Exploring Font-independent Features for Scene Text Recognition

Yizhi Wang
Peking University
8 PUBLICATIONS **10** CITATIONS

Zhouhui Lian
Peking University
65 PUBLICATIONS **1,074** CITATIONS

Exploring Font-independent Features for Scene Text Recognition

Yizhi Wang
Wangxuan Institute of Computer Technology, Peking University
China
wangyizhi@pku.edu.cn

Zhouhui Lian*
Wangxuan Institute of Computer Technology, Peking University
China
lianzhouhui@pku.edu.cn

ABSTRACT

Scene text recognition (STR) has been extensively studied in last few years. Many recently-proposed methods are specially designed to accommodate the arbitrary shape, layout and orientation of scene texts, but ignoring that various font (or writing) styles also pose severe challenges to STR. These methods, where font features and content features of characters are tangled, perform poorly in text recognition on scene images with texts in novel font styles. To address this problem, we explore font-independent features of scene texts via attentional generation of glyphs in a large number of font styles. Specifically, we introduce trainable font embeddings to shape the font styles of generated glyphs, with the image feature of scene text only representing its essential patterns. The generation process is directed by the spatial attention mechanism, which effectively copes with irregular texts and generates higher-quality glyphs than existing image-to-image translation methods. Experiments conducted on several STR benchmarks demonstrate the superiority of our method compared to the state of the art.

CCS CONCEPTS

• **Computing methodologies** → **Computer vision**; *Shape modeling*.

KEYWORDS

text recognition, font-independent feature, attention mechanism

ACM Reference Format:
Yizhi Wang and Zhouhui Lian. 2020. Exploring Font-independent Features for Scene Text Recognition. In *Proceedings of the 28th ACM International Conference on Multimedia (MM '20), October 12–16, 2020, Seattle, WA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3394171.3413592

1 INTRODUCTION

Feature representation plays a crucial role in scene text recognition. Before the popularity of deep learning, most methods employ HOG (Histogram of Oriented Gradient) [5] like features for text recognition, but these handcrafted features are not able to satisfactorily deal with noisy data. Afterwards, deep learning techniques have been widely adopted for scene text recognition achieving impressive performance [12, 13]. At present, most prevalent models are devised by combining convolutional neural networks (CNNs) and recurrent neural networks (RNNs), such as [3, 18, 29, 37]. The CNN extracts visual cues within images and the RNN generates character sequences on the basis of CNN features. There are also some works [6, 26] proposing to replace RNNs with fully convolutional networks or self-attention mechanism [30]. In the training phase, these models usually employ the softmax classifier and cross-entropy loss function, following the common object recognition or classification frameworks. However, there is no explicit mechanism in these frameworks to guarantee the removal of font style information from the learnt features. In other words, these frameworks have no clear idea of what the font style and content of a text image are, which weakens their generalization ability. As a result, these models tend to fail in extracting discriminative features for accurately recognizing text images in novel font styles, as is shown in Figure 1. A recent survey paper [1] supports our opinions by reporting that "difficult fonts" remains a challenging and ongoing problem in STR.

In this paper, we investigate how to extract font-independent features from scene texts, so that our model generalizes well on those scene texts in novel fonts. Specifically, we introduce trainable font embeddings, which are concatenated with the scene text features, to generate glyphs in various font styles. The font embeedings are trained to serve as the font features of target glyphs, so as to decrease the unnecessary font style information in scene text features. To accommodate the arbitrary shape and layout of scene texts, we follow the guidance of spatial attention mechanism to generate glyphs one by one. In each generation step, we employ a GAN (Generative Adversarial Network) [8] based generator to translate CNN features from a focusing position into glyphs of multiple fonts. Through our experiments, we find that our technique makes the model less sensitive to font variance, and markedly enhances the STR performance. [1]

2 RELATED WORK

2.1 Regular Scene Text Recognition

Considering there is a considerable body of literature on STR, we only discuss works that are most closely related to ours. [12, 13, 33] are among the early works in using deep convolutional neural networks as feature extractors for STR. [10] and [27] considered words

*Corresponding author

[1]Source code is available at https://actasidiot.github.io/EFIFSTR/

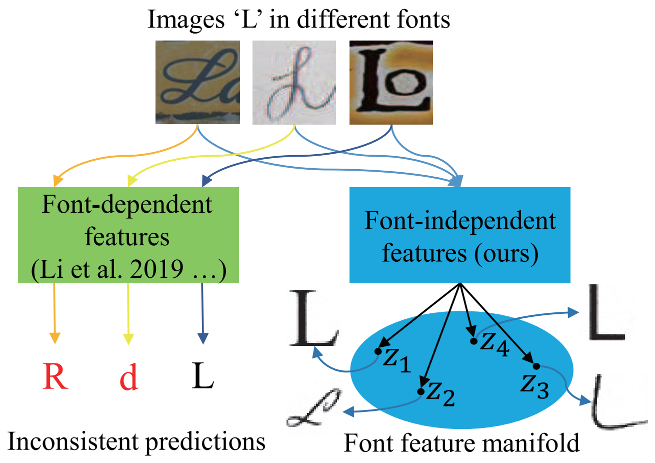
Images 'L' in different fonts
Font-dependent features (Li et al. 2019 …)
Font-independent features (ours)
R
d
L
Inconsistent predictions
z_1
z_2
z_3
z_4
Font feature manifold

Figure 1: Exploring font-independent features makes our model generalize better on text images whose font styles are rarely or never seen in the training dataset. In contrast, most existing methods, such as [18, 29], tend to make wrong predictions on text images in novel font styles. We introduce trainable font embeedings serving as the font features of generated glyphs (shown in the bottom right), so as to decrease the unnecessary font style information in scene text features.

as one-dimensional sequences of varying lengths, and employed RNNs to model the sequences without explicit character separation. A Connectionist Temporal Classification (CTC) layer was adopted to decode the sequences. Inspired by the sequence-to-sequence framework for machine translation, [17] and [28] proposed to recognize text using an attention-based encoder-decoder framework.

2.2 Irregular Scene Text Recognition

[28, 29] proposed an explicit rectification mechanism, which is based on Thin Plate Spline, to rectify distorted and curved texts for recognition. However, many texts in the wild have arbitrary shapes and layouts, which makes it difficult to transform them into horizontal texts through their proposed interpolation methods. [19] presented the Char-Net to detect and rectify individual characters, which, however, requires extra character-level annotations. [4] applied LSTMs [11] in four directions to encode arbitrarily-oriented text. [18] adopted a 2D attention based encoder-decoder network for irregular text recognition inspired by [36]. [26] introduced the Transformer [30] model into STR which can be trained in parallel and good at capturing dependency relationships in sequence.

2.3 Generative Models for Scene Text Recognition

Generative models, such as the Bayesian Network and Generative Adversarial Network (GAN), model the distribution of individual classes while discriminative models learn the (hard or soft) boundary between classes. There is a growing trend that generative models are introduced into scene text recognition in many recent works, such as [7, 20, 34, 39, 40]. Previous to our work, [20, 34] proposed to extract more robust features by mapping scene texts into canonical glyphs. [20] proposed to transform the whole scene text image into corresponding horizontally-written canonical glyphs for promoting feature learning. Through their experiments, the guidance of canonical forms of glyphs is proved to be effective for feature learning in STR. [34] utilized glyphs in four fonts as targets to generate, employing random vectors as the font embeddings. However, these font embeddings were fixed in the training phase and could not appropriately serve as the font features of target glyphs. Without reliable font features as embeddings, the feature extractor of scene texts could be distracted from extracting font-independent features. Besides, both [20] and [34] employed the basic CNN-DCNN (DeConvolutional Neural Network) framework which cannot cope well with irregular texts. Motivated by these analyses, we propose a novel method, i.e., attentional glyph generation with trainable font embeddings, to overcome the deficiencies of above-mentioned methods.

2.4 Multi-font Character Recognition

It is worth noting that most character (text) recognition methods, which are trained with (a sufficient number of) multiple font images, have the same purpose to extract the font-independent feature. [41] proposed the multi-pooling operation for CNN to increase its robustness to some simple font transformation. However, the learning of font-independent features is still very difficult for existing methods, with the character category labels as the only guidance. In Figure 1, the letter 'L' in different font styles causes many existing models to make wrong predictions. A simple idea is providing the font labels of the training images for CNNs to learn. However, the annotation cost will be huge, especially for real-world images. Instead, we instruct CNNs to learn the most "essential" features for reconstructing glyphs in various font styles. We utilize glyphs in **a large number of** fonts as explicit guidance (compared to the training images, the number of these glyph images can be ignored) and achieve significant recognition improvement.

3 METHOD DESCRIPTION

3.1 Overview

We first briefly illustrate the pipeline of proposed model shown in Figure 2. Given an input image $x \in \mathbb{R}^{H_0 \times W_0 \times C_0}$, we first employ a CNN Feature Extractor to extract its visual feature $F(x) \in \mathbb{R}^{H \times W \times C}$. The above-mentioned three dimensions represent height, width and channel number, respectively ($H > 1$ and $W > 1$ for preserving more spatial information of the input image).

Then we send the CNN features $F(x)$ into the Sequence Encoder and Decoder (such as LSTM [11] and Transformer [30]) for sequence modeling. During the decoding step t, the CNN feature maps $F(x)$ and the hidden layer's output of Sequence Decoder $h(x, t)$ are altogether fed into the attention module to calculate the attention mask $M(x, t) \in \mathbb{R}^{H \times W}$:

$$M(x, t) = Attention(F(x), h(x, t)), \tag{1}$$

deciding which position is supposed to be paid more attention to at this moment. Afterwards, by multiplying visual features $F(x)$ by the attention mask over all channels, we get a weighted vector:

$$c(x, t) = M(x, t) \cdot F(x), \tag{2}$$

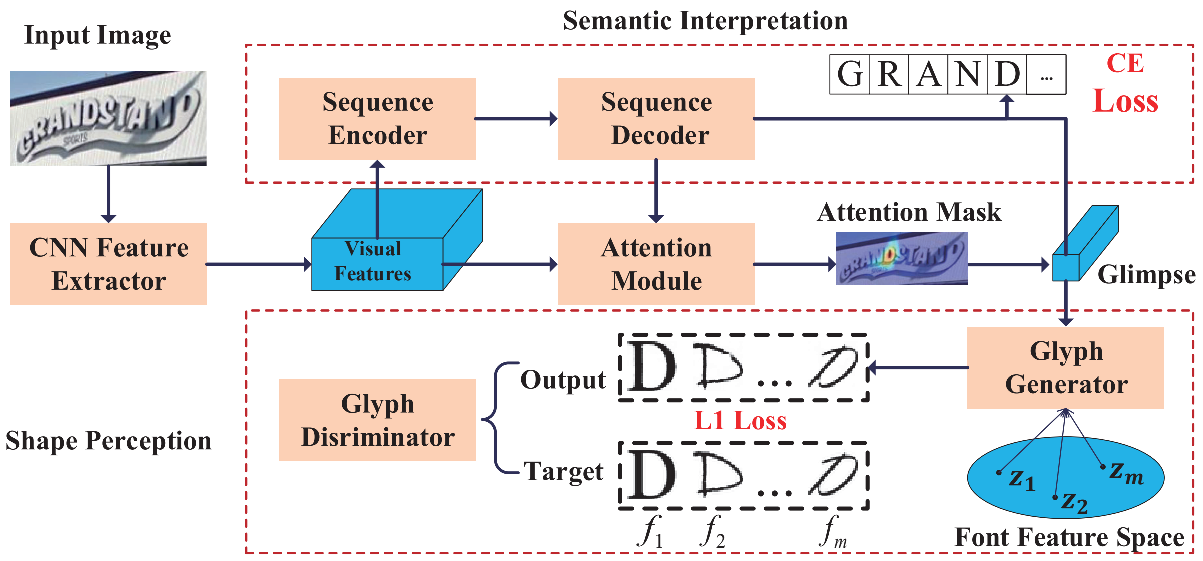

Figure 2: The pipeline of our proposed method.

which is commonly known as “the glimpse vector”.

The next process, where our main contribution lies, is utilizing the glimpse vector to generate its multi-font glyphs and predict character symbols synchronously. A Glyph Generator based on GAN is adopted for generating glyphs:

$$\hat{g}_k(x,t) = GlyphGen([c(x,t); z_k]), 1 \le k \le m, \tag{3}$$

where $z_k \in \mathbb{R}^{C'}$ denotes font embedding, which determines the font style of target glyphs; the square bracket denotes concatenation; m is the number of selected target fonts. Different from Conditional GAN [22], $z_1, z_2, ...z_m$ will be treated as trainable variables to better serve as the font conditions of glyphs. Meanwhile, $h(x,t)$ and $c(x,t)$ are taken for predicting the current-step symbol:

$$p(y_t) = softmax(W_o[h(x,t); c(x,t)] + b_o). \tag{4}$$

The two learning branches work cooperatively and synchronously, contributing to the enhancement of scene text feature learning.

3.2 CNN Feature Extractor

The CNN Feature Extractor of our model is adapted from [29]. The difference is that the vertical dimension of CNN feature maps will not be down-sampled to 1, by keeping the stride 1×1 in the fourth and fifth residual blocks. Then the output feature $F(x)$ has the shape of $H \times W \times C$ where $H = H_0/8$ and $W = W_0/4$.

3.3 Sequence Encoder-Decoder and Attention Module

The attention module intends to output an attention mask according to the hidden layer of LSTM or Transformer and encoded visual features. We try several popular methodologies for calculating the attention mask, including the 2D-Attention mechanism proposed in [18], the traditional 1D-attention employed in [29] and the self-attention mechanism employed in [26]. Based on the experimental results (will be shown in the Section 4.5), we adopt the scheme in [18] which results in the best performance for our framework.

Here we briefly review the 2D-Attention mechanism in [18]. Both the Sequence Encoder and Decoder are 2-layer LSTM models with 512 hidden state size per layer. At each encoding time step, the LSTM Encoder receives one column of $F(x)$ followed by max-pooling along the vertical axis, and updates its hidden state $h_e(x,t)$. The final hidden state of the LSTM Encoder, $h_e(x,W)$, is provided for the LSTM Decoder as the initial state. The attention mask is calculated by:

$$M'_{ij}(x,t) = tanh(\sum_{p,q \in N(i,j)} W_F F_{pq}(x) + W_h h(x,t)), \tag{5}$$

$$M(x,t) = sotfmax(W_M M'), \tag{6}$$

where W_F, W_h and W_M are linear transformations to be learned, and $N(i,j)$ is the neighborhood around position (i,j) (i.e., $i-1 \le p \le i+1, j-1 \le q \le j+1$), $1 \le i \le H, 1 \le j \le W$.

3.4 Multi-font Glyph Generation

The Glyph Generator is composed of a bunch of deconvolution layers, mapping the glimpse vector into glyphs progressively. The highlights of our proposed glyph generation method are two-folded: (1) font embeddings are optimized along with the network's parameters synchronously. They better serve as the font styles of generated glyphs so that the CNN and attention module can concentrate on extracting font-independent features from scene texts. (2) an attentional and hierarchical generation framework is introduced which is effective for generating high-quality glyph images.

Trainable font embeddings. As shown in Equation 3, the font style of generated glyphs is controlled by the font embedding z_k. We want to make sure $F(x)$ has captured enough reliable content features of the input character, so as to let z_k manipulate the font style of generated glyphs. $z_1, z_2, ..., z_m$ are treated as trainable variables and will be fine-tuned according to the font styles of selected

Table 1: The detailed configuration of our Glyph Generator and Discriminator. "$k \times k$ $(de)conv$" means the kernel size of a (de)convolutional layer is k. "s" stands for stride of the (de)convolutional layer. "Out Size" is the size of output feature maps of a block or a (de)convolutional layer (height $\times$ width $\times$ output channels). The layers whose names are in bold receive the skip connections from CNN features.

Layers	Out Size	Configuration
CNN Feature Extractor		
See Section 3.2		
Glyph Generator		
Layer1	$2 \times 2 \times 128$	2×2 $deconv$, s 2×2
Layer2	$4 \times 4 \times 64$	3×3 $deconv$, s 2×2
Layer3	$8 \times 8 \times 32$	3×3 $deconv$, s 2×2
Layer4	$16 \times 16 \times 16$	3×3 $deconv$, s 2×2
Layer5	$32 \times 32 \times 1$	3×3 $deconv$, s 2×2
Glyph Discriminator		
Layer1	$16 \times 16 \times 16$	3×3 $conv$, s 2×2
Layer2	$8 \times 8 \times 32$	3×3 $conv$, s 2×2
Layer3	$4 \times 4 \times 64$	3×3 $conv$, s 2×2
Layer4	$2 \times 2 \times 128$	3×3 $conv$, s 2×2
Layer5	$1 \times 1 \times 128$	3×3 $conv$, s 2×2
Layer6	$1 \times 1 \times 1$	1×1 $conv$, s 1×1

glyphs. We adopt gradient descent to optimize font embeddings, which will be discussed in Section 3.6. Through optimizing the font embeddings, our model concretize various font styles into a meaningful feature space.

Font-aware attentional skip connection. Multi-scale features from CNN are utilized for more accurately reconstructing the glyph's shape, which is shown in Figure 3. Let i $(1 \le i \le l)$ be the index of selected multi-scale features from CNN. The CNN features $F^i(x)$ are first multiplied by the attention mask $M^i(x,t)$, then transformed by fully connection or summation over all channels, afterwards concatenated with the font embedding z_k^i , and finally sent into the Glyph Generator. Note that $M^l(x,t) = M(x,t)$ and $z_k^l = z_k$. $M^i(x,t)$ and z_k^i $(1 \le i \le l-1)$ are up-sampled from the original $M(x,t)$ and z_k, respectively. For the attention masks, the up-sampling operation is implemented by bilinear interpolation into the shape of $H_i \times W_i$, where H_i and W_i are the height and width of $F^i(x)$, respectively. For the font embeddings z_k^i, the up-sampling operation is implemented by duplicating z_k by $2^{l-i} \times 2^{l-i}$ times. The "FC" operation (for $1 \le i \le l-1$) transforms the feature maps of $F^i(x) \bigotimes M^i(x,t)$ into the shape of $2^{l-i} \times 2^{l-i}$ by full connection. The "Sum" operation (for $i = l$) means the feature map of each channel is reduced to 1×1 by summation.

Glyph Discriminator. Following the architecture of GANs, a Glyph Discriminator is introduced to distinguish between generated glyphs and real glyphs. The Glyph Discriminator is a lightweight CNN followed by a fully connected layer with sigmoid activation. Given a real (ground-truth) glyph or a generated glyph as input, it outputs a single value interpreted as the probability of the input glyph being real: $p(y_d = 1|g_k(x,t) \; or \; \hat{g}_k(x,t))$. The Glyph Generator tries to maximize $p(y_d = 1|\hat{g}_k(x,t))$ while the Glyph

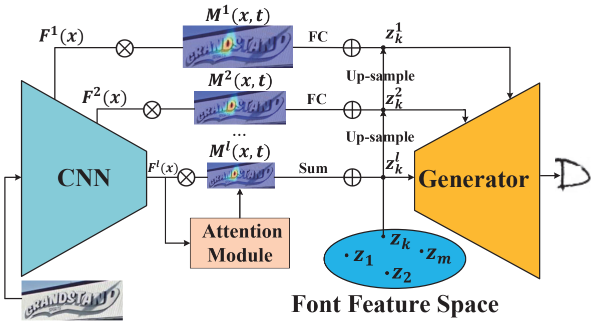

Figure 3: Our proposed attentional glyph generation. Multi-scale features from CNN are first multiplied by the attention masks, then concatenated with the font embeddings and finally sent into the Glyph Generator. $\bigotimes$ denotes point-wise multiplication over all channels and $\bigoplus$ denotes feature concatenation.

Discriminator tries to minimize $p(y_d = 1|\hat{g}_k(x,t))$ and maximize $p(y_d = 1|g_k(x,t))$. Through the adversarial game between them, the quality of generated glyphs can be continuously improved. The detailed configuration of the Glyph Generator and Glyph Discriminator can be found in Table 1.

3.5 Loss Functions

The basic loss function is composed of two items (the cross-entropy loss and the L1 loss):

$$L = -\sum_{t=1}^{T} \log p(y_t|x) + \sum_{t=1}^{T} \|\hat{g}_{i_t}(x,t) - g_{i_t}(x,t)\|, \tag{7}$$

where $y_1, ..., y_i, ..., y_T$ are the ground-truth character labels represented in image x; i_t is a random integer sampled from $\{1, 2, ..., m\}$; $g_{i_t}(x,t)$ is the glyph in font i_t of the t-th character in image x. We find that this sampling method for glyph generation not only reduces the computation cost but also helps to achieve good performance.

When implementing adversarial training, there are two objective functions to be optimized iteratively:

$$\begin{aligned} L_G = &-\sum_{t=1}^{T}[\log p(y_t|x) + \alpha \log p(y_d = 1|\hat{g}_{i_t}(x,t))] \\ &+ \sum_{t=1}^{T} \|\hat{g}_{i_t}(x,t) - g_{i_t}(x,t)\|, \end{aligned} \tag{8}$$

$$L_D = -\alpha \sum_{t=1}^{T}[\log p(y_d = 0|\hat{g}_{i_t}(x,t)) + \log p(y_d = 1|g_{i_t}(x,t))], \tag{9}$$

where α is a hyper-parameter and set as 0.01.

3.6 Optimizing Font Embeddings

We optimize font embeddings with gradient descent, following the equation:

$$\begin{aligned}\frac{\partial L}{\partial z_k} &= \sum_{t=1}^{T} \mathbb{1}\{i_t = k\} \frac{\partial \|\hat{g}_{i_t} - g_{i_t}\|}{\partial z_{i_t}} \\ &= \sum_{t=1}^{T} \mathbb{1}\{i_t = k\} \frac{\partial \|f(W_c c + W_z z_{i_t}) - g_{i_t}\|}{\partial z_{i_t}} \\ &= \sum_{t=1}^{T} \mathbb{1}\{i_t = k\} (W_z f^{'}(W_c c + W_z z_{i_t}) sgn(\hat{g}_{i_t} - g_{i_t})),\end{aligned} \tag{10}$$

where $\hat{g}_{i_t}$, g_{i_t}, and c are the abbreviations of $\hat{g}_{i_t}(x,t)$, $g_{i_t}(x,t)$, and $c(x,t)$, respectively; $\mathbb{1}$, sgn and f are the indicator function, sign function and activation function, respectively; W_c and W_z are the parameters of Glyph Generator which are applied to c and z_{i_t}, respectively; $f^{'}$ is the derivative of f. In practice, the Glyph Generator is composed of a bunch of deconvolutional layers and takes multi-scale features from CNN as input. Here we formulate $\hat{g}_{i_t}$ as $f(W_c c + W_z z_{i_t})$ for brevity. W_c and W_z are also trainable variables and hence the optimization of W_c, W_z and $z_1, ..., z_m$ are alternate. The computation of $\frac{\partial L_G}{\partial z_k}$ is in the same way.

4 EXPERIMENTS

We conduct extensive experiments to verify the effectiveness of our model and compare its performance with other state-of-the-art methods.

4.1 Datasets

There exist two publicly available synthetic datasets that are widely used to train text recognizers: **Syn90k** released by [13] and **SynthText** proposed by [9]. To compensate the lack of spatial characters in Syn90k and SynthText, [18] synthesized additional 1.6 million word images (denoted as **SynthAdd**).

The real-world datasets include **IIIT5K** [23], **SVT** [32], **IC13** [15], **IC15** [14], **SVTP** [24], **CT80** [25] and **COCO-T** [31]. The testing images of these datasets are benchmarks for evaluating the performance of STR models.

As scene texts in novel font styles only make up a small proportion in above-mentioned testing datasets, our method's superiority shown in Table 2 is not that significant. From IIIT5k , IC13, IC15, SVTP and COCO-T datasets , we collect 100 text images with novel or unusual font styles to form a new dataset named as the Novel Font Scene Text (**NFST**) dataset (see Figure 5). Here we give a detailed description of how we select images for NFST. Firstly, we consider a novel font as an eccentric and unusual font. A distribution map of font features of candidate images is estimated to help our selection, which is shown in Figure 4. Specifically, we employ a font recognizer [35] to extract the font features of all candidate text images. The extracted features are then reduced into two dimensions via t-SNE. Afterwards, these text images are attached according to the coordinates of their reduced features. We select the most eccentric cases in the figure, where some examples are marked in red rectangles. Specifically, we mainly select text images which have considerably less neighbors in the figure than others. Meanwhile, we avoid selecting text images which are blurry, distorted, very small, etc. In this manner, we believe that our NFST can be served as a good benchmark to measure the font-robustness of STR models.

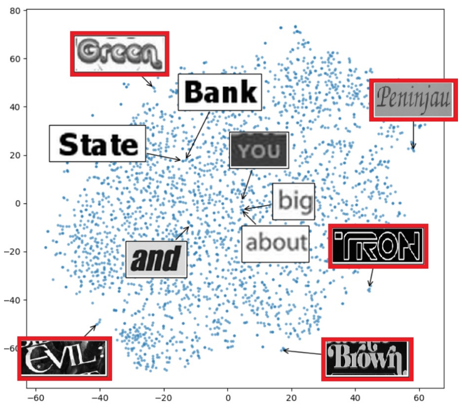

Figure 4: The distribution map of font features of text images in the IIIT5k dataset.

To fully explore font-independent features, we should select those most representative fonts as targets, in terms of their popularity and style diversity, for our model to learn. In our experiments, we choose 325 fonts ($m = 325$) from the **Microsoft Typography**[2] library. Character samples 'A' rendered by these fonts are shown in Figure 9. In our experiments, the image of each glyph is rendered with the font size 64 and resized into the resolution of 32×32.

4.2 Implementation Details

The proposed model is implemented in Tensorflow and trained on two NVIDIA 1080ti GPUs in parallel. The whole network is end-to-end trained using the ADAM optimizer [16]. The learning rate is set to 10^{-3} initially, with a decay rate of 0.9 every 20000 iterations until it reaches 10^{-5}. The input images are resized to the resolution of 48×160. The dimension of font embeddings $C^{'}$ is set as 128.

4.3 Comparison with Other Methods

Higher-quality generated glyphs. We compare the averaged L1 losses of five different generative frameworks, including CNN-DCNN, CNN-LSTM-DCNN, CNN-2DAtt-DCNN (No Skip, FFE), CNN-2DAtt-DCNN (FFE) and CNN-2DAtt-DCNN (TFE). The CNN-DCNN framework is employed in [20]. The CNN-LSTM-DCNN framework is an improved version of CNN-DCNN, where $F(x)$ is first sent into another LSTM encoder and then into DCNN, which is employed in [34]. CNN-2DAtt-DCNN is our proposed attentional glyph generation framework. "No Skip" denotes no skip connection, i.e., only the CNN's last-layer features are utilized to generate glyphs. FFE and TFE denote fixed font embeddings and trainable font embeddings, respectively. For fair comparison, these frameworks are built with the same CNN, DCNN and LSTM configurations proposed in this paper and fed with the same training data (90k+ST+SA+R and glyphs in 325 fonts). Fixed font embeddings are

[2]The font list can be found in https://docs.microsoft.com/en-us/typography/font-list

Table 2: Recognition accuracy (in percentages) on public benchmarks in lexicon-free mode. "90k", "ST" and "SA" denote Synth90k, SynthText and SynAdd datasets, respectively. "ST*" denotes that the character location information is exploited in SynthText. "R" denotes the training datasets of IC13, IC15 and COCO-T. The best performing result for each dataset is shown in bold. Our approach achieves the best recognition performance on most benchmark datasets. [29] have revised their result on SVT from 93.5 to 89.5 (see https://github.com/bgshih/aster).

Method	Training Data	IIIT5k	SVT	IC13	IC15	SVTP	CT80	COCO-T
Jaderberg et al. [12]	90k	-	71.7	81.8	-	-	-	-
Jaderberg et al. [13]	90k	-	80.7	90.8	-	-	-	-
Shi et al. [27] : CRNN	90k	81.2	82.7	89.6	-	-	-	-
Lee et al. [17] : R2AM	90k	78.4	80.7	90.0	-	-	-	-
Liu et al. [20]	90k	89.4	87.1	94.0	-	-	-	-
Cheng et al. [3] : FAN	90k+ST*	87.4	85.9	93.3	70.6	71.5	63.9	-
Liu et al. [19] : Char-Net	90k+ST	92.0	85.5	91.1	74.2	78.9	-	-
Bai et al. [2] : EP	90k+ST	88.3	87.5	94.4	73.9	-	-	-
Cheng et al. [4] : AON	90k+ST	87.0	82.8	-	68.2	73.0	76.8	-
Shi et al. [29] : ASTER	90k+ST	93.4	89.5*	91.8	76.1	78.5	79.5	-
Zhan et al. [38] : ESIR	90k+ST	93.3	90.2	91.3	76.9	79.6	83.3	-
Wang et al. [34]	90k+ST	94.0	-	94.4	-	-	-	-
Ours	90k+ST	94.4	89.8	93.7	75.1	80.2	86.8	-
Li et al. [18] : SAR	90k+ST+SA+R	95.0	91.2	94.0	78.8	**86.4**	**89.6**	66.8
Ours	90k+ST+SA+R	**95.8**	**91.3**	**95.1**	**80.9**	86.0	88.5	**68.4**

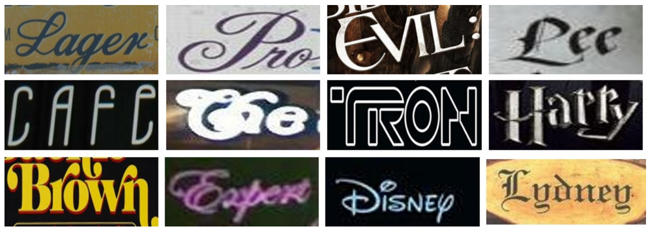

Figure 5: Some samples in our NFST (Novel Font Scene Text) dataset. The text images in NFST possess unusual and novel font styles and are difficult to be recognized by existing methods.

also deployed in CNN-DCNN and CNN-LSTM-DCNN. The curves in the training stage are shown in Figure 7, in which our method significantly outperforms others (lower is better). This demonstrates that our model generates more accurate glyphs under the guidance of trainable font embeddings and 2D attention mechanism. Our model can effectively cope with irregular texts which cannot be handled by [20] (see Figure 6). The generated glyphs directly reflect the quality of extracted CNN features: for the right image in Figure 6, our model correctly recognizes it as "BALLYS" while [20] recognizes it as "setes".

More precise perception on local shapes. Accurate location on the discriminative parts of each single character is also very important for correct recognition. Figure 8 shows that our model achieves a better perception on them after being trained to generate glyphs. Without glyph generation (GG), our model is a 2D attention based text recognizer, sharing similar architecture with [18]. Taking the first case for example, without the guidance of GG, our model pays its attention to the upper part of 'L' in round hand (see the

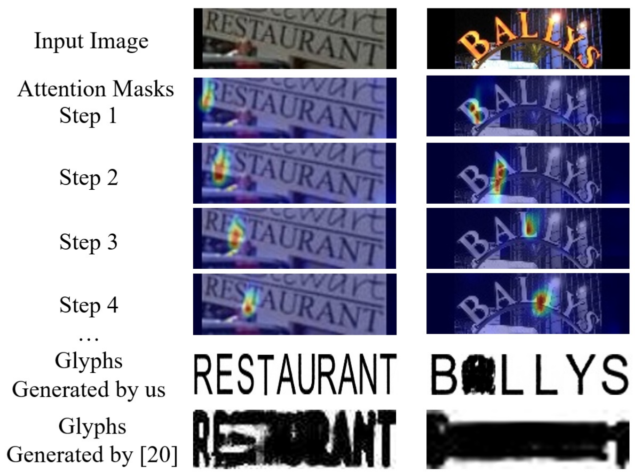

Figure 6: The spatial attention mechanism effectively assists our model with the glyph generation for irregular scene texts. Attention heat maps (visualization of the weights in attention masks) in the first 4 decoding steps are presented in this figure. In contrast, the method proposed by [20] typically fails in coping with them. The glyph images generated by [20] are quoted from their paper. The glyph images (in Arial font) generated by our model are re-scaled and arranged horizontally for a better view.

red positions in heat map), which wrongly recognize it as 'R'. After introducing the generation of multi-font glyphs, our model focuses on the lower part of 'L' and correctly recognize it. The key lies in

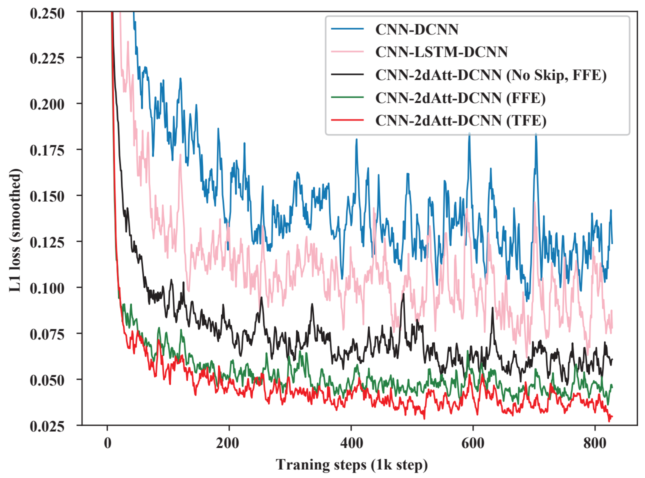

Figure 7: The curves in that L1 losses of different frameworks vary with training steps. Compared to other frameworks that are commonly seen in image-to-image translation, our proposed framework reconstructs more accurate glyphs of scene texts.

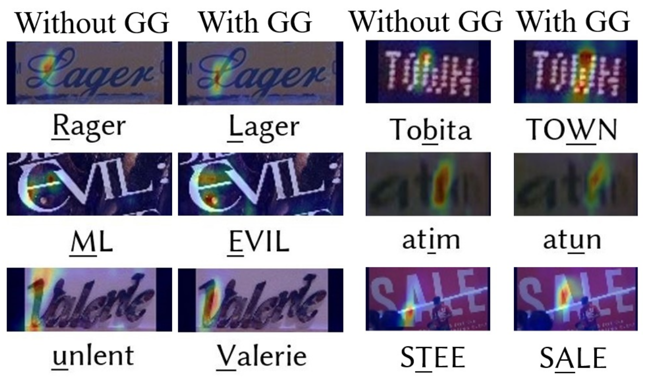

Figure 8: The proposed model shows more accurate perception on the discriminative parts of single character, in addition to the location of different characters. "GG" denotes Glyph Generation. The heat map and underlined character show the weight of attention mask and predicted character label of a certain step.

that our model can successfully exclude undesired nuisance factors from semantic features.

Higher recognition accuracy. Owing to the positive impacts of the proposed approach, our model achieves the best performance on most benchmark datasets (shown in Table 2). On SVTP and CT80, which are relatively small datasets containing 639 and 288 images respectively, our model performs slightly worse than [18]. Generally, our model achieves the state-of-the-art performance compared to other models. Although we adopt the spatial attention module proposed in [18], our proposed generative learning branch further improves the recognition performance. Our method remarkably improves the recognition accuracy on the IC15 dataset, where many scene texts are placed in cluttered environments and possess novel font or writing styles.

Recognizing scene texts in novel styles. In this section we conduct a quantitative experiment on the NFST dataset to demonstrate our model's robustness to the variance of font styles. We compare our method with other two state-of-the-art methods [18, 29] whose codes are publicly available. Our method significantly outperforms others on this dataset (see Table 3), whose robustness to font style variance is proved.

Table 3: Recognition accuracy (in percentages) of different methods on the NFST dataset.

Training data	Ours	SAR [18]	ASTER [29]
90k+ST	**55.0**	45.0	44.0
90k+ST+SA+R	**71.0**	63.0	58.0

4.4 The Optimization Process of Font Embeddings

In this section, we illustrate the optimization process of font embeddings to have a better understanding of how they work. As mentioned in the previous sections, the font embeddings are in charge of controlling the font styles of glyphs. Thereby, the font embeddings need to represent the actual distribution of font styles of selected glyphs. In the top-right corner of Figure 9, we show how the distribution of values in font embeddings varies with the training steps. The font embeddings are randomly initialized, obeying the Gaussian distribution $N(0, 0.01)$. It changes significantly in the early stage to fit the actual distribution of all font styles. Afterwards, it remains relatively stable in the training phase. Figure 9 also demonstrates the visualization of all trained font embeddings that are reduced into two dimensions with PCA (Principal Component Analysis). The glyphs 'A' in different fonts are attached according to the coordinates of corresponding font embeddings. We can observe that similar fonts are located closer while dissimilar fonts are located farther away. Those novel fonts tend to locate at the border of the 2D distribution space. This figure clearly demonstrates that our font embeddings are endowed with actual meanings through training.

4.5 Ablation Study

For the purpose of analyzing the impacts of different modules on the recognition performance, we conduct a series of ablation studies as shown in Table 4. "Att-1 and Att-2" denotes our model which employs the attention mechanism employed in [29] and [26] respectively. "-GD, -GG" denotes our model in which Glyph Generator and Glyph Discriminator are both removed (i.e., no glyph generation). "FFE" denotes our model which employs fixed font embeddings. Without glyph generation, our model's architecture is similar to [18] thus the performance is also nearly the same. After introducing glyph generation, the learnt features of different kinds of characters become less tangled, which is demonstrated in Figure 10 by utilizing t-SNE [21]. In Figure 10, each cluster consists of features of character images in different fonts but the same character category. The clusters are more centralized and outliers

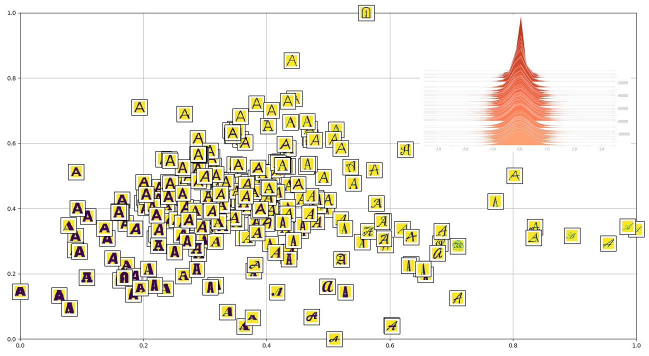

Figure 9: Visualization of all trained font embeddings that are reduced into two dimensions with PCA. The glyphs 'A' in different fonts are attached according to the coordinates of corresponding font embeddings. In the top-right corner, we show how the distribution histogram of values in font embeddings varies with training steps (the vertical axis).

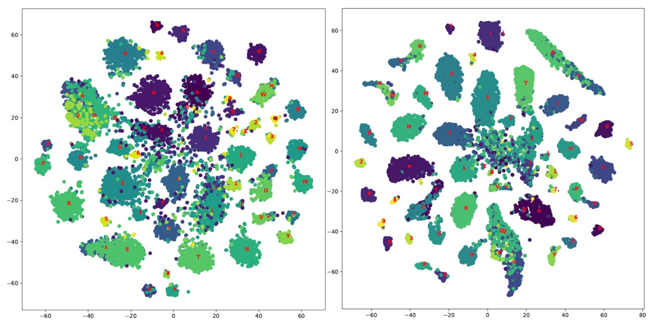

Figure 10: The visualization result of CNN's last-layer features of our model without (left) and with (right) glyph generation. The experiment is conducted on the IIIT5K testing dataset. After introducing glyph generation, the features of different characters with similar shapes are less tangled, such as '9' and 'g'. The feature clusters of different characters are more clearly separated and the ambiguous cases are less scattered.

are significantly reduced in our model, which justifies the claim of font-independent features. Our model understands characters' font styles more deeply and performs better by introducing trainable font embeddings (more results are shown in Figure 11). To investigate how our model's performance varies with the number of fonts (m), we train our model with different settings of m (10, 50, 100, 325). For $m = 10, 50, 100$, we employ the K-means algorithm for the learnt 325 embeddings to find m representative fonts as the new targets for our model to generate.

5 CONCLUSION

In this paper, we proposed the attentional glyph generation with trainable font embeddings for improving the feature learning of scene text recognition. Trainable font embeddings make significant contributions for removing the font information from the CNN features of scene texts. Besides, we proposed to utilize the spatial attention mechanism for glyph generation, which effectively deals with irregular scene texts. Experimental results demonstrated that our model generates higher-quality glyphs, acquires more precise perception on text shapes and achieves higher recognition accuracy compared to the state of the art. In the future, we will try to apply more advanced GAN-related techniques on the Glyph Generator and Discriminator to further improve the quality of generated glyphs. There is also much room for improving the spatial attention mechanism to better locate each character in text images.

Table 4: Results of ablation studies by removing or changing the proposed modules in our model.

Method	IIIT5k	SVT	IC13	IC15	NFST
Att-1	95.4	90.9	94.7	79.8	-
Att-2	95.5	91.2	94.9	80.5	-
-GG, -GD	95.0	91.1	94.1	78.5	-
FFE	95.3	91.1	94.5	80.3	61.0
10fonts	95.5	91.1	94.8	80.1	59.0
50fonts	95.5	91.1	94.8	80.3	63.0
100fonts	95.6	91.2	94.9	80.7	67.0
full	95.8	91.3	95.1	80.9	71.0

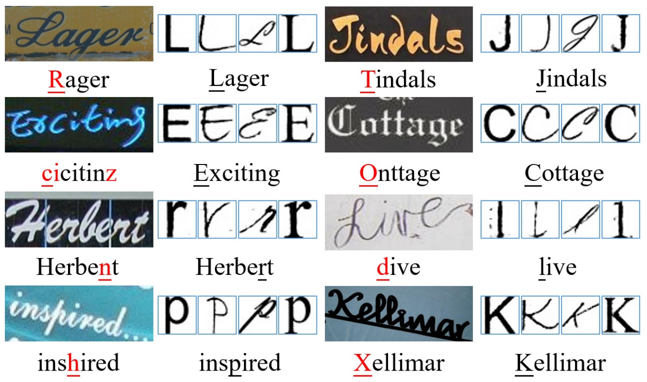

Figure 11: Text samples in the NFST dataset which were wrongly recognized with fixed font embeddings but correctly recognized with trainable font embeddings. The red underline characters show the wrong predictions and next to them our predictions and generated glyphs are demonstrated.

ACKNOWLEDGMENTS

This work was supported by National Natural Science Foundation of China (Grant No.: 61672056 and 61672043), Beijing Nova Program of Science and Technology (Grant No.: Z191100001119077), Center For Chinese Font Design and Research, and Key Laboratory of Science, Technology and Standard in Press Industry (Key Laboratory of Intelligent Press Media Technology).

REFERENCES

[1] Jeonghun Baek, Geewook Kim, Junyeop Lee, Sungrae Park, Dongyoon Han, Sangdoo Yun, Seong Joon Oh, and Hwalsuk Lee. 2019. What is wrong with scene

text recognition model comparisons? dataset and model analysis. In *Proceedings of the IEEE International Conference on Computer Vision.* 4715–4723.

[2] Fan Bai, Zhanzhan Cheng, Yi Niu, Shiliang Pu, and Shuigeng Zhou. 2018. Edit Probability for Scene Text Recognition. In *CVPR.* 1508–1516.

[3] Zhanzhan Cheng, Fan Bai, Yunlu Xu, Gang Zheng, Shiliang Pu, and Shuigeng Zhou. 2017. Focusing Attention: Towards Accurate Text Recognition in Natural Images. In *ICCV.* 5086–5094.

[4] Zhanzhan Cheng, Yangliu Xu, Fan Bai, Yi Niu, Shiliang Pu, and Shuigeng Zhou. 2018. AON: Towards Arbitrarily-Oriented Text Recognition. In *CVPR.* 5571–5579.

[5] Navneet Dalal and Bill Triggs. 2005. Histograms of oriented gradients for human detection. In *CVPR*, Vol. 1. 886–893.

[6] Shancheng Fang, Hongtao Xie, Zheng-Jun Zha, Nannan Sun, Jianlong Tan, and Yongdong Zhang. 2018. Attention and language ensemble for scene text recognition with convolutional sequence modeling. In *ACM MM.* ACM, 248–256.

[7] Dileep George, Wolfgang Lehrach, Ken Kansky, Miguel Lázaro-Gredilla, Christopher Laan, Bhaskara Marthi, Xinghua Lou, Zhaoshi Meng, Yi Liu, Huayan Wang, et al. 2017. A generative vision model that trains with high data efficiency and breaks text-based CAPTCHAs. *Science* 358, 6368 (2017), eaag2612.

[8] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems 27.* 2672–2680.

[9] Ankush Gupta, Andrea Vedaldi, and Andrew Zisserman. 2016. Synthetic Data for Text Localisation in Natural Images. In *CVPR.* 2315–2324.

[10] Pan He, Weilin Huang, Yu Qiao, Chen Change Loy, and Xiaoou Tang. 2016. Reading scene text in deep convolutional sequences. *national conference on artificial intelligence* (2016), 3501–3508.

[11] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9, 8 (1997), 1735–1780.

[12] Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2015. Deep Structured Output Learning for Unconstrained Text Recognition. *ICLR* (2015).

[13] Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2016. Reading Text in the Wild with Convolutional Neural Networks. *IJCV* 116, 1 (2016), 1–20.

[14] Dimosthenis Karatzas, Lluis Gomez-Bigorda, Anguelos Nicolaou, Suman K. Ghosh, Andrew D. Bagdanov, Masakazu Iwamura, Jiri Matas, Lukas Neumann, Vijay Ramaseshan Chandrasekhar, Shijian Lu, Faisal Shafait, Seiichi Uchida, and Ernest Valveny. 2015. ICDAR 2015 competition on Robust Reading. In *ICDAR.* 1156–1160.

[15] Dimosthenis Karatzas, Faisal Shafait, Seiichi Uchida, Masakazu Iwamura, Lluis Gomez i Bigorda, Sergi Robles Mestre, Joan Mas, David Fernandez Mota, Jon Almazàn Almazàn, and Lluís Pere de las Heras. 2013. ICDAR 2013 Robust Reading Competition. In *ICDAR.* 1484–1493.

[16] Diederik P. Kingma and Jimmy Lei Ba. 2015. Adam: A Method for Stochastic Optimization. *ICLR* (2015).

[17] Chen-Yu Lee and Simon Osindero. 2016. Recursive Recurrent Nets with Attention Modeling for OCR in the Wild. In *CVPR.* 2231–2239.

[18] Hui Li, Peng Wang, Chunhua Shen, and Guyu Zhang. 2019. Show, attend and read: A simple and strong baseline for irregular text recognition. In *AAAI*, Vol. 33. 8610–8617.

[19] Wei Liu, Chaofeng Chen, and Kwan-Yee K. Wong. 2018. Char-Net: A Character-Aware Neural Network for Distorted Scene Text Recognition. In *AAAI.* 7154–7161.

[20] Yang Liu, Zhaowen Wang, Hailin Jin, and Ian J. Wassell. 2018. Synthetically Supervised Feature Learning for Scene Text Recognition. In *ECCV.* 449–465.

[21] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, Nov (2008), 2579–2605.

[22] Mehdi Mirza and Simon Osindero. 2014. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784* (2014).

[23] Anand Mishra, Karteek Alahari, and C. V. Jawahar. 2012. Top-down and bottom-up cues for scene text recognition. In *CVPR.* 2687–2694.

[24] Trung Quy Phan, Palaiahnakote Shivakumara, Shangxuan Tian, and Chew Lim Tan. 2013. Recognizing Text with Perspective Distortion in Natural Scenes. In *ICCV.* 569–576.

[25] Anhar Risnumawan, Palaiahankote Shivakumara, Chee Seng Chan, and Chew Lim Tan. 2014. A robust arbitrary text detection system for natural scene images. *Expert Systems With Applications* 41, 18 (2014), 8027–8048.

[26] Fenfen Sheng, Zhineng Chen, and Bo Xu. 2018. NRTR: A No-Recurrence Sequence-to-Sequence Model For Scene Text Recognition. *arXiv preprint arXiv:1806.00926* (2018).

[27] Baoguang Shi, Xiang Bai, and Cong Yao. 2017. An End-to-End Trainable Neural Network for Image-Based Sequence Recognition and Its Application to Scene Text Recognition. *IEEE TPAMI* 39, 11 (2017), 2298–2304.

[28] Baoguang Shi, Xinggang Wang, Pengyuan Lyu, Cong Yao, and Xiang Bai. 2016. Robust Scene Text Recognition with Automatic Rectification. In *CVPR.* 4168–4176.

[29] Baoguang Shi, Mingkun Yang, Xinggang Wang, Pengyuan Lyu, Cong Yao, and Xiang Bai. 2018. ASTER: An Attentional Scene Text Recognizer with Flexible Rectification. *IEEE TPAMI* (2018), 1–1.

[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems.* 5998–6008.

[31] Andreas Veit, Tomas Matera, Lukas Neumann, Jiri Matas, and Serge J. Belongie. 2016. COCO-Text: Dataset and Benchmark for Text Detection and Recognition in Natural Images. *arXiv preprint arXiv:1601.07140* (2016).

[32] Kai Wang, Boris Babenko, and Serge J. Belongie. 2011. End-to-end scene text recognition. In *ICCV.* 1457–1464.

[33] Tao Wang, David J. Wu, Adam Coates, and Andrew Y. Ng. 2012. End-to-end text recognition with convolutional neural networks. In *ICPR.* 3304–3308.

[34] Yizhi Wang, Zhouhui Lian, Yingmin Tang, and Jianguo Xiao. 2019. Boosting scene character recognition by learning canonical forms of glyphs. *IJDAR* (2019), 1–11.

[35] Zhangyang Wang, Jianchao Yang, Hailin Jin, Eli Shechtman, Aseem Agarwala, Jonathan Brandt, and Thomas S Huang. 2015. Deepfont: Identify your font from an image. In *Proceedings of the 23rd ACM international conference on Multimedia.* 451–459.

[36] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. *ICML* (2015), 2048–2057.

[37] MingKun Yang, Yushuo Guan, Minghui Liao, Xin He, Kaigui Bian, Song Bai, Cong Yao, and Xiang Bai. 2019. Symmetry-constrained Rectification Network for Scene Text Recognition. *arXiv preprint arXiv:1908.01957* (2019).

[38] Fangneng Zhan and Shijian Lu. 2019. ESIR: End-To-End Scene Text Recognition via Iterative Image Rectification. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

[39] Fangneng Zhan, Chuhui Xue, and Shijian Lu. 2019. GA-DAN: Geometry-Aware Domain Adaptation Network for Scene Text Detection and Recognition. In *Proceedings of the IEEE International Conference on Computer Vision.* 9105–9115.

[40] Fangneng Zhan, Hongyuan Zhu, and Shijian Lu. 2019. Spatial fusion gan for image synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 3653–3662.

[41] Zhuoyao Zhong, Lianwen Jin, and Ziyong Feng. 2015. Multi-font printed Chinese character recognition using multi-pooling convolutional neural network. In *2015 13th International Conference on Document Analysis and Recognition (ICDAR).* 96–100.