Text Detection in Natural Images with Convolutional Neural Networks and Synthetic Training Data

Marco Grond*, Willie Brink† and Ben Herbst‡
Department of Mathematical Sciences, Division of Applied Mathematics
Stellenbosch University
Stellenbosch, South Africa
E-mail: *grondmm@gmail.com, †wbrink@sun.ac.za, ‡herbst@sun.ac.za

***Abstract*—Recognizing text in natural images can be a useful tool for image understanding. We focus on the detection problem, which is to find regions in an image occupied by text. We consider multi-layered convolutional neural networks as a means to classify local regions as text or not, and take a sliding-window approach to scan a full image. For training we generate large synthetic datasets to complement the much smaller available sets of labelled natural images. Our results suggest that larger networks can perform better on this problem, and the highest test accuracy is achieved from training with synthetic data and fine-tuning with natural data.**

I. Introduction

The automatic finding and interpretation of text in natural images (like the one in Fig. 1) can be extremely useful for a variety of applications including autonomous navigation, driver assistance, accessibility for the visually impaired, mobile technologies, and the computer vision problem of complete image understanding. Significant advances have been made to recognize text in more controlled environments, such as OCR systems that operate on scanned documents. The more general setting, however, remains challenging. Text in natural images tends to be far less distinguishable from the rest of the image content, and is often distorted by unpredictable lighting conditions, perspective, and occlusion. The massive variety in size, font and colour further complicates matters.

In this paper we focus on the detection problem, which is to locate regions in a given image that are occupied by identifiable text. Once detected, these regions can be fed to a text recognition system for interpretation.

It is our view that instances of text in natural images are better characterized by local appearance than by their meaning as a whole. Unlike many other approaches that search for specific characters or words, we look at the problem as an appearance-based classification problem. Our approach is to scan the image under consideration in a sliding-window manner, and implement a convolution neural network (CNN) to classify local image patches as "text" or "not text".

CNNs have shown tremendous promise in image-based object classification [1] and have noteworthy roots in text recognition [2]. These techniques are also particularly adept at extracting both low- and high-level features from image data for better class separation. This is an important property for us, since the appearance of text in natural images may require abstract description to compensate for the variety of distortion factors mentioned above.

A significant limitation of CNNs is that to be effective they usually require large, manually labelled training sets. For our problem, however, we can synthesize arbitrarily large training sets by overlaying machine generated text on natural images. Even though our synthetic images are rather unnatural in appearance, we find that they can be combined with a much smaller set of manually labelled natural images to boost classification accuracy.

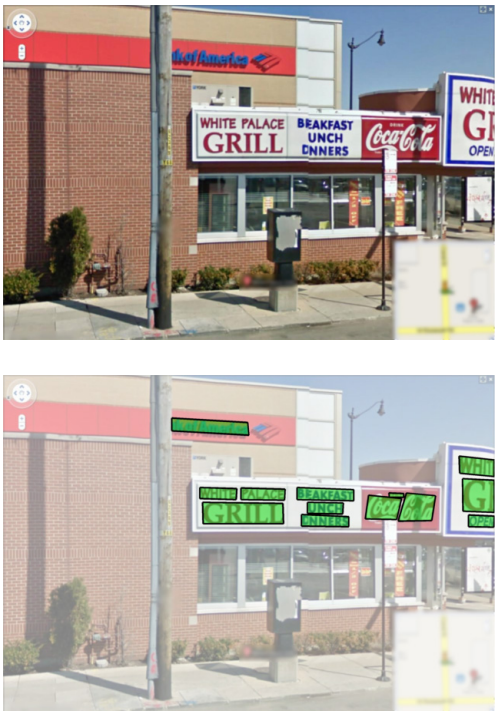


Fig. 1: An example of a natural image containing text (top), and manually selected text regions to be detected (bottom), from the Street View Text dataset [3].

We proceed with a brief overview of related literature in Section II, and some background on CNNs in Section III. Details of our implementation and experimental results follow in Sections IV and V, and we conclude in Section VI.

II. Related Work

The problem of detecting text in natural images has been approached in a number of different ways. The most recent and successful effort appears to be one where stacked convolutional neural networks are employed [4]. Unlike our approach, this one specializes on recognizing words and trains a multi-class classifier to that end. It also uses synthetic data to train the system for translating image data to character strings.

A relatively popular approach has been to detect single characters and then group them together for recognition. Edge detection and the stroke width transform can be used to extract features for detection [5], [6]. It has also been reported that the MSER feature detector can work well for finding text [7], [8]. Gradient features and colour uniformity are further options for identifying individual characters [9].

Apart from the work of Jaderberg et al. [4], other attempts have been made to detect entire words or lines of text, for example using a support vector machine to recognize text-like texture [10].

III. Convolutional Neural Networks

In this section we provide a brief overview of convolutional neural networks. The convolutional layers in such a network are often stacked, resulting in what is termed deep neural networks. The reader is referred to the survey papers of Schmidhuber [11] and LeCun et al. [12] for more detail.

A neural network typically takes an input vector through one or more hidden layers, where combinations of the values in one layer are transformed in a certain way and then fed to the next layer, to be turned into an output vector. For image classification the input can be the pixel intensities of an image to be classified, and the output can be class probabilities. Parameter values that specify the transformations in the various layers are learned in a training phase from known input-output correspondences.

A. Layers

A convolutional layer in a neural network slides filters across the input vector (which for the first level is usually unprocessed image data) for feature extraction. A nonlinearity can be applied to the layer's output, allowing the network to find nonlinear relationships in the data. A further common approach is max pooling, which outputs maxima of localized but overlapping subsets of the incoming vector. A pooling layer can also reduce the input dimension for the next layer, and introduces a form of invariance to translation.

Repeating these types of layers by stacking one after the other may give the network the ability to extract both low-level features (e.g. edges) and more abstract, higher-level features. It does, however, increase the number of filter weights to be learned which in turn may increase the required size of the training set quite dramatically.

The convolutional and pooling layers are usually followed by fully connected layers where, unlike convolutional and pooling layers that operate on local regions of their inputs, exhaustive connections are considered between all the outputs of a previous layer. This may allow the network to reason about the data on a high level.

In the case of classification, the final output layer can be a softmax function, as used in multi-class logistic regression, that outputs a vector of probabilities associated with the predefined set of classes.

B. Training

As mentioned, the parameters in the various layers (such as the filter weights in a convolutional layer) must be learned. This involves consideration of known input-output correspondences, which in terms of image classification amount to a representative set of images for each of the classes.

Training can be accomplished by a process termed back-propagation. Firstly all the network parameters are initialized, often randomly. The training images are then passed through the network, while their true class labels are passed backwards through the network, and errors (or losses) are computed for every layer. The errors are scaled by a specified learning rate, and stochastic gradient descent [13] can be utilized to update the network parameters in a manner that will iteratively minimize the error. Momentum can also be added to take previous parameter updates into account, and the learning rate is often decreased as training progresses to allow for convergence to an optimum.

A multi-layered CNN with fully connected layers built for image classification may contain millions of parameters. This creates a need for large training sets, often far too large for manual labelling to be feasible. Training can also be a time consuming process but, once trained, passing data through the network would involve a series of simple filter operations which can be implemented very efficiently on a GPU.

IV. Implementation

In this section we describe our solution to the text detection problem. Our approach is to consider local image regions, and attempt to classify each as "text" or "not text" using a multi-layered CNN that outputs the two class probabilities for a given image patch. The full image can then be scanned in a sliding-window manner, as detailed in Section IV-E.

We will compare two different network architectures, as explained in Section IV-D. Both of these consist of multiple convolutional and pooling layers, and require a large amount of training data (labelled examples of the two classes). While annotated datasets are available, such as the one containing the image in Fig. 1, we found them to be fairly small and decided to augment our training set with synthetic data.

A. Natural images

We collected 350 images from the Street View Text dataset [3] and 185 from the MSRA-TD500 dataset [6] (which has 500 images, but many contain only Chinese characters). The latter is fully annotated, in the sense that all regions containing text are specified as metadata. The former provides annotations only for the larger names of businesses on buildings, so we added annotations of all other instances of text. Additionally, we took 39 frames from news broadcasts on cnn.com and selected text regions on those. These sources provide us with 574 natural images in total, and we supplement this collection with the 12,503 natural single character images from the Chars74K dataset [14].

B. Synthetic images

In order to increase our training data by a substantial amount, we can consider synthetic data. Machine generated text over a natural image background may contain the essence of the appearance of text occurring naturally, which a neural network should be able to extract. We created a range of words of varying fonts, sizes, orientations, colours and opacities, and blended these words onto natural background images. Since we control the placing of words, annotation (i.e. specifying the text regions) is trivial.

We extracted 4,657 of the larger images from the SUN2012 dataset [15] to use as backgrounds. For each of these images we picked between one and ten words from a list of the 10,000 most common English words. For each word we randomly selected a font from about 700 options, a scale based on the size of the image (to avoid extremely small or extremely large text), a colour, and an opacity between 0.6 and 1. We then placed the word on the background at a random position and orientation, on the condition that no two word placements overlap. In keeping with the annotation style of the natural datasets from Section IV-A, we stored the bounding rectangle of every added word.

C. Sampling image windows

From our 574 natural images and 4,657 synthetic images we can extract windows (square regions) which will be used to train and evaluate a network. We randomly select 200×200 windows from all the images, and label each according to its overlap with the known text regions. If more than 50% of the window overlaps with text regions, the window is labelled as "text". If less than 5% of the window overlaps with text regions, the window is labelled as "not text". Windows that do not fit these categories are discarded, to avoid ambiguity in the data.

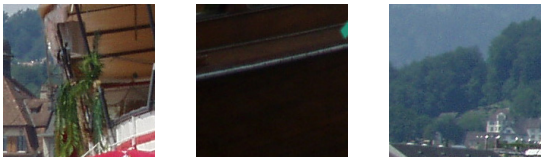
Figure 2 provides examples of these windows from one of the synthetic images. It may be noted that the appearance of text in this image is rather unnatural, but perhaps less so on the scale of the extracted windows (which is the scale the network will focus on).

We split the labelled windows into a number of training, validation and test datasets, as indicated in Table I. We keep the natural and synthetic data separate in order to test the

(a) text added to natural image

(b) selected windows

(c) windows labelled as "text"



(d) windows labelled as "not text"

(e) windows discarded due to ambiguity

Fig. 2: An example of text synthetically added to a natural background image, and windows extracted from the result. For training and testing purposes each of these windows is labelled as "text" or "not text", or discarded if between 5% and 50% of its area overlaps with text regions (i.e. it has neither enough text nor enough background).

TABLE I: Details of the various datasets used for training, validation and testing in this study. Note that nat_train1 is a subset of nat_train2 which is a subset of the 373 natural images in nat_train3. The latter set is supplemented with 12,503 single character images.

dataset	# images	# windows	# positives (text)	# negatives (not text)
synth_train	3,634	1,090,709	416,195	674,514
synth_val	1,023	307,721	115,782	191,939
nat_train1	200	15,837	8,223	7,613
nat_train2	320	25,979	13,758	12,221
nat_train3	373 + 12,503	57,714	28,880	28,834
nat_val	86	7,640	3,750	3,890
nat_test	115	10,695	5,488	5,207

efficacy of each before we combine them. The training set of windows from the natural images is supplemented with the single character images from the Chars74K dataset [14]. Note also that we use only natural image windows in the holdout test set, in accordance with the aim of this study.

D. Network structures

We consider two network architectures: LeNet-5 [2] and ImageNet [1]. The classic LeNet structure is chosen since it was initially implemented for character recognition, which is similar to our problem. The much larger ImageNet structure was originally used for object classification and is chosen for its impressive performance on image data. LeNet consists of two convolutional layers followed by two fully connected layers. ImageNet has five convolutional layers followed by three fully connected layers. We use these structures as is, except that we replace the sigmoidal activation functions in LeNet with rectified linear unit activations [12] because of their recent popularity.

We train a number of networks using the various training sets listed in Table I. Firstly, we train each of the two structures using random initialization and the synthetic data in synth_train. Randomization is accomplished by sampling a zero-mean Gaussian in the case of ImageNet, and the Xavier algorithm [16] in the case of LeNet. For comparison we then also train randomly initialized networks using the small, medium and large natural datasets (nat_train1, nat_train2 and nat_train3 respectively). Finally, we initialize networks with the parameters obtained through training with the synthetic data, and perform training with the natural sets. The idea is to leverage the size of our synthetic training set, but then fine-tune the network parameters according to a much smaller natural training set.

With random initialization we use a larger learning rate than when we fine-tune. In most cases we implement inverse learning rate decay, and the only exception is the ImageNet structure trained on synthetic data where we reduce the learning rate at specific step intervals. It is not always clear why different learning rate policies influence training in the manner that they do, but we experimented with various ones to give each network a fair chance of training successfully.

E. Text detection in a full image

We train the various networks to classify a local image region, like the windows shown in Fig. 2, as either text or not. If a window is passed through a network we obtain a probability that at least 50% of that window is text (recall that this was our definition of the "text" class when we assigned labels to our training data).

To detect text in a full image, we can simply scan the image in a sliding-window manner and determine a probability for each window. By allowing the windows to overlap, and averaging probabilities per pixel, we can achieve a higher effective resolution and possibly improve the localization of text regions. We note that there is room for improvement in this strategy, by combining probabilities in a more statistically rigorous manner for example, and we comment further on the matter in Section VI. However, as we indicate at the end of the next section, this simple approach can already produce useful results.

V. Experimental Results

In total we trained 14 networks using combinations of different structures, initialization strategies, and training sets. These networks were then tested against the holdout set nat_test, which consists of windows extracted from natural images. Table II summarizes our findings. Here accuracies refer to the combined true-positive and true-negative rates (i.e. what percentage of test windows are classified correctly).

Networks A and H resulted from training the two structures on synthetic data. Although training and validation accuracies are quite high (around 81 % for the LeNet structure and 94 % for ImageNet), the test accuracies are noticeably poorer (72 % and 66 %). This indicates that the features extracted from pure synthetic training data might not immediately generalize to naturally occurring text.

We then experimented with training the two structures with natural data only, using the small, medium and large training sets nat_train1, nat_train2 and nat_train3. This produced networks B, C, D and I, J, K in Table II. Training accuracy improves a lot for LeNet (to around 98 %), while it decreases slightly for ImageNet (now around 90 %). However the difference between training and validation accuracy increases, particularly for LeNet, indicating over-fitting. The effect is reduced with larger training sets which, as might be expected, suggests that more training data is better. Test accuracy does improve from what is obtained with networks A and H, and quite significantly so for the ImageNet structure.

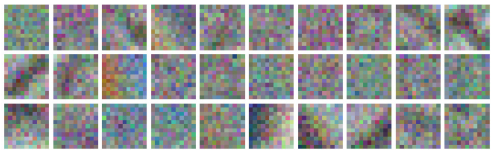
Finally we considered the combination of synthetic and natural training data, by initializing networks with the parameters obtained from training on synthetic data (LeNet network A and ImageNet network H) and re-training with the natural training sets. This resulted in networks E, F, G and L, M, N. It is this strategy that performs best on the test data, with improvements in accuracy of about 7 % for LeNet and 18 % for ImageNet compared to the networks trained with synthetic data only, and improvements of about 3 % compared to the networks trained with natural data only.

TABLE II: Classification accuracies (combined true-positive and true-negative rates) achieved from various network structures, initialization strategies and training sets. We used synth_val as validation set for networks A and H, and nat_val for the others. The trained networks were all tested on the holdout set nat_test.

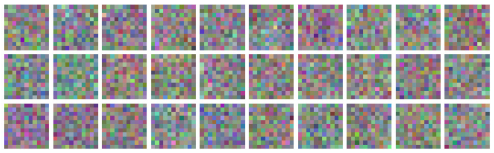
network	structure	initialization	training set	training accuracy	validation accuracy	test accuracy
A	LeNet	random	synth_train	81.7 %	81.4 %	72.2 %
B	LeNet	random	nat_train1	99.6 %	74.8 %	75.0 %
C	LeNet	random	nat_train2	98.9 %	79.4 %	78.8 %
D	LeNet	random	nat_train3	97.1 %	78.9 %	76.7 %
E	LeNet	network A	nat_train1	99.5 %	77.6 %	77.7 %
F	LeNet	network A	nat_train2	97.5 %	80.3 %	80.3 %
G	LeNet	network A	nat_train3	95.4 %	81.1 %	81.0 %
H	ImageNet	random	synth_train	94.8 %	94.2 %	66.3 %
I	ImageNet	random	nat_train1	92.6 %	82.5 %	79.2 %
J	ImageNet	random	nat_train2	88.6 %	83.9 %	81.2 %
K	ImageNet	random	nat_train3	88.4 %	83.3 %	84.3 %
L	ImageNet	network H	nat_train1	96.2 %	82.5 %	81.7 %
M	ImageNet	network H	nat_train2	94.8 %	85.6 %	85.2 %
N	ImageNet	network H	nat_train3	92.9 %	85.7 %	**87.0 %**

It is clear that more training data improves performance (networks C and D offer an exception, but not a significant one). We also conclude that the ImageNet structure performs better overall than the LeNet structure, suggesting that higher-level description is required to successfully separate the appearance of "text" from that of "not text".

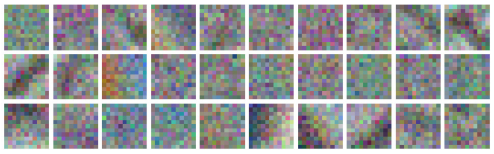
Figure 3 provides a visualization of filter weights in the first layers of networks H, K and N. Some of the structures seen in (a) and (c) are reminiscent of edge detectors, and hint to the idea that larger training sets may enable a network to extract more meaningful features that generalize better.



(a) trained with synth_train



(b) trained with nat_train3

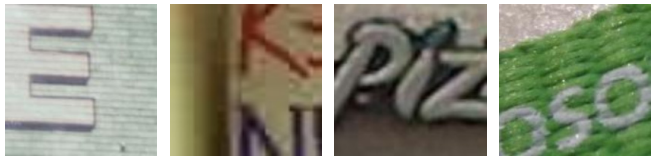


(c) trained with synth_train, fine-tuned with nat_train3

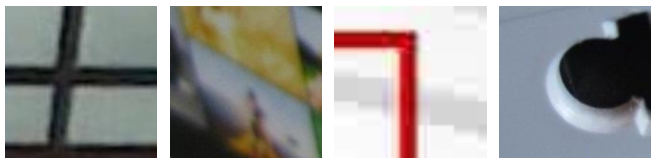
Fig. 3: A visualization of 30 filters in the first convolutional layers of networks H, K and N. The filters in (b) appear almost random, while a few in the other two (trained with much more data) contain structure similar to edge detectors.

A few examples of windows incorrectly classified by network N are shown in Fig. 4. We find that those incorrectly classified as "text" (i.e. false positives) often contain text-like properties such as sharp edges or homogeneous texture. Those windows incorrectly classified as "not text" (false negatives) may contain less pronounced differences between text and background or unusual fonts. Sometimes the reason for misclassification is less clear, likely due to the highly complex nature of multi-layered CNNs.

As a final experiment we applied the trained networks to full images in a sliding-window fashion. We allow overlap between windows and average the probabilities return by the network for every pixel. Some examples generated in this way with networks H, K and N are shown in Fig. 5. For these particular examples it seems as if network H (trained with synthetic data) tends to under-detect and network K (trained with natural data) tends to over-detect, while network N (trained with synthetic data and fine-tuned with natural data) offers a good compromise.

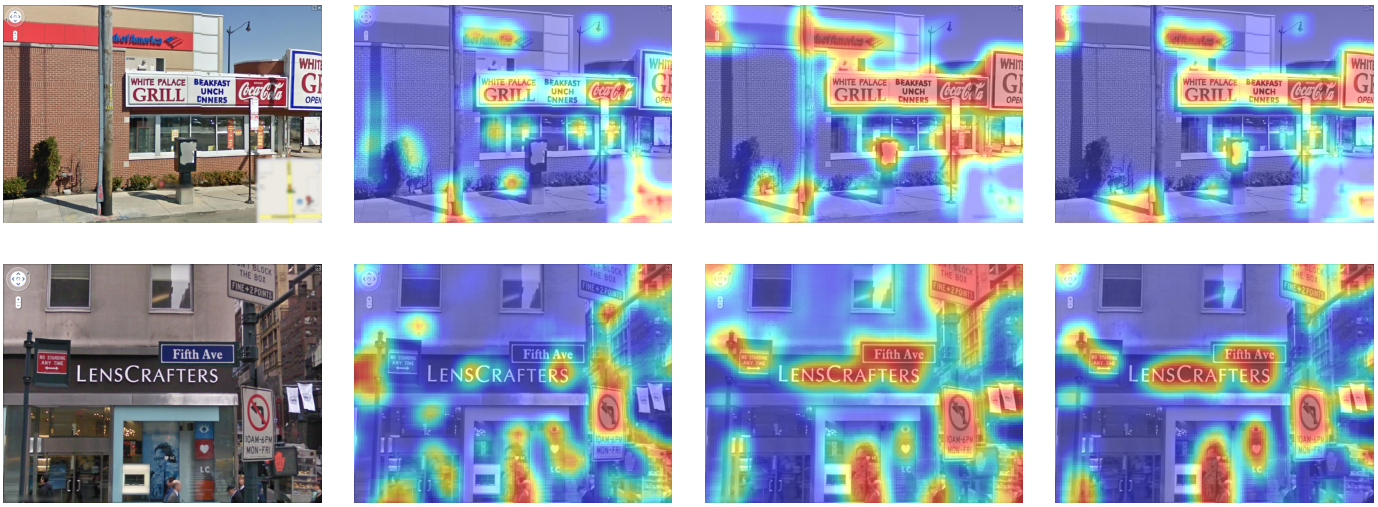


(a) false negatives (incorrectly classified as "not text")



(b) false positives (incorrectly classified as "text")

Fig. 4: A few examples of windows incorrectly classified by network N.



(a) input image (b) network H (c) network K (d) network N

Fig. 5: Text detection in full images by classifying local windows and per-pixel averaging probabilities returned by a network (probabilities are displayed in the jet colourmap). All three networks have the ImageNet structure, H was trained on synth_train, K on nat_train3, and network N was initialized with the parameters of H and then fine-tuned with nat_train3.

VI. Conclusion

The aim of this work was to create a system capable of detecting text in natural images. Because of the intricate variabilities to be expected in the appearance of text regions (including size, orientation, font and colour) we decided to implement a multi-layered CNN and train it to classify a local image region as "text" or "not text". These networks require large labelled training sets to be effective, but in this case we could synthesize data quite easily by overlaying machine generated text on background images.

Our results indicate that a larger network structure (like ImageNet) is needed for this problem, and that a network with fewer layers (like LeNet) can be prone to over-fitting. Our results further suggest that surprisingly high accuracy is achievable with a relatively small natural training set, but initializing the network parameters to those obtained from training on a large synthetic dataset can improve matters. Even though this improvement is an increase in test accuracy of about 3 %, further analysis indicates that we can expect significantly better generalization from the approach.

For future work it might be fruitful to revisit the strategy of averaging probabilities of overlapping windows to arrive at probabilities on pixel-level. Image pyramids can also be implemented to account for larger variation in text size. Once a result like one of those in Fig. 5 is obtained, the next step would be to delineate text regions and pass them to a recognition system for interpretation. False positives could also be eliminated in that process.

References

[1] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[2] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[3] K. Wang and S. Belongie, "Word spotting in the wild," in *European Conference on Computer Vision*, 2010, pp. 591–604.

[4] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman, "Reading text in the wild with convolutional neural networks," *International Journal of Computer Vision*, vol. 116, no. 1, pp. 1–20, 2016.

[5] B. Epshtein, E. Ofek, and Y. Wexler, "Detecting text in natural scenes with stroke width transform," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2010, pp. 2963–2970.

[6] C. Yao, X. Bai, W. Liu, Y. Ma, and Z. Tu, "Detecting texts of arbitrary orientations in natural images," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 1083–1090.

[7] H. Chen, S. Tsai, G. Schroth, D. Chen, R. Grzeszczuk, and B. Girod, "Robust text detection in natural images with edge-enhanced maximally stable extremal regions," in *IEEE International Conference on Image Processing*, 2011, pp. 2609–2612.

[8] X. Yin, X. Yin, K. Huang, and H. Hao, "Robust text detection in natural scene images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 5, pp. 970–983, 2014.

[9] C. Yi and Y. Tian, "Text string detection from natural scenes by structure-based partition and grouping," *IEEE Transactions on Image Processing*, vol. 20, no. 9, pp. 2594–2605, 2011.

[10] K. Kim, K. Jung, and J. Kim, "Texture-based approach for text detection in images using support vector machines and continuously adaptive mean shift algorithm," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 12, pp. 1631–1639, 2003.

[11] J. Schmidhuber, "Deep learning in neural networks: an overview," *Neural Networks*, vol. 61, pp. 85–117, 2015.

[12] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 51, no. 7553, pp. 436–444, 2015.

[13] L. Bottou, "Stochastic gradient descent tricks," in *Neural Networks: Tricks of the Trade*. Springer, 2012, pp. 421–436.

[14] T. de Campos, B. Babu, and M. Varma, "Character recognition in natural images," in *International Conference on Computer Vision Theory and Applications*, 2009, pp. 273–280.

[15] J. Xiao, J. Hays, K. Ehinger, A. Oliva, and A. Torralba, "Large-scale scene recognition from abbey to zoo," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2010, pp. 3485–3492.

[16] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *International Conference on Artificial Intelligence and Statistics*, 2010, pp. 249–256.