Synthesizing Scene Text Images for Recognition with Style Transfer

Haoran Liu, Anna Zhu*
School of Computer, Wuhan University of Technology, Wuhan, Hubei, China.

***Abstract*—Majority of the existing datasets for tasks like scene text recognition only include a few thousand images with a very limited words and characters. Therefore, it cannot meet the need of the typical deep learning based text recognition approaches. In the same time, although the standard synthetic datasets usually comprise millions of scene text images, which means that the data distribution of the small target datasets cannot be learned very well. We propose a word image generating method called Synth-Text Transfer Network to solve these problems. It has the capability of estimating and simulating the distribution of target datasets. Synth-Text Transfer Network utilizes a style transfer approach to synthesis images with arbitrary text content with preserving the texture of the referenced style image in the target dataset. The large amount of synthesized images can help to alleviate the overfitting problem and improve the accuracy in latter scene text image recognition tasks. In addition, our proposed method is flexible and fast, which has a relatively fast speed among regular style transfer approaches.**

Keywords: data synthesis; style transfer; deep learning based; scene text recognition

I. INTRODUCTION

Scene text recognition has been a challenging but very useful task in recent years. Text is one of the basic tools for saving and delivering information. This makes scene text recognition crucial in many areas of information retrieval, and is also essential for human-computer interaction.

Recently, convolutional networks have made great breakthrough and progress in text recognition task, far beyond traditional approaches. Generally speaking, deep learning based methods need massive data to do the task. Meanwhile, these approaches should avoid the overfitting problem (the accuracy of the model on the training dataset achieve relatively high, but it is low using the test dataset). However, trending benchmark datasets for scene text recognition contains only thousands of images, which means they cannot achieve the standard of latest deep learning based methods. The lack of data problem has one common solution: Transfer Learning. It contains two steps: 1) pre-trains the model on large datasets which contain millions of images such as SynthText90k [2] and ImageNet [1]; 2) fine tunes the pre-trained model on the target dataset with few-shot learning [4] or data augmentation [3]. But this method cannot effectively solve the overfitting problem. Therefore, we came up with a solution that generating enough scene text images with the same distribution as the original training datasets.

Early works define style using similarity measures or local statistics based on the pixel values [18]. However, recent methods that achieve impressive visual quality come from the convolutional neural networks (CNN) for feature extraction [19], [20], [21]. Despite this renewed interest, the actual process of style transfer is based on solving a complex optimization procedure, which can take minutes on today's hardware. A typical speedup solution is to train another neural network that approximates the optimum of the optimization in a single feed-forward pass [22], [23]. While much faster, existing works that use this approach sacrifice the versatility of being able to perform style transfer with any given style image, as the feed-forward network cannot generalize beyond its trained set of images.

Therefore, we proposed a method named Synth-Text Transfer Network to generate images with different text content and the arbitrary texture for specific tasks. In this model, we select an image from the training dataset as the style image and a binary image as the content image which is generated by the corresponding dataset. The Content Image Initialization module that we newly proposed initializes the content image according to the mean grey value of the style image. And then the Style Transfer Network generates a synthetic image that preserves the content word and have similar texture of the referenced style image. In the experiments section, we selected some classic style transfer networks to compare the speed of generating one image and evaluate the appearance of synthetic images.

In summary, our proposed method has two contributions showed as follows: Firstly, we put forward a novel pipeline to generate massive text images based on a few collected scene text images. Secondly, the Synth-Text Transfer Network can produce appealing and adequate synthesized images, which may further make the accuracy of current text recognition models better and avoiding the overfitting problem.

II. RELATED WORK

A. Style transfer

Style transfer networks can be separated into two mainstream categories: Generative Adversarial Networks (GAN) and Neural Style Transfer. Gatys et al. [5] firstly propose neural network method for style transfer, which add the style into the noise image by directly updating pixels in the noise image iteratively. But this method is rather slow and time-consuming. There are a number of researches to enhance this approach from both speed [7] and quality [6]. There are

[1] * Anna Zhu is the Corresponding Author.

978-1-7281-5054-3/19/$31.00 ©2019 IEEE
DOI 10.1109/ICDARW.2019.40073

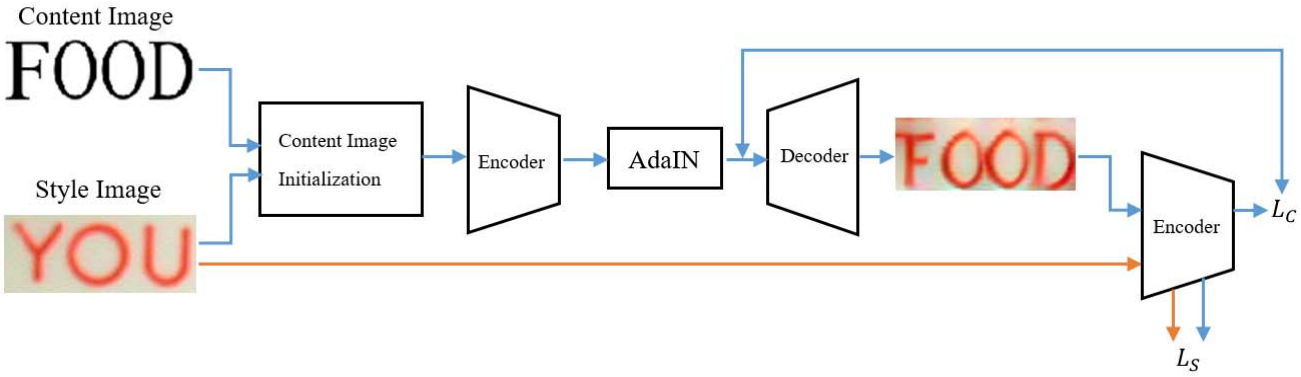

Figure 1. The architecture of the proposed method. Using an Content Image Initialization and an Encoder-AdaIN-Decoder architecture. The first few layers of VGG-19 network is fixed to encode both content and style images. The decoder is learned to invert the AdaIN output to the image spaces. The AdaIN layer is used to conduct style transfer in the feature space. Then we compute the content loss L_C and the style loss L_S by using the same VGG encoder.

some very novel and effective methods appeared recently. Patch Swap [7], AdaIn [8], WCT [9] allow arbitrary style and content images transfer through one feed-forward network. Patch Swap [7] generally refers to swapping each content activation output patch x from feature maps of content image with its closest style patch result y from feature maps of style image, with using the normalized cross-correlations to calculate patch similarity. Adaptive Instance Normalization (AdaIn) [8] creatively produces the styled image by aligning the variance and the channel wise mean of feature map x to match those of y. The main intention of Whitening and Coloring Transforms (WCT) [9] is to transform the feature map of content image directly to align the covariance matrix of the feature map which is the output of style image.

B. Deep generative image modeling

Generative Adversarial Networks is proposed by [10]. It attracts great interest from the field of machine learning and computer vision. Soon after, GAN make a novel development in [11]. It is not only applied in unsupervised learning which produce a noise image with random initialized, but also image-to-image translation tasks like [12] was inspired by GAN method recently. The main purpose is to make the synthesized image look more real. GANs use an adversarial loss and the reconstruction loss between the target images and generated images to produce expected results.

However, our approach is designed to generate a specific number of instances with strongly controllable spatial constraints and precise details, and it is hard to complete with a single GAN. In addition, the training process of GANs is time-consuming to some extent, so we do not want to adopt this method directly in this situation.

Although, neural style transfer methods can be trained unsupervised, we select an image from the target dataset as the style image and initialize the content image to generate the content image in this case, which is fed into the Style Transfer Network as the input.

III. PROPOSED METHOD

In this section, we talk about the proposed method in detail. As shown in Fig. 1, our method is composed of a Content Image Initialization module and an encoder-decoder network. They can be trained jointly in a way of end-to-end as well. The input of our method is one image from the target dataset as style image and a binary image as the content image. Then, the style image and the content image are fed into the Content Image Initialization module to decide the color of the background area and the text area (the background area is white and the text region is black or the background area is black and the text region is white). After that, The STN transfers the content image to one particular style, which is a feedforward transmission network. When it has been trained, it will able to convert the input image into arbitrary style by a single forward pass with a considerable speed.

A. Content Image Initialization Module

Neural style transfer networks typically take an artistic image or texture image as the style image and a text or natural image as the content image. In this situation, we considered an image collected from the standard dataset as the style image which has been cropped by us. And we use a binary image as the content image. Therefore, we utilize an interpretable and relatively easy algorithm to generate a content image with the wanted word. The color of both text and background regions in content image depend on the grey value of style image. The process of content image initialization includes the following four steps:

- Use existent scene text image datasets for text recognition to crop images in suitable size as the style image through the coordinates of characters in their labels.
- Randomly choose some characters to generate binary image, which contains no more than 6 characters, because the cropped text images that we use contain relatively small numbers of characters.
- Split the style image into two unmarked areas using the K-means algorithm. We consider the region which

is located in the top left of the cropped image as the background area and another one as the text part.

- Calculate the mean value of pixels in background region of the style image, which is converted to single channel image firstly. If the result value is greater than 127, then the background of content image should be white. Otherwise, it should be black.

B. Style transfer network

The STN (Style Transfer Network) has an analogous structure with [7]. As shown in Fig. 1, we use a pre-trained VGG-19 network [13] as the encoder to map images from image space into feature space and send the features produced by the encoder to the AdaIN [8] module. After that, we employ a trained invert network as the decoder to invert the feature maps output by AdaIN [8] back to image space.

C. Adaptive Instance Normalization

The definition of Adaptive instance normalization can be found in [8]: AdaIN [8] acquires a style input y and a content input x, and simply aligns the variance and channel wise mean of x to match those of y. Simple as it is, AdaIN [8] adaptively calculates the affine parameters from the input of style image with spending relatively short time. This module simply scale the normalized content input with $\sigma(y)$, and shift it with $\mu(y)$. Similar to IN [14], these statistics are computed across spatial locations.

$$\mathrm{AdaIN}(x, y) = \sigma(y)\left(\frac{x - \mu(x)}{\sigma(x)}\right) + \mu(y) \tag{1}$$

Intuitively, we build up a feature channel that detects a certain style. A feedforward decoder can be used to invert the brush features into the image space while preserving the spatial structure of the content image. The variance of this feature channel is designed to encoder subtler style information. Also, it is transferred to the AdaIN [8] output image and the final synthesized image. All in all, by transferring feature statistics, it carries out style transfer in the feature space, especially channel variance and mean. AdaIN [8] layer plays a similar role to the style swap layer proposed in [7]. The calculation process of style swap layer is quite memory-consuming and time-consuming. On the contrary, the AdaIN [8] layer almost not increase the computational cost, as simple as the IN [16] layer.

D. Loss Function

With the help of pre-trained VGG-19 [13] network, we are able to train the decoder and compute the loss:

$$L = L_C + \lambda L_S \tag{2}$$

which is a weighted sum of the style loss L_S with the weight of λ and the content loss L_C. The content loss is calculated by the Euclidean distance to measure the difference between the final output features and the target features. Our method regards the AdaIN output t (Equ. 6) as the content target, instead of the common feature consequence of the content image. That is the reason why our approach reach convergence very quickly and in accordance with our intend of converting the AdaIN output t as well.

$$L_C = \left\| f\big(g(t)\big) - t \right\|_2 \tag{3}$$

Style loss is designed to compute the mean and variance of generated images, and make it more similar to style image in different layers. Owing to the AdaIN layer merely transmits the standard deviation and mean of the style features, the style loss only depends on these statistic number. Even though the Gram matrix loss is able to yield similar results, we adopt the Instance Normalization [16] statistics which produce theoretically cleaner results. This style loss invented by Li et al. [17] showed as follows:

$$L_S = \sum_{i=1}^{L} \left\| \mu\left(\phi_i\big(g(t)\big)\right) - \mu\big(\phi_i(s)\big) \right\|_2 + \sum_{i=1}^{L} \left\| \sigma\left(\phi_i\big(g(t)\big)\right) - \sigma\big(\phi_i(s)\big) \right\|_2 \tag{4}$$

where each ϕ_i refers to a layer in VGG-19 [13] network. Using the L2 norm to calculate the style loss, combining with relu1_1, relu2_1, relu3_1 and relu4_1 layers that has identical weights.

IV. Experiments

In this section, we will evaluate our proposed network for the performance of scene text image stylization. We first show the implementation details of cropped word images that we made from standard datasets. Besides, we present our experimental results.

A. Architecture Overview

Fig. 1 reveals an overview of our proposed network that relied on the AdaIN [8] module. First, content image is initialized according to the Content Image Initialization module, which determine the color of both background and text region (black and white or opposite).

$$\mathrm{c} = \mathrm{Initialization}(c, s) \tag{5}$$

Style Transfer Network (STN) receives an output of content initialization module c and an input of arbitrary style image s. The synthesized image combines the content of the former (low frequency information) with the style of the latter (high frequency information). We use a simple encoder-decoder architecture where the encoder f is a pre-trained VGG-19 [13] with first few layers fixed, and its structure is represented in next subsection. After encoding the content and style image to feature space, we feed feature maps of style and content image to the AdaIN [8] layer, which aligns the mean and variance of the content feature map with the mean and variance of the style feature map, thus generate the target feature maps t:

Figure 2. Examples generated by our method. Alphabets and numbers in content images are all standard Times New Roman font. On the left is the referenced style images and its corresponding generated images,whose content word are all "food". And the content word on the right is randomly selected.

$$\mathrm{t} = \mathrm{AdaIN}\big(f(c), f(s)\big) \tag{6}$$

Decoder g is randomly initialized and it is trained to map t back to the RGB space, which generate the synthesized image $\mathrm{T}(c, s)$:

$$\mathrm{T}(c, s) = g(t) \tag{7}$$

The encoder and decoder are mirrored and all pooling layers are substituted by the nearest upsampling to decrease the checkerboard effect. We utilize reflection padding in both g and f to prevent border phenomenon. Another critical architectural alternative is whether the decoder ought to adopt batch, instance or none of above. Instance Normalization [16] normalizes each image to a single style. However, Batch Normalization [15] produce a serial of images which should be centered on a single style. Both of them are no need, when we desire the decoder to generate images in various styles. Thus, Normalization layers in the decoder will not be used in this case.

B. Network Details

As shown in the Fig. 1, the architecture of the truncated VGG-19 [13] network used in the experiments is shown in Table I, and the inverse network architecture is exactly symmetry with the VGG-19 [13] network, using the nearest neighbor up sampling. It is possible that better architectures achieve better results, as we did not try many different types of convolutional neural network architectures.

TABLE I. TRUNCATED VGG-19 NETWORK FROM THE INPUT LAYER TO "RELU4_1" (LAST LAYER IN THE TABLE).

Layer Type	Activation Dimensions
Input	H × W× 3
Conv-ReLU	H × W× 64
Conv-ReLU	H × W× 64
MaxPooling	1/2H × 1/2W× 64
Conv-ReLU	1/2H × 1/2W× 128
Conv-ReLU	1/2H × 1/2W× 128
MaxPooling	1/4H × 1/4W× 128
Conv-ReLU	1/4H × 1/4W× 256
Conv-ReLU	1/4H × 1/4W× 256
Conv-ReLU	1/4H × 1/4W× 256
Conv-ReLU	1/4H × 1/4W× 256
MaxPooling	1/8H × 1/8W× 256
Conv-ReLU	1/8H × 1/8W× 512

- The rectified linear unit (ReLU) layer:

$$\mathrm{ReLU}(x) = max\{x, 0\} \tag{8}$$

- Max pooling layers down sample by a factor of 2 by using filter sizes of 2 × 2 and stride of 2.
- Nearest neighbor (NN) up sampling layers up sample by a factor of 2 by using filter sizes of 2 × 2 and stride of 2.

Figure 3. More samples generated by Synth-Text Transfer. The referenced style images from benchmark datasets with uncommon font, complex background texture or low resolution are in the left column. The right colunm is transferred results, which cannot learn the style very well and even reverse the color of background and text region.

C. Results

There are many datasets for text recognition. In our model, we decide to use standard scene text recognition datasets like IC03 (ICDAR 2003) [24] which consists of 1107 cropped text images and 1156 for testing and training, respectively. IC13 (ICDAR 2013) [25] includes1095 images and 849 for testing and training respectively, and both of them are majorly inherited from IC03. And we decided to use these two datasets to evaluate our model.

1) Visual Quality

As shown in Fig. 2, The images generated by our approach have the following features. First of all, it will alternate normal font to arbitrary artistic font from style image. Secondly, our method can complement content image with edge effect from style image. Meanwhile, The AdaIn [8] module concentrate on the similarity of global statistic variable. Also it can render the texture and color of the style image onto the content image and keep the overall text structure unchanged.

However, our method has limitations when the style image has complicated background texture or uncommon font. In Fig. 3, the synthesized image has irregular texture and the background is unlike the referenced style image. When style image has low resolution like the last row in Fig. 3, then the transferred image cannot keep original font and hard to identification. These image are considered as noise data and will influence the accuracy of text recognition.

2) Quantitative Evaluation

Subjective judgment by visual observation is not enough to illustrate the performance of our method. We use structural similarity (SSIM) and cosine similarity to compare the differences between style image and synthesized image. SSIM compares the similarity of images from three aspects: brightness, contrast and structure. Cosine similarity treats image as a vector, and the similarity is represented by calculating the cosine distance between the vectors. Then we calculate the mean value of each similarity, which is presented in Table II.

TABLE II. MEAN SIMILARITY OF DIFFERENT QUANTITATIVE METHOD

Method	SSIM	Cosine Similarity
Mean Value	0.34	0.83

3) Speed Analysis

Time consumption is a significant factor when it comes to generate massive data. Table III displays the time consumption for different methods to generate one image with the size of 48 × 120. Patch Swap [7], WCT [9] and ours are all feedforward style transfer network but they are different in transfer operation. [5] is generates highest quality image but relatively slower, which stand for iteratively style transfer method. As shown in Table II, it is obvious that feed forward style transfer [7], [8], [9] speed much less time than iteratively style transfer [5]. And ours is the fastest way among these methods. Time consuming is the prime factor to evaluate one solution, especially when we need to creating large amount data.

TABLE III. TIME CONSUMPTION FOR DIFFERENT STYLE TRANSFER METHODS TO GENERATE ONE IMAGE

Method	Ours	PatchSwap [7]	WCT [9]	Gatys et al. [5]
Time to generate one image	0.10s	1.04s	0.53s	165s

D. Models for Text Recognition

There are two mainstream text recognition models. The first type is a network, named CRNN [26]. CRNN model have

7 convolution layers, 2 bi-directional LSTM and CTC Loss layers. The second type is named AttentionOCR [27], which is make up of 7 CNN layers, 2 Bi-directional RNN layers using Attention mechanism.

From the Table IV, we can draw a conclusion of the improvement of the Synth-Text Transfer by comparing others. Compared with other data augmentation methods, Synth-Text Transfer uses data generated by random content and arbitrary styles to expand the vocabulary and diversity of the dataset, which significantly improve the accuracy of text recognition. Experiments statistically indicate that ours is superior to other method that can enlarge datasets, especially to data augmentation, which use random rotation and Gaussian noise to original data. Therefore, with the help of creating massive amount of data by style transfer, text recognition network can avoid overfitting problem during training process and make improvement on the accuracy.

TABLE IV. ACCURACY OF ATTENTIONOCR AND CRNN USING DIFFERENT METHODS

Method	Data Aug.	WCT[9]	Ours
AttentionOCR[27]	0.852	0.874	0.898
CRNN[26]	0.861	0.882	0.903

V. CONCLUSION

Our proposed method resolves the lack of in training data problem through synthesizing more samples with arbitrary textures and different content text in a style transfer pipeline. The consequence of our experiments showed that AdaIN is the most proper module for arbitrary style transfer. Synth Text-Transfer's interpretable and flexible pipeline allows users to tightly control the fonts, content text, distortion or random noise. Moreover, it is convenient for us to create training datasets effectively in arbitrary language for text recognition task. If there is not enough training image for this language, it will bring significant improvement. In the future, we will make our effort to investigate new one-stage approaches to generate datasets that can put into use for both text localization and recognition tasks with validate their effectiveness for more languages.

REFERENCES

[1] Deng J, Dong W, Socher R, Li LJ, Li K, Fei-Fei L (2009) Imagenet: a large-scale hierarchical image database Computer vision and pattern recognition, 2009. CVPR 2009. IEEE conference on, pp. 248–255. Ieee.

[2] Jaderberg M, Simonyan K, Vedaldi A, Zisserman A (2014) Synthetic data and artificial neural networks for natural scene text recognition. arXiv:1406.2227.

[3] Liu X, Liang D, Yan S, Chen D, Qiao Y, Yan J (2018) Fots: fast oriented text spotting with a unified network. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 5676–5685.

[4] Ravi S, Larochelle H (2016) Optimization as a model for few-shot learning.

[5] Gatys L, Ecker A, Bethge M (2015) A neural algorithm of artistic style. Nature communications.

[6] Risser E, Wilmot P, Barnes C (2017) Stable and controllable neural texture synthesis and style transfer using histogram losses. arXiv:1701.08893.

[7] Chen TQ, Schmidt M (2016) Fast patch-based style transfer of arbitrary style. arXiv:1612.04337.

[8] Huang X, Belongie SJ (2017) Arbitrary style transfer in real-time with adaptive instance normalization. In: ICCV, pp 1510–1519.

[9] Li Y, Fang C, Yang J, Wang Z, Lu X, Yang MH (2017) Universal style transfer via feature transforms. In: Advances in neural information processing systems, pp 386–396.

[10] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Advances in neural information processing systems, pages 2672–2680, 2014.

[11] Emily L Denton, Soumith Chintala, Rob Fergus, et al. Deep generative image models using a laplacian pyramid of adversarial networks. In Advances in neural information processing systems, pages 1486–1494, 2015.

[12] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-toimage translation with conditional adversarial networks. arXiv preprint arXiv:1611.07004, 2016.

[13] Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image recognition. In: ICLR.

[14] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. In CVPR, 2017.

[15] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In JMLR, 2015. 2.

[16] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. In CVPR, 2017.

[17] Y. Li, N. Wang, J. Liu, and X. Hou. Demystifying neural style transfer. arXiv preprint arXiv:1701.01036, 2017.

[18] V. Kwatra, I. Essa, A. Bobick, and N. Kwatra. Texture optimization for example-based synthesis. ACM Transactions on Graphics (ToG), 24(3):795–802, 2005.

[19] M. Elad and P. Milanfar. Style-transfer via texture-synthesis. arXiv preprint arXiv:1609.03057, 2016.

[20] O. Frigo, N. Sabater, J. Delon, and P. Hellier. Split and match: Example-based adaptive patch sampling for unsupervised style transfer. 2016.

[21] L. A. Gatys, A. S. Ecker, and M. Bethge. A neural algorithm of artistic style. CoRR, abs/1508.06576, 2015.

[22] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual Losses for Real-Time Style Transfer and Super-Resolution. Arxiv, 2016.

[23] D. Ulyanov, V. Lebedev, A. Vedaldi, and V. Lempitsky. Texture Networks: Feed-forward Synthesis ofTextures and Stylized Images. CoRR, 2016.

[24] Karatzas D, Shafait F, Uchida S, Iwamura M, i Bigorda LG, Mestre SR, Mas J, Mota DF, Almazan JA, De Las Heras LP (2013) Icdar 2013 robust reading competition. In: Document analysis and recognition (ICDAR), 2013 12th international conference on. IEEE, pp 1484–1493.

[25] Lucas SM, Panaretos A, Sosa L, Tang A,Wong S, Young R, Ashida K, Nagai H, Okamoto M, Yamamoto H et al (2005) Icdar 2003 robust reading competitions: entries, results, and future directions. Int J Doc Anal Recogn (IJDAR) 7(2-3):105–122.

[26] Shi B, Bai X, Yao C (2017) An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. IEEE transactions on pattern analysis and machine intelligence 39(11):2298–2304.

[27] Shi B, Wang X, Lyu P, Yao C, Bai X (2016) Robust scene text recognition with automatic rectification. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4168–4176.