Font Recognition in Natural Images via Transfer Learning

Yizhi Wang, Zhouhui Lian*, Yingmin Tang, and Jianguo Xiao

Institute of Computer Science and Technology, Peking University

Abstract. Font recognition is an important and challenging problem in areas of Document Analysis, Pattern Recognition and Computer Vision. In this paper, we try to handle a tougher task that aims to accurately recognize the font styles of texts in natural images by proposing a novel method based on deep learning and transfer learning. Major contributions of this paper are threefold: First, we develop a fast and scalable system to synthesize huge amounts of natural images containing texts in various fonts and styles, which are then utilized to train the deep neural network for font recognition. Second, we design a transfer learning scheme to alleviate the domain mismatch between synthetic and real-world text images. Thus, large numbers of unlabeled text images can be adopted to markedly enhance the discrimination and robustness of our font classifier. Third, we build a benchmarking database which consists of numerous labeled natural images containing Chinese characters in 48 fonts. As far as we know, it is the first publicly-available dataset for font recognition of Chinese characters in natural images.

1 Introduction

Font recognition is an important and challenging problem in areas of Document Analysis, Pattern Recognition and Computer Vision. Automatic font recognition can greatly improve the efficiency of many people's work. First and foremost, it helps people (not limited to designers) to know what their favorite font styles are in the text in images they see. Besides, font producers employ it to find copyright infringements by automatic font identification. Moreover, font recognition is useful in improving the accuracy and time of optical text recognition. Actually, it is a specific problem of object detection and classification, in which deep neural networks [4, 5, 9, 13, 14] have made great success. As we know, methods based on deep neural network demand large-scale training data and thus time-consuming manual labeling is typically required. Unlike other object detection and classification tasks, the real-world font annotations are extremely hard to get because large numbers of experts are needed to identify the fonts of texts in images. This problem can be resolved to some extends by synthesizing high-quality images with texts in different fonts. However, there still exist domain mismatch problems between synthesized and real-world text images. In

* Corresponding author. E-mail: lianzhouhui@pku.edu.cn

this paper, we put emphasis on how to synthesize high-quality text images of different fonts and how to conduct effective learning from massive unlabeled images without supervision.

Up to now, many algorithms have been proposed for font recognition, such as modified quadratic discriminant functions (MQDF) [8], wavelet feature descriptors [3], the texture descriptor based on fractal geometry [11], Gaussian mixture models [15], local binary patterns (LBPs) [17], local feature embedding (LFE) [2] and sparse features [16] etc. However, these traditional methods based on handcrafted features are not able to satisfactorily deal with noisy data.

Recently, neurodynamic models have been presented for solving font recognition problem. The DeepFont system proposed in [20] employs a Convolutional Neural Network (CNN) architecture to recognize the font of English text lines. In addition to synthetic data augmentation, a Stacked Convolutional Auto-Encoder (SCAE) trained with unlabeled real-world text images is also utilized to reduce overfitting. Another system reported in [18] is specifically designed to handle the Chinese Character Font Recognition (CCFR) task. They considered CCFR as a sequence classification problem and developed a 2-D long short-term memory (2DLSTM) model to capture a character's trajectory and identify its font style. Although these recently-developed methods could markedly outperform traditional methods, they also have their own shortcomings. For instance, the system proposed in [20] can only deal with alphabetic language systems, such as English, that consist of small number of different characters. For hieroglyph like Chinese with more than 6000 different characters whose geometric structures are often quite complicated, through experiments we found that the SCAE does not work well. For real-world images, the method developed by Tao *et. al* [18] may fail to capture characters' trajectories especially when they are under complicated background and appear with various special effects.

Based on above-mentioned reasons, we select Chinese as one of the text languages in our experiment and propose a transfer learning algorithm to make use of unlabeled text images. Also, our system aims to recognize the font of texts in natural images, instead of synthetic text images adopted in [18]. Moreover, our method can be applied to any other language systems. Experiments conducted on publicly-available databases demonstrate the effectiveness of our system for font recognition in natural images.

2 Overview of the System

As shown in Figure 1, the proposed font recognition system can be built as follows. First, we employ our engine to synthesize huge amounts of natural images containing texts in various fonts and styles and meanwhile information of each text line's font and location is also recorded. Then, by using the location information we train a text localizer to automatically detect texts in images collected from internet. Thus, we have both labeled synthetic text images and unlabeled real-world text images. Afterwards, our initial font classifier base on Convolutional Neural Networks (CNNs) can be obtained by training on labeled synthetic

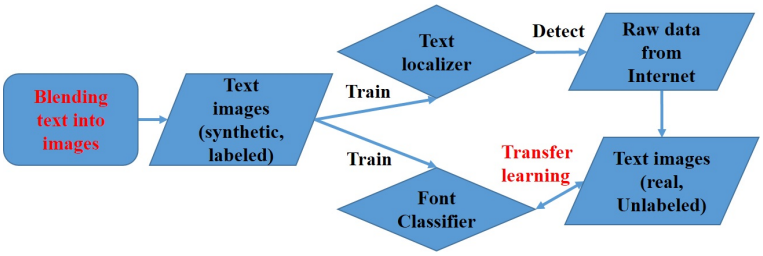

Fig. 1. The pipeline of our font recognition system.

text images. Finally, the proposed transfer learning algorithm is implemented to make use of unlabeled data and improve the classifier's performance on recognizing fonts of texts in natural images. Details of each step in our system will be explained explicitly in the following sections.

3 Synthesizing Text Images

The synthetic text image datasets such as the one described in [20] only contain word-level image regions and simple backgrounds. Thus they are unsuitable to train text detectors and font classifiers for natural images. The method proposed in [6] for generating synthetic text images naturally blends texts in existing natural scenes, using off-the-shelf deep learning and segmentation techniques to align texts to the geometry of a background image and respect scene boundaries. Inspired by the idea of this method, we develop a new system to generate synthetic text images with texts in different fonts and styles in cluttered conditions. As long as we get the TTF (True Type Font) or OTF (Open Type Font) files of some fonts, we can generate nearly real text images in these fonts automatically, along with the font label and location of each character.

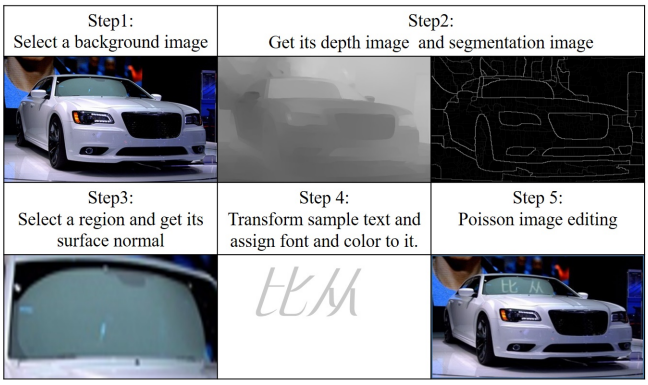

Fig. 2. Main steps to blend texts into a background image.

3.1 Blending Texts into Images

As shown in Figure 2, texts can be naturally blended into a given image by using our system. As we know, texts tend to be contained in well-defined regions in real-world images, instead of crossing strong image discontinuities. For this reason, we segment the image into contiguous regions based on the cues of local color and texture information using the approach presented in [1]. After obtaining segmentation regions, we choose suitable candidates from them for placing texts. Suitable regions should not be too small, should not have an extreme aspect ratio, or have surface normal orthogonal to the view direction.

In natural images, texts are typically painted on top of surfaces. In order to achieve a similar effect in our synthetic data, we need to calculate the local surface normal of the region where we are going to put the text. To get the local surface normal of each contiguous region, we need to obtain an dense pixel-wise depth map, which can be estimated by the CNN model proposed in [10].

Next, the text sample is assigned with a color and a font. The font is chosen randomly from a font list. With this font's library file we can render the glyphs of text. Then the text is assigned with a color which matches well with the background color. Finally, the text is transformed according to the local surface orientation and is blended into the scene using Poisson image editing [12].

Note that, here we record the font type for each character instead of its content. In our experiment, the text's font is chosen randomly from one of the fonts mentioned in Section 6. Beyond that, we introduce more data augmentations to single character images to make them possess more visually similar appearance to real-world data. Details are discussed in the following section.

3.2 Text and Image Source

We employ news corpora of different languages, including Arabic, Bangla, Chinese, Japanese, Korean and English, as our text source. Each time we randomly select some words from the corpus and blend them into a background image.

The background image for blending text can neither be too simple nor too complex. To cover common scenes in our daily lives, we select images from Open Images, an open dataset with 9 million URLs to images that were uploaded by users and have been annotated with labels spanning over 6000 categories. We pick about 30,000 images from the dataset with the labels such as "person" and "natural scenes" instead of "street" etc. whose backgrounds are too cluttered.

3.3 Data Augmentation

The synthetic texts generated by the above method are painted on flat surfaces and match well with the background color. As a matter of fact, texts photographed in natural scenes may be blurred or in uneven illumination. To increase the diversity of our synthetic dataset, we apply some augmentation processing to those synthetic text images, including rotation, changing contrast and brightness, GaussianBlur, adding Gaussian noise, and shear. The parameter of

each effect is selected randomly within certain range and these effects are stacked on one image.

Fig. 3. Texts in different languages blended into natural images.

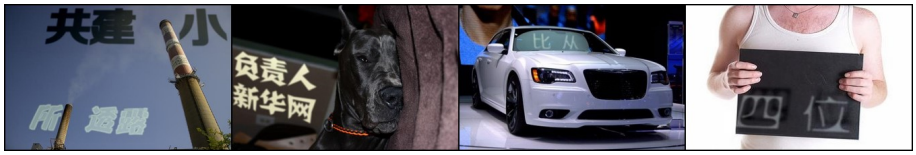

Fig. 4. Blending Chinese texts in different fonts into natural images.

4 Text Detection and Font Recognition

To recognize the fonts of texts in natural images, we first need to accurately localize texts and then correctly identify its font style. Since text spotting techniques have been extensively studied in the last few years [6, 7, 19], our work does not focus on this problem. The synthetic images with text location labels are utilized to train a CTPN (Connectionist Text Proposal Network) [19] as our text localizer, which detects text lines by finding and grouping a sequence of fine-scale text proposals.

The font recognition methods employed in this paper are patch-based CNN models, the same as [20]. We first extract square patches from a word image, and send them to the convolutional neural network. Each extracted patch, whose side length equals the word image's height, contains one or more characters from the word image. For each patch, the network outputs a vector with each element corresponding to the probability it belongs to each font. we average all vectors to determine the final classification result of the word image. Our font classifiers are constructed by modifying two famous CNN models (see Figure 5), i.e., AlexNet and VGG16, proposed in [9] and [14], respectively.

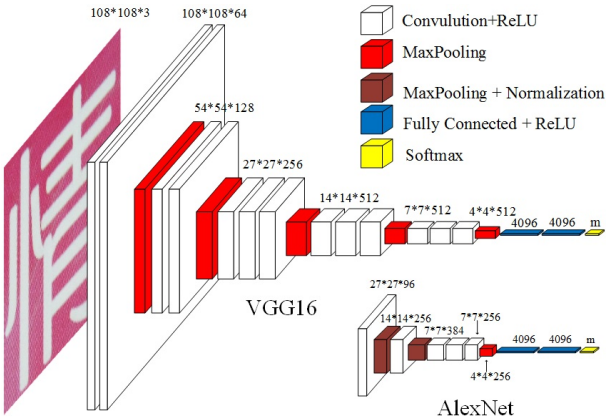

Fig. 5. An illustration of two modified CNN models for font recognition. A 108*108 image patch is put into a network and the network ouputs a m-dimensional probability vector (m represents the number of font classes). AlexNet is a lightweight network with fast training speed while VGG16 is a much deeper and more complex network. As we can see, VGG16 has more convolutional layers and smaller convolution strides than AlexNet.

5 Boosting Accuracy via Transfer Learning

Due to the domain mismatch between synthetic data and real-world data, there still exist a lot of text images that can not be classified correctly. Thus, we want to exploit more information from unlabeled real-world text images. Fortunately, real-world text images are easy to obtain from internet. For example, when we type keywords like "text images" in a search engine, we can get large numbers of images with texts in various kinds of fonts.

The proposed transfer learning algorithm aims to further improve the performance of font recognition for real-world text images by making use of the knowledge we gain from the synthesized data. The key idea of our method is to try to assign the unlabeled text images with correct tags by using our initial font classifiers. Then, these newly-labeled texts images can be adopted with our synthetic images to train our CNN-based classifiers again to make them more robust and effective.

It is worthy of note that how we label text images whose font categories are out of the range we consider. Since some of them have similar font styles, we should label them with the most similar fonts included in our font list. Meanwhile, we discard those images whose font styles are very different against the fonts that we are interested in.

The problem to be addressed can be formulated as follows. Assume that the unlabeled dataset is composed of n text lines (a text line contains one or more words), let x^i be the ith text line of our dataset. The text line x^i contains $t\,(i)$ extracted patches and each patch in x^i is denoted by x^i_j $(1 \leq i \leq n, 1 \leq j \leq t\,(i))$.

Our m-class font classifier takes a single patch as input. After feeding a patch x_j^i into our pre-trained classifier, we get the probability distribution $P\left(x_j^i\right) = \left(P_1\left(x_j^i\right), P_2\left(x_j^i\right), ..., P_m\left(x_j^i\right)\right)$, in which $P_k\left(x_j^i\right)$ means the probability of x_j^i belonging to the font f_k $(1 \leq k \leq m)$, and $\sum_{k=1}^{m} P_k\left(x_j^i\right) = 1$. The classification result of x_j^i is $L_p\left(x_j^i\right) = \arg\max_k P_k\left(x_j^i\right)$. For a given text line, we intend to predict each patch's most probable font or discard it based on the above-mentioned analyses.

Intuitively, the font type of patch image x_j^i can be labeled according to the classification result $L_p\left(x_j^i\right)$. If $P_{L_p\left(x_j^i\right)}\left(x_j^i\right)$ is smaller than a threshold, we discard this patch. However, our pre-trained CNN classifier may make mistakes when handling real-world images it has never seen before. If we label these unknown images with inaccurate classification results, the classifier would be incorrectly supervised which often results in a decline in its performance. **Typically, text images possess a property that characters (or words) in one text line are usually in the same font style.** On account of this, the font labels of patches in one text line identified by the classifier are supposed to be identical. If the labels are not identical, it means that our pre-trained classifier fails to adapt new data. Through a statistical analysis of all patches' classification results in the same text line, we select a representative font style to relabel them. In this manner, the probability of making mistake can be greatly reduced.

Our method is designed as follows: for each font f_k $(1 \leq k \leq m)$, we define two variables to estimate how likely this entire text line x^i is in font f_k. The first variable is $A(k) = \sum_{j=1}^{t(i)} 1\left\{L_p\left(x_j^i\right) = k\right\}$, meaning the times that label k appears in the predicted labels of patches in x^i. The other one is $B(k) = \sum_{j=1}^{t(i)} P_k\left(x_j^i\right)$, denoting the probability of x^i belonging to font f_k accumulated by patches in the text line. We use A as the first sort key and B as the second sort key to rank these fonts $(f_1, f_2, ..., f_m)$ (f_k ranks ahead if $A(k)$ or $B(k)$ is higher). Let the font ranking first be f_l, if $B(l) \geq th * t(i)$ (th is set to 0.4 here), we assign the label of each patch in text line x^i with f_l. The method is actually a voting procedure to decide a text line's font. As a general rule, in turn-based games who wining more rounds wins the game. The pseudo code of this method is shown in Algorithm 1.

6 Experiments

6.1 Font Recognition of Chinese Text Images

To measure the performance of our font recognition method, we need to collect real-world text images containing characters in various font styles as our test dataset. We cooperate with Founder Electronics, one of the world's largest Chinese font producer, to build a large-scale database for font recognition in natural images, named **VFRWild-CHS**[1]. Specifically, the VFRWild-CHS dataset consists of 816 text images captured in natural scenes, from which 6,827 single

[1] http://http://www.icst.pku.edu.cn/zlian/FRWild

Algorithm 1 Predicting Labels for Unlabeled Text Images

Input The font label list $(f_1, f_2, ..., f_m)$, the unlabeled patches set $T = \left\{x_j^i | 1 \leq i \leq n, 1 \leq j \leq t\left(i\right)\right\}$, the pre-trained font classification function P.

Output The most probable font $L\left(x_j^i\right)$ for each patch x_j^i.

for $i = 1 \rightarrow n$ **do**
  for $j = 1 \rightarrow t\left(i\right)$ **do**
    $\left(P_1\left(x_j^i\right), P_2\left(x_j^i\right), ..., P_m\left(x_j^i\right)\right) \leftarrow P\left(x_j^i\right)$
    $L_p\left(x_j^i\right) \leftarrow \arg\max_k P_k\left(x_j^i\right)$
  end for
  for $k = 1 \rightarrow m$ **do**
    $A\left(k\right) \leftarrow \sum_{j=1}^{t(i)} 1\left\{L_p\left(x_j^i\right) = k\right\}$
    $B\left(k\right) \leftarrow \sum_{j=1}^{t(i)} P_k\left(x_j^i\right)$
  end for
  $sortedlist \leftarrow sort(f_1, f_2, ..., f_m), sortkey(A, B)$
  $f_l \leftarrow sortedlist[0]$
  if $B\left(l\right) \geq th * t\left(i\right)$ **then**
    for $j = 1 \rightarrow t\left(i\right)$ **do**
      $L\left(x_j^i\right) \leftarrow f_l$
    end for
  end if
end for

Chinese character images in 48 fonts are extracted and labeled. As we can see from Figure 6, the noisy backgrounds and special effects added artificially make it quite difficult to locate the characters and recognize their font styles.

We prepare three different datasets to train our font classifier. The first dataset, denoted as Syn_Simple, consists of simple synthetic character images with no augmentations (black characters rendered in white background images). The second dataset, denoted as Syn_Blend, is composed of single Chinese character images cropped from synthetic images generated by our text image synthesizing method without data augmentations. Sample images of these two datasets are shown in figure 7. We apply the augmentation methods mentioned in Section 3.3 to Syn_Blend and get a larger dataset Syn_Blend_Aug. Detailed information of these datasets is described in Table 1. Besides, we build a database consisting of more than 200,000 unlabeled images which are collected from internet (See Figure 8).

We compare the performance of our classifiers on the test set which are trained on the above-mentioned training datasets, respectively. Through our experiments, we find that compared to Syn_Simple, the synthetic text images generated by our method can significantly improve the classification performance on our test dataset. As it can be observed from Table 2, the accuracy of our classifier based on AlexNet and VGG16 is very low when trained with Syn_Simple, but improves considerably when trained with Syn_Blend and Syn_Blend_Aug. The result is reasonable because the images in Syn_Blend and Syn_Blend_Aug look more natural than images in Syn_Simple. To sum up, the method mentioned in Section 3 is an effective solution to recognize fonts of texts in natural images.

Fig. 6. Examples of single Chinese character images cropped from text images in our test dataset.

Fig. 7. The Left images (in Dataset Syn_Blend) are synthetic characters generated by our method. The right images (in Dataset Syn_Simple) are corresponding blank-background and no-special-effect characters.

Fig. 8. Some text lines detected by the our text localizer. These images come from Internet and have some special effects and manual designs our synthetic images don't have.

Table 1. Comparison of all datasets adopted in our experiment

name	Source	Label?	Purpose	Size	Class
VFRWild-CHS	Real	Y	Test	6,827	48
Syn_Simple	Syn	Y	Train	324,624	48
Syn_Blend	Syn	Y	Train	474,757	48
Syn_Blend_Aug	Syn	Y	Train	670,873	48
Unlabeled Dataset	Real	N	Train	229,044	N/A

[1] The unlabeled dataset consists of text line images. The others consist of single Chinese character images.

Next, we would like to verify the effectiveness of our transfer learning scheme. As shown in Table 2, our transfer learning algorithm further improves the classifier's performance. On the contrary, if we directly label each character with the predicted result given by the initial classifiers (discard it if the classification probability is lower than th), we witness a decline in classification accuracy: AlexNet top-1 70.30%, VGG16 top-1 83.03% in our experiment. This demonstrates the effectiveness of the proposed transfer learning scheme in font recognition tasks.

Table 2. Our method's performance on VFRWild-CHS

Accuracy / Method / Model	SS	SB	SBA	TL
AlexNet(top-1)	13.85%	69.30%	71.14%	77.75%
AlexNet(top-5)	46.80%	90.75%	91.12%	93.93%
VGG16(top-1)	34.21%	81.93%	84.83%	87.68%
VGG16(top-5)	53.68%	95.22%	96.14%	97.53%

[1] SS, SB and SBA denote our proposed methods trained on Syn_Simple, Syn_Blend and Syn_Blend_Aug datasets, respectively. TL denotes transfer learning.

6.2 Comparison with Other Methods

We compare the performance of our methods with other recently-proposed approaches on VFRWild-CHS. LFE (local feature embedding) introduced in [2] is a representative traditional method which fuses handcrafted local features. DeepFont F introduced in [20] uses synthetic text images with traditional augmentations to train a convolutional neural network. These two methods, along with our SBA method, are supervised learning methods. DeepFont CAEFR [20] and our transfer learning method are both semi-supervised methods exploiting unlabeled real-world images.

For comparative analysis, we employ **the same network architecture** as [20], which is very similar to AlexNet. The difference is that we utilize meth-

ods introduced in Section 3 and 5 to synthesize training data and exploit unlabeled data. As shown in Table 3, our method outperforms other methods. The VFRWild-CHS dataset features noisy backgrounds and distortions, which can not be properly handled by methods of [2] and [20]. Results shown here verify the effectiveness and generality of our methods.

Table 3. Comparison of different methods' performance on VFRWild-CHS

Accuracy / Methods	TOP-1	TOP-5
LFE [2]	32.65%	60.69%
DeepFont F [20]	50.26%	72.93%
SBA(ours)	**70.97%**	**91.05%**
DeepFont CAEFR [20]	55.58%	76.21%
TL(ours)	**77.68%**	**93.97%**

7 Conclusion

In this paper, we developed a new system for accurate font recognition in natural images. One major advantage of our system is that time-consuming and costly font annotations for images in the training dataset can be avoided. On the one hand, by blending text into background images and implementing data augmentations, the synthesized text images look more real and thus large-scale high-quality training data can be automatically constructed for our CNN based font classifiers. On the other hand, the introduction of our transfer learning algorithm exploits a large corpus of unlabeled real-world images and thereby significantly improves the capacity and accuracy of classification. Experimental results on a publicly-available database we built demonstrated that considerable good performance of font recognition in natural images can be obtained by using our system.

References

1. Arbeláez, P., Pont-Tuset, J., Barron, J.T., Marques, F., Malik, J.: Multiscale combinatorial grouping. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 328–335 (2014)
2. Chen, G., Yang, J., Jin, H., Brandt, J., Shechtman, E., Agarwala, A., Han, T.X.: Large-scale visual font recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3598–3605 (2014)
3. Ding, X., Chen, L., Wu, T.: Character independent font recognition on a single chinese character. IEEE Transactions on pattern analysis and machine intelligence 29(2), 195–204 (2007)
4. Girshick, R.: Fast r-cnn. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1440–1448 (2015)

5. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 580–587 (2014)
6. Gupta, A., Vedaldi, A., Zisserman, A.: Synthetic data for text localisation in natural images. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2315–2324 (2016)
7. Jaderberg, M., Simonyan, K., Vedaldi, A., Zisserman, A.: Reading text in the wild with convolutional neural networks. International Journal of Computer Vision 116(1), 1–20 (2016)
8. Kimura, F., Takashina, K., Tsuruoka, S., Miyake, Y.: Modified quadratic discriminant functions and the application to chinese character recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence (1), 149–153 (1987)
9. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. pp. 1097–1105 (2012)
10. Liu, F., Shen, C., Lin, G.: Deep convolutional neural fields for depth estimation from a single image. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5162–5170 (2015)
11. Moussa, S.B., Zahour, A., Benabdelhafid, A., Alimi, A.M.: New features using fractal multi-dimensions for generalized arabic font recognition. Pattern Recognition Letters 31(5), 361–371 (2010)
12. Pérez, P., Gangnet, M., Blake, A.: Poisson image editing. In: ACM Transactions on Graphics (TOG). vol. 22, pp. 313–318. ACM (2003)
13. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in neural information processing systems. pp. 91–99 (2015)
14. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. Computer Science (2014)
15. Slimane, F., Kanoun, S., Hennebert, J., Alimi, A.M., Ingold, R.: A study on font-family and font-size recognition applied to arabic word images at ultra-low resolution. Pattern Recognition Letters 34(2), 209–218 (2013)
16. Song, W., Lian, Z., Tang, Y., Xiao, J.: Content-independent font recognition on a single chinese character using sparse representation. In: Document Analysis and Recognition (ICDAR), 2015 13th International Conference on. pp. 376–380. IEEE (2015)
17. Tao, D., Jin, L., Zhang, S., Yang, Z., Wang, Y.: Sparse discriminative information preservation for chinese character font categorization. Neurocomputing 129, 159–167 (2014)
18. Tao, D., Lin, X., Jin, L., Li, X.: Principal component 2-d long short-term memory for font recognition on single chinese characters. IEEE transactions on cybernetics 46(3), 756–765 (2016)
19. Tian, Z., Huang, W., He, T., He, P., Qiao, Y.: Detecting text in natural image with connectionist text proposal network. In: European Conference on Computer Vision. pp. 56–72. Springer (2016)
20. Wang, Z., Yang, J., Jin, H., Shechtman, E., Agarwala, A., Brandt, J., Huang, T.S.: Deepfont: Identify your font from an image. In: Proceedings of the 23rd ACM international conference on Multimedia. pp. 451–459. ACM (2015)