2	50.2	47.2	71.6	56.9
(3)	41.6	62.2	49.9	51.8	60.4	55.8
(4)	38.7	62.7	47.9	46.1	65.5	54.1
(5)	39.3	63.4	48.5	48.7	58.0	52.9
mean	41.6	62.2	49.9	48.2	64.1	55.0
std	1.6	1.9	1.6	2.1	5.2	1.5

Table 5.10 The repeatability of text detection with precision, recall, F1 scores for Instagram datasets.

Count	TCRNN Accuracy	CTSN Accuracy
(1)	87.1	90.4
(2)	87.0	87.9
(3)	90.4	90.9
(4)	88.4	89.6
(5)	89.2	91.0
mean	88.4	90.0
std	1.4	1.3

Table 5.11 The repeatability of cropped word recognition for Instagram datasets.

Count	TER			CTSN		
	Rec.	Prec.	F1	Rec.	Prec.	F1
(1)	23.4	29.8	26.2	39.2	48.3	43.3
(2)	30.2	39.7	34.3	30.7	57.3	40.0
(3)	26.8	42.1	32.8	33.7	51.3	40.7
(4)	23.0	40.5	29.3	30.0	55.7	39.0
(5)	27.1	39.1	32.0	31.7	49.3	38.6
mean	26.1	38.2	30.9	33.0	53.1	40.4
std	3.0	4.9	3.2	3.7	3.9	1.9

Table 5.12 The repeatability of end to end with precision, recall, F1 scores for Instagram datasets.

Chapter 6

Conclusion

We proposed the contextual text spotting network (CTSN) model to detect and recognize of 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. Another future work is to detect and recognize the embedded text in videos.

Bibliography

[1] D. Amodei, R. Anubhai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, J. Chen, M. Chrzanowski, A. Coates, G. Diamos, E. Elsen, J. Engel, L. Fan, C. Fougner, T. Han, A. Hannun, B. Jun, P. LeGresley, L. Lin, S. Narang, A. Ng, S. Ozair, R. Prenger, J. Raiman, S. Satheesh, D. Seetapun, S. Sengupta, Y. Wang, Z. Wang, C. Wang, B. Xiao, D. Yogatama, J. Zhan, and Z. Zhu. Deep Speech 2: End-to-End Speech Recognition in English and Mandarin. In *arXiv*, 2015.

[2] A. Arnab, S. Jayasumana, S. Zheng, and P. H. S. Torr. Higher order conditional random fields in deep neural networks. In *ECCV*, 2016.

[3] V. Badrinarayanan, A. Kendall, and R. Cipolla. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Scene Segmentation. *TPAMI*, 1(1):1–14, 2016.

[4] S. Bell, C. L. Zitnick, K. Bala, and R. Girshick. Inside-Outside Net: Detecting Objects in Context with Skip Pooling and Recurrent Neural Networks. In *ArXiv*, 2015.

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

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

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

[8] L.-C. Chen, J. T. Barron, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic Image Segmentation with Task-Specific Edge Detection Using CNNs and a Discriminatively Trained Domain Transform. In *CVPR*, 2016.

[9] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. In *arXiv*, 2016.

[10] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs. In *ICLR*, 2016.

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

[12] X. Chen and C. L. Zitnick. Mind's Eye: A Recurrent Visual Representation for Image Caption Generation. In *CVPR*, 2015.

[13] Z. Cui, H. Chang, S. Shan, and X. Chen. Deep Network Cascade for Image Super-resolution. In *ECCV*, 2014.

[14] G. E. Dahl, D. Yu, L. Deng, and A. Acero. Context-Dependent Pre-Trained Deep Neural Networks for Large-Vocabulary Speech Recognition. *IEEE TRANSACTIONS ON AUDIO, SPEECH, AND LANGUAGE PROCESSING*, 20:30–42, 2012.

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

[16] J. Dean, G. S. Corrado, R. Monga, K. Chen, M. Devin, Q. V. Le, M. Z. Mao, M. A. Ranzato, A. Senior, P. Tucker, K. Yang, and A. Y. Ng. Large Scale Distributed Deep Networks. In *NIPS*, 2012.

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

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

[19] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term Recurrent Convolutional Networks for Visual Recognition and Description. In *arXiv*, 2014.

[20] C. Dong, C. C. Loy, K. He, and X. Tang. Learning a Deep Convolutional Network for Image Super-Resolution. In *ECCV*, 2014.

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

[22] H. Fang, S. Gupta, F. Iandola, R. Srivastava, L. Deng, P. Dollár, J. Gao, X. He, M. Mitchell, J. C. Platt, C. L. Zitnick, and G. Zweig. From Captions to Visual Concepts and Back. In *CVPR*, 2015.

[23] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient Graph-Based Image Segmentation. *IJCV*, 59(2):167—181, 2004.

[24] R. Girshick. Fast R-CNN. In *arXiv*, 2015.

[25] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In *CVPR*, 2014.

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

[27] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTAT*, 2010.

[28] I. Goodfellow, Y. Bengio, and A. Courville. Deep learning. Book in preparation for MIT Press, 2016.

[29] A. Graves. Generating sequences with recurrent neural networks. In *arXiv*, 2013.

[30] A. Graves and N. Jaitly. Towards End-to-End Speech Recognition with Recurrent Neural Networks. In *ICML*, 2014.

[31] A. Graves, A. rahman Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *ICASSP*, 2013.

[32] B. Hariharan, P. Arbelaez, R. Girshick, and J. Malik. Hypercolumns for object segmentation and fine-grained localization. In *CVPR*, 2015.

[33] K. He, X. Zhang, S. Ren, and J. Sun. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. In *ECCV*, 2014.

[34] K. He, X. Zhang, S. Ren, and J. Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *ICCV*, 2015.

[35] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *CVPR*, 2016.

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

[37] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. rahman Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury. Deep Neural Networks for Acoustic Modeling in Speech Recognition. *IEEE Speech Processing Magazine*, 1:82–97, 2012.

[38] G. E. Hinton, S. Osindero, and Y.-W. Teh. A Fast Learning Algorithm for Deep Belief Nets. *Neural Computation*, 18:1527–1554, 2006.

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

[40] S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735—1780, 1997.

[41] C. K. M. Hogn and S. I. Yoo. Learning Texture Image Prior for Super Resolution Using Restricted Boltzmann Machine. In *International Conference on Image Analysis and Processing (ICIAP)*, 2015.

[42] S. Hong, J. Oh, B. Han, and H. Lee. Learning transferrable knowledge for semantic segmentation with deep convolutional neural network. In *arXiv*, 2015.

[43] M. Hradis, J. Kotera, P. Zemcik, and F. Sroubek. Convolutional Neural Networks for Direct Text Deblurring. In *BMVC*, 2015.

[44] W. Huang, Z. Lin, J. Yang, and J. Wang. Text Localization in Natural Images using Stroke Feature Transform and Text Covariance Descriptors. In *ICCV*, 2013.

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

[46] F. Iandola, M. Moskewicz, S. Karayev, R. Girshick, T. Darrell, and K. Keutzer. DenseNet: Implementing Efficient ConvNet Descriptor Pyramids. In *ECCV*, 2014.

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

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

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

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

[51] R. Jozefowicz, O. Vinyals, M. Schuster, N. Shazeer, and Y. Wu. Exploring the Limits of Language Modeling. In *ICLR*, 2016.

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

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

[54] A. Karpathy and L. Fei-Fei. Deep Visual-Semantic Alignments for Generating Image Description. In *CVPR*, 2015.

[55] J. Kim, J. K. Lee, and K. M. Lee. Accurate Image Super-Resolution Using Very Deep Convolutional Networks. In *CVPR*, 2016.

[56] J. Kim, J. K. Lee, and K. M. Lee. Deeply-Recursive Convolutional Network for Image Super-Resolution. In *CVPR*, 2016.

[57] R. Kiros, R. Salakhutdinov, and R. Zemel. Multimodal Neural Language Models. In *ICML*, 2014.

[58] A. Kolesnikov and C. Lampert. Seed, expand and constrain: Three principles for weakly-supervised image segmentation. In *ECCV*, 2016.

[59] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *NIPS*, 2012.

[60] G. Kulkarni, V. Premraj, S. Dhar, S. Li, Y. Choi, A. C. Berg, and T. L. Berg. Baby Talk: Understanding and Generating Image Descriptions. In *CVPR*, 2011.

[61] R. Lebret, P. O. Pinheiro, and R. Collobert. Phrase-based Image Captioning. In *ICML*, 2015.

[62] Y. LeCun, Y. Bengio, and G. E. Hinton. Deep Learning. *Nature*, 521(1):436—444, 2015.

[63] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based Learning Applied to Document Recognition. *Proc. IEEE*, 86(11):2278—2324, 1998.

[64] J. Lee, K. Cho, and T. Hofmann. Fully Character-Level Neural Machine Translation without Explicit Segmentation. In *arXiv*, 2016.

[65] M. Liao, B. Shi, X. Bai, X. Wang, and W. Liu. Text Boxes: A fast text detector with a single deep neural network. In *AAAI*, 2017.

[66] G. Lin, C. Shen, I. Reid, and A. van dan Hengel. Efficient piecewise training of deep structured models for semantic segmentation. In *arXiv*, 2015.

[67] Z. Liu, X. Li, P. Luo, C. C. Loy, and X. Tang. Semantic image segmentation via deep parsing network. In *ICCV*, 2015.

[68] J. Long, E. Shelhamer, and T. Darrel. Fully Convolutional Networks for Semantic Segmentation. In *CVPR*, 2015.

[69] S. M. Lucas, A. Panaretos, L. Sosa, A. Tang, S. Wong, and R. Young. ICDAR 2003 Robust Reading Competitions. In *ICDAR*, 2003.

[70] M.-T. Luong, H. Pham, and C. D. Manning. Effective Approaches to Attention-based Neural Machine Translation. In *arXiv*, 2015.

[71] M.-T. Luong, I. Sutskever, Q. V. Le, O. Vinyals, and W. Zaremba. Addressing the Rare Word Problem in Neural Machine Translation. In *arXiv*, 2015.

[72] C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang. Hierarchical Convolutional Features for Visual Tracking. In *ICCV*, 2015.

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

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

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

[76] M. Mostajabi, P. Yadollahpour, and G. Shakhnarovich. Feedforward semantic segmentation with zoom-out features. In *CVPR*, 2015.

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

[78] Y. Nesterov. A method of solving a convex programming problem with convergence rate O (1/k2). *Soviet Mathematics Doklady*, 27(2):372––376, 1983.

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

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

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

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

[83] G. Papandreou, L.-C. Chen, K. Murphy, and A. L. Yuille. Weakly- and Semi-Supervised Learning of a Deep Convolutional Network for Semantic Image Segmentation. In *ICCV*, 2015.

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

[85] Y. Pu, Z. Gan, R. Henao, X. Yuan, C. Li, A. Stevens, and L. Carin. Variational Autoencoder for Deep Learning of Images, Labels and Captions. In *NIPS*, 2016.

[86] S. Reed, Z. Akata, H. Lee, and B. Schiele. Learning Deep Representations of Fine-Grained Visual Descriptions. In *CVPR*, 2016.

[87] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *arXiv*, 2015.

[88] H. Sak, A. Senior, K. Rao, and F. Beaufays. Fast and accurate recurrent neural network acoustic models for speech recognition. In *arXiv*, 2015.

[89] A. M. Saxe, J. L. McClelland, and S. Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In *ICLR*, 2014.

[90] R. Sennrich, B. Haddow, and A. Birch. Neural Machine Translation of Rare Words with Subword Units. In *arXiv*, 2016.

[91] A. Sharma, O. Tuzel, and D. W. Jacobs. Deep hierarchical parsing for semantic segmentation. In *CVPR*, 2015.

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

[93] B. Shi, X. Wang, P. Lyu, C. Yao, and X. Bai. Robust Scene Text Recognition with Automatic Rectification. In *CVPR*, 2016.

[94] H. Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 2000.

[95] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *arXiv*, 2014.

[96] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *JMLR*, 15(1):1929—1958, 2014.

[97] J. Sun, W. Cao, Z. Xu, and J. Ponce. Learning a Convolutional Neural Network for Non-uniform Motion Blur Removal. In *CVPR*, 2015.

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

[99] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to Sequence Learning with Neural Networks. In *NIPS*, 2014.

[100] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

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

[102] Z. Tian, W. Huang, T. He, P. He, and Y. Qiao. Detecting Text in Natural Image with Connectionist Text Proposal Network. In *ECCV*, 2016.

[103] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders. Selective Search for Object Recognition. *IJCV*, 104(2):154–171, 2013.

[104] S. Venugopalan, H. Xu, J. Donahue, M. Rohrbach, R. Mooney, and K. Saenko. Translating Videos to Natural Language Using Deep Recurrent Neural Networks. In *NAACL-HLT*, 2015.

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

[106] K. Wang and S. Belongie. Word Spotting in the Wild. In *ECCV*, 2010.

[107] L. Wang, W. Ouyang, X. Wang, and H. Lu. Visual Tracking with fully Convolutional Networks. In *ICCV*, 2015.

[108] N. Wang and D.-Y. Yeung. Learning a Deep Compact Image Representation for Visual Tracking. In *NIPS*, 2013.

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

[110] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, and M. Norouzi. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. In *arXiv*, 2016.

[111] W. Xiong, J. Droppo, X. Huang, F. Seide, M. Seltzer, A. Stolcke, D. Yu, and G. Zweig. Achieving Human Parity In Conversational Speech Recognition. In *arXiv*, 2016.

[112] K. Xu, J. L. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. S. Zemel, and Y. Bengio. Show, Attend, and Tell: Neural Image Caption Generation with Visual Attention. In *ICML*, 2015.

[113] L. Xu, J. S. Ren, C. Liu, and J. Jia. Deep Convolutional Neural Network for Image Deconvolution. In *NIPS*, 2014.

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

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

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

[117] M. D. Zeiler. ADADELTA: An Adaptive Learning Rate Method. In *ArXiv*, 2012.

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

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

[120] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. S. Torr. Conditional random fields as recurrent neural networks. In *ICCV*, 2015.

[121] G. Zhu, F. Porikli, and H. Li. Robust Visual Tracking with Deep Convolutional Neural Network based Object Proposals on PETS. In *CVPR*, 2016.

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

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

요약

본 논문은 mobile 영상 촬영기기 및 휴대폰의 증가에 따라 급격히 늘어나고 있는 Internet 상의 영상에서, 사용자가 전하고자 하는 의미나 메시지를 더 잘 표현하고자 추가한 Text를 검출하고 인식하는 문제를 다루고 있다. 검출을 위해서는 Text내의 연속적인 Character 들을 Context 정보로 활용하고, 영상에 동반되는 metadata, 즉, title, tag, comment 등을 Context 정보로 활용하여 인식한다. 이러한 두 가지 Context 정보들이 Neural Network Frame 내에서 활용될 수 있도록 Contextual Text Spotting Network을 제안하였다. 제안된 방법의 검증을 위해서 Internet 상 Social Network Service의 영상 및 관련 metadata를 수집하였고, 이러한 영상들에 대해서 제안된 방법으로 검출 및 인식 실험을 수행한 결과 기존 방법들보다 우수한 성능을 보였다.

주요어: 텍스트 검출, 텍스트 인식, 문맥 모델, 딥러닝

학번: 2013-30218