



Scene text recognition based on two-stage attention and multi-branch feature fusion module

Shifeng Xia^{1,2} · Jinqiao Kou³ · Ningzhong Liu^{1,2} · Tianxiang Yin^{1,2}

Accepted: 4 October 2022

© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2022

Abstract

Text image recognition in natural scenes is challenging in computer vision, even though it is already widely used in real-life applications. With the development of deep learning, the accuracy of scene text recognition has been continuously improved. The encoder-decoder architecture is currently a general framework in scene text recognition. With 2D attention on the decoder, better attention can be paid to the position of each character. However, many methods based on the encoder-decoder architecture only adopt the attention mechanism on the decoder. Therefore, their ability to locate characters is limited. In order to solve this problem, we propose a Transformer-based encoder-decoder structure with a two-stage attention mechanism for scene text recognition. At the encoder, a first-stage attention module integrating spatial attention and channel attention is used to capture the overall location of the text in the image, while at the decoder, a second-stage attention module is used to pinpoint the position of each character in the text image. This two-stage attention mechanism can locate the position of the text more effectively and improve recognition accuracy. Also, we design a multi-branch feature fusion module for the encoder that can fuse features from different receptive fields to obtain more robust features. We train the model on synthetic text datasets and test it on real scene text datasets. The experimental results show that our model is very competitive.

Keywords Scene text recognition · Transformer · Attention mechanism · Feature fusion

1 Introduction

Text images, which can be utilized for knowledge transfer and communication, contain more semantic information than other types of images. Therefore, letting computers read the text in the image automatically, i.e., scene text recognition (STR), is a hot research field in computer vision. STR is utilized extensively in a variety of artificial

intelligence applications, including automatic driving, text image translation, and intelligent schooling.

Unlike optical character recognition (OCR), which is used for printed text, STR targets text images in natural scenes. Consequently, STR is more complicated than OCR. Figure 1 shows some common complex problems in STR, including irregular texts, complex backgrounds, different fonts, and low-quality images. Thankfully, training data for STR is readily available, and the amount is staggering. Considering the annotation cost, there are many methods [1–3] to manually synthesize text image data.

Previous approaches can be unified into the framework in Fig. 2, which has four steps: image preprocessing, feature extraction, sequence modeling, and prediction [4]. The image preprocessing step is optional, but it can improve image quality. Wang et al. [5] proposed TextZoom, a scene text super-resolution dataset, and a novel text super-resolution network. This network enhances the resolution of scene text images for better recognition. Shi et al. [6] employed thin-plate spline (TPS) transformation, which is a variant of the spatial transformer network (STN) [7], to learn the transformation parameters of irregular text images, and

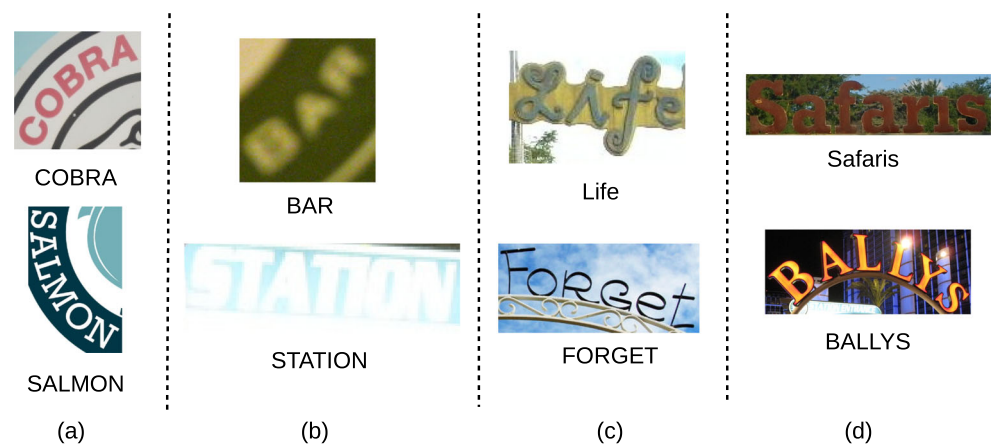
✉ Ningzhong Liu
lnz_nuaa@163.com

¹ College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, 211106, China

² MIIT Key Laboratory of Pattern Analysis and Machine Intelligence Collaborative Innovation Center of Novel Software Technology and Industrialization, Nanjing, 211106, China

³ Beijing Institute of Computer Technology and Application Fangzhou Key Laboratory, Beijing, 100854, China

Fig. 1 Some typical difficult image examples of STR. (a) some irregular text images. (b) some low-quality images. (c) some images with different fonts and colors. (d) some images with complex backgrounds



then applied these parameters to rectify the irregular images. Based on the work of [6], many rectification methods [8–10] have been proposed to more accurately rectify irregular text images. The second stage is feature extraction, and its outcome is the representation of text images. Convolutional neural networks (CNNs), such as VGGNet [11], ResNet [12], GoogLeNet [13], are usually used as feature extractors, and occasionally recurrent neural networks (RNNs), such as GRU [14], LSTM [15], BiLSTM [16], are integrated.

After acquiring the representation, the third step is to construct a sequence model for texts. Instead of separating each character for individual recognition, a sequence model can extract the context information of the character sequence. RNNs and Transformers [17–19] are currently popular sequence models. Compared with RNNs, Transformers have the advantage of being easier to train and, with the help of the mask mechanism, can input text in parallel during the training phase to speed up the training process. The fourth stage is prediction, which predicts the target character sequence of an input text image. Aligning the output characters with the input characters is difficult because character-level annotation information is not available in most cases. Shi et al. [20] proposed a convolutional recurrent neural network (CRNN), which for the first time integrated a connectionist temporal classification (CTC) [21] loss to STR. The CTC

algorithm effectively alleviates the character alignment problem. In addition to CTC, the attention mechanism can also effectively alleviate this problem by learning the weights of the features and aligning the output with the most relevant features..

An encoder-decoder architecture also adheres to the aforementioned four stages. The encoder is used to extract features from the input text image. Then, the decoder receives these features and learns the sequence information of the text. The encoder-decoder architecture is a common framework utilized in a variety of computer vision subfields. The input to both the encoder and the decoder can be various types of data such as text, image, audio, video. The encoder-decoder architecture has the following advantages: 1) its learning and inference procedures are end-to-end; 2) it can learn cross-modal data well; 3) it can handle variable-length sequences efficiently, such as in neural machine translation (NMT), where both the input and output are variable-length text sequences; 4) its decoder side can model conditional probability:

$$p(y_t | v, y_1, \dots, y_{t-1}), \quad (1)$$

where t is the time step, v is the context feature and y_1, \dots, y_{t-1} is the input feature. Based on the above four points, the encoder-decoder architecture is widely used in STR to learn from image to text (cross-modal data) end-to-end.

Existing approaches [6, 22, 23] based on the encoder-decoder architecture adopt only a single attention module on the decoder. These attention modules focus on aligning the output layer features with the features of each character of the text image, and these modules are sometimes not accurate enough. In this paper, we propose a Transformer-based encoder-decoder architecture for STR. The overall structure of the encoder is kept the same as the original Transformer encoder but using the convolutional layer instead of the linear layer. This choice is natural because the feature map of the input text image require the spatial

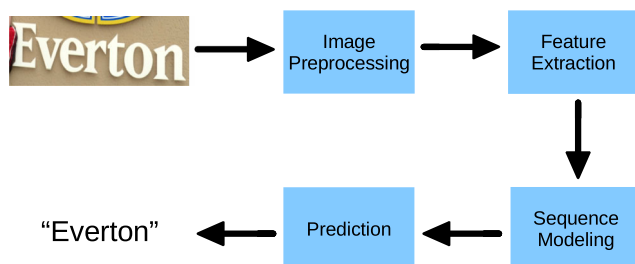


Fig. 2 The typical four steps of STR include image preprocessing, feature extraction, sequence modeling, and prediction. The image preprocessing step is optional

structure to be maintained. The self-attention module [17] is not adopted in the encoder, because this module is not suitable for CNNs. We propose a multi-branch feature fusion module with a designed attention module on each branch. This attention module combines spatial and channel attention, capturing the overall text location information and providing more specific features for the decoder. The decoder is the original Transformer decoder. The attention module we proposed and the self-attention module at the decoder form a two-stage attention architecture to better focus on character regions.

Finally, three contributions of this paper can be summarized as follows: 1) first, we present a multi-branch feature fusion module, which utilizes convolutions with varied kernel sizes to fuse the features of different receptive fields; 2) second, we propose a novel attention module that combines channel attention and spatial attention to enhance the features of the overall text region effectively; 3) third, a simple and efficient Transformer-based encoder-decoder architecture is designed for STR, while the encoder is improved on the basis of the previous two points to better suit our task.

2 Related work

2.1 Scene text recognition

Before deep learning, STR methods use the hand-designed feature, such as histogram of oriented gradient (HOG) [24], scale-invariant feature transform (SIFT) [25] and speeded-up robust features (SURF) [26]. Su et al. [27] used HOG features to construct sequence features of text images. These hand-designed features have an inadequate capacity for representation and are challenging to scale, which hinders the development of STR.

Many STR methods started to use deep features instead of hand-designed features with the development of deep learning. Previous research can be broadly divided into bottom-up and top-down methods [9]. The bottom-up approach first segments each character in the text image, and then recognizes it. Bissacco et al. [28] used an over-segmentation step to divide text lines into multiple segments, each containing at most one character, and then used a deep fully connected network for character classification. Jaderberg et al. [29] proposed a CNN-based structure that incorporates a conditional random field (CRF) graphical model, which predicts the character at each position in the output. Liao et al. [30] designed a semantic segmentation network, called character attention fully convolutional network (CA-FCN), for recognizing arbitrarily shaped text. This network is combined with a word formation module to recognize text and predetermine

the position of each character simultaneously. Wan et al. [31] proposed a semantic segmentation based model, called TextScanner, which generates pixel-wise, multi-channel segmentation maps for character class, position and order. Two shortcomings restrict the development of the bottom-up method: 1) this kind of method requires high accuracy in segmenting or positioning of each independent character; 2) these methods do not exploit the context information of the text.

The top-down method recognizes the text image as a whole without segmenting each character. Jaderberg et al. [32] proposed a method for synthesizing scene text images and treated STR as a multi-class classification problem. However, treating STR as a multi-class classification problem will lose the context information of the text. In the natural language processing (NLP) domain, RNNs can extract the context information of the sequences. Therefore, The RNNs can be introduced into STR. Shi et al. [20] proposed an end-to-end trainable framework using a CNN to extract visual features of text images and an RNN to extract context information of text without relying on character level annotations. RARE [6] is a sequence-to-sequence (Seq2Seq) based model with an attention module for STR. RARE encodes images as features, and aligns the characters by the attention module at the decoder. Since the work of [6], many Seq2Seq-based methods [4, 33–36] have been proposed, which have continuously improved the accuracy of STR. Baek et al. [4] unified the previous Seq2Seq-based methods into a four-stage STR framework, including: transformation, feature extraction, sequence modeling and prediction. Using this framework can greatly improve the accuracy of previous methods. Ma et al. [36] found that the LSTM network in the Seq2Seq framework tends to ignore certain position or visual information. They proposed a Position Information Enhanced Encoder-Decoder (PIEED) framework for STR to address this issue. A Positional Information Enhancement (PIE) module is embedded to make up for the shortcomings of the LSTM network. In recent years, many no-recurrence models [35, 37–39] have been proposed for STR. Sheng et al. [37] proposed a no-recurrence Seq2Seq text recognizer (NRTR) based on Transformer. This model has no recurrences and convolutions at all. Lu et al. [39] proposed the MASTER model, which is a novel encoder-decoder architecture consisting of two parts: a multi-aspect global context attention based encoder and a transformer based decoder. The encoder of the MASTER model can learn different aspects of spatial 2D attention and acts similarly to a multi-head self-attention module. The top-down method is gradually becoming the mainstream method because it can avoid the uncertainty caused by segmenting characters and improve the recognition accuracy by encoding the context information of the text.

2.2 Attention mechanism

When viewing the text as a sequence, one encounters the problem of aligning output features with regions of characters in the text image. There are two main approaches to solving this problem: CTC and attention mechanism. In the CTC-based method [20, 40–43], CNNs and RNNs are used to extract the visual features of the text image and establish the sequence model, respectively, and then CTC loss is used for training. Although the inference time of the CTC-based method is shorter than that of the attention-based method, the accuracy of the former is lower. To address this issue, Hu et al. [43] proposed the guided training of CTC (GTC). This GTC model can learn a better alignment and image feature from stronger attention guidance. The attention mechanism was first proposed by [44] in the field of NMT. STR is similar to NMT in that it “translates” text images into character sequence, so naturally, the attention mechanism can also be used in STR. Cheng et al. [22] proposed an arbitrary orientation network (AON) to directly capture the deep features of irregular texts, which are combined into an attention-based decoder to generate character sequences. Lee et al. [45] proposed a recursive recurrent neural network (R^2AM) with a soft-attention module that directly performs image to character sequence learning. These methods use 1D attention in neural networks, assuming that the text is horizontal or vertical in the image. Hence, they are not suitable for irregular text images where the text is in a 2D plane. Therefore, it is necessary to use 2D attention to capture the features of irregular text images. Li et al. [23] proposed a model composed of a 31-layer ResNet, an encoder-decoder framework based on LSTM, and a customized 2D attention module. Yang et al. [35] used a CNN-based encoder to extract a 2D feature map and feed it into the Transformer decoder, which adopts multi-head self-attention modules. Recently, some methods [34, 38, 39] combine self-attention modules to extract global information of images or texts efficiently. Yu et al. [34] proposed an end-to-end trainable framework named Semantic Reasoning Network (SRN) for STR, in which a Global Semantic Reasoning Module (GSRM) was introduced to capture global semantic information.

3 Method

Figure 3 illustrates our novel encoder-decoder architecture, based on the standard Transformer. ResNet34 is employed as the feature extractor, and other CNNs can also be used. Since STR aims to map an input text image to a sequence of characters, convolutional layers are used in the encoder instead of linear layers to serve images

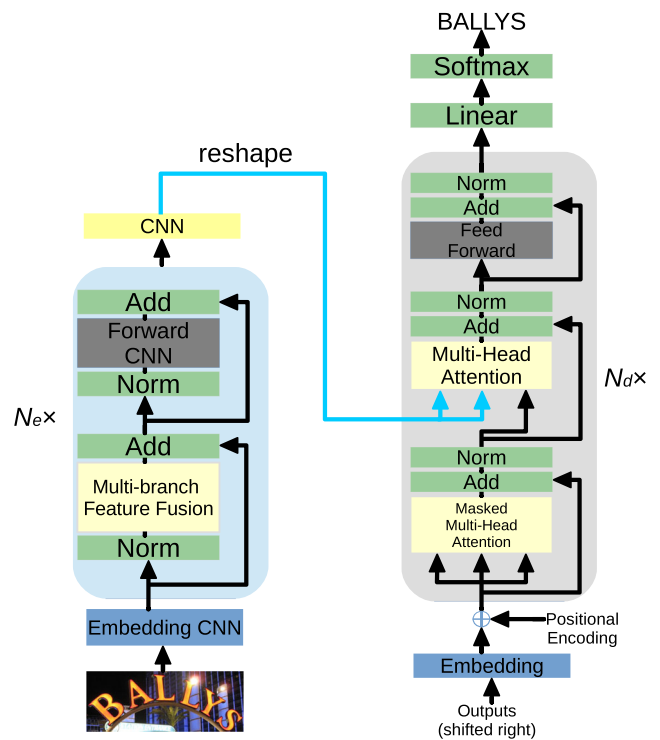


Fig. 3 Proposed Transformer-based encoder-decoder architecture. The left side is the encoder, and the right side is the decoder. $N_e \times$ and $N_d \times$ indicate that there are N_e and N_d layers respectively

better. In contrast, the decoder is the standard Transformer decoder. We propose a multi-branch feature fusion module for the encoder to extract features at different scales using convolutions with different kernel sizes, as shown in Fig. 4. The multi-head self-attention module on the original Transformer encoder does not apply to 2D feature maps but only to 1D feature maps. To this end, on the encoder, we propose an attention module, which combines spatial attention and channel attention. This attention module can roughly notice the position of the whole text in the image, providing better features for the multi-head self-attention module at the decoder. On the decoder, the original Transformer decoder is used, while only some hyperparameters are changed to obtain a higher recognition accuracy. The attention module of the encoder and the multi-head self-attention module of the decoder form a two-stage attention architecture. Next, we present the proposed model in detail, divided into three parts: encoder (Section. 3.1), decoder (Section. 3.2) and prediction (Section. 3.3).

3.1 Encoder

The main goal of the encoder is to extract features, which serve as a representation of the input text image and is one of the inputs to the decoder. Given an input text image

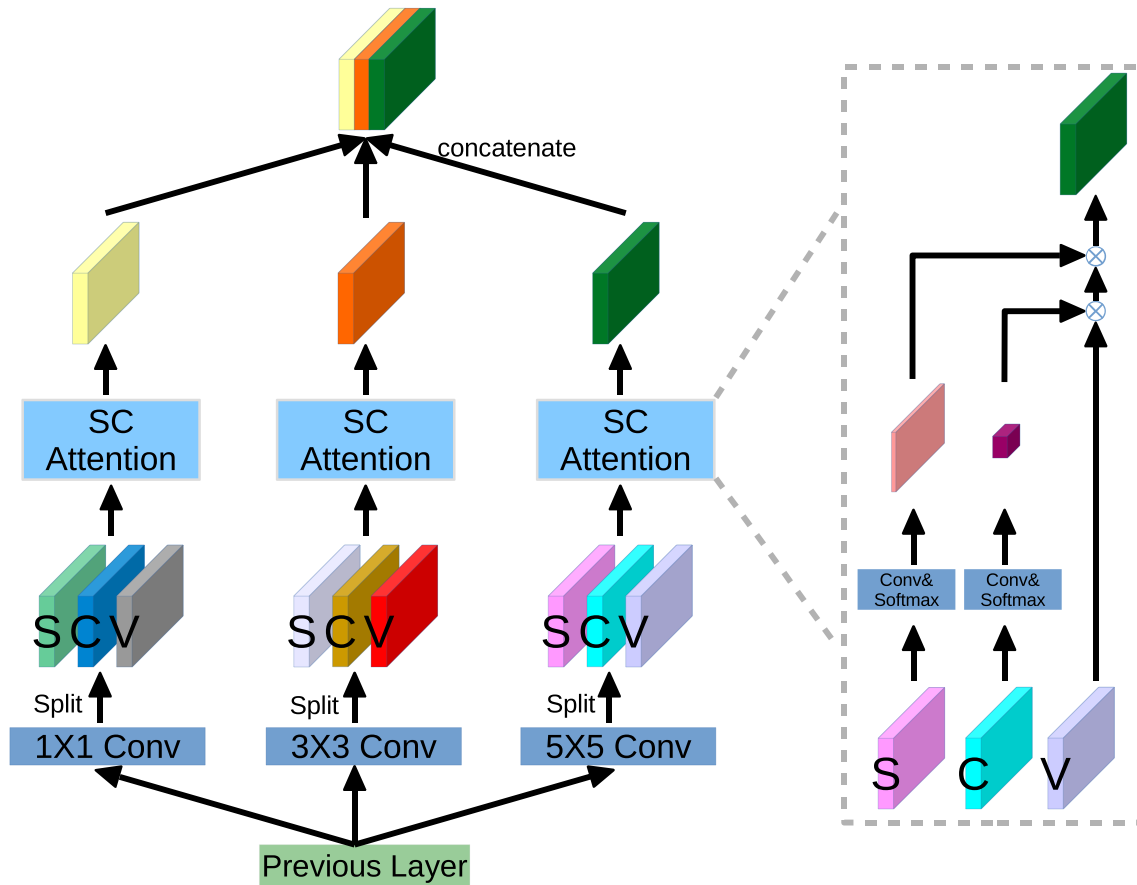


Fig. 4 Proposed SC attention and multi-branch feature fusion module. The left side is the multi-branch feature fusion module, and the right side is the SC attention

$I \in \mathbb{R}^{C \times H \times W}$, where (H, W) is the resolution of the image and C is the number of channels, it first resized into $C \times 64 \times 176$ and then input the resized image into an embedding CNN to extract the embedding feature. Note that C could be 1 for the gray image or 3 for the rgb image. Scaling the input image to 64×176 is based on the following two points: 1) when embedding CNN performs $2 \times$ downsampling, 64 and 176 can be exactly divisible; 2) we expect the output dimension of embedding CNN to have as large a resolution as possible, which is beneficial for 2-dimensional spatial learning at the decoder. ResNet34 is chosen as the embedding CNN, which is modified to fit the input dimension of the encoder. Unlike the decoder, the positional encoding for embedding features is not used because the CNN comes with location information. After that, the embedding feature, which has a dimension of $540 \times 8 \times 22$, is fed into N_e encoder layers. The structure of the encoder layer is shown on the left side of Fig. 3. To keep the same structure as the original Transformer encoder, two residual blocks are used for each encoder layer. Both residual blocks use the pre-norm structure as [46], i.e., the batch normalization is put into the residual block.

This pre-norm structure is conducive to the speed of model convergence. The first residual block contains a multi-branch feature fusion module combined with an attention module we designed, which will be described in detail in the next paragraph. The second residual block contains a simple forward CNN, the structure of which is shown in Table 1. After N_e encoder layers, the output feature with a dimension of $540 \times 8 \times 22$ is fed into a CNN, which has the same configuration as the forward CNN. The role of this CNN is to enhance the feature, and the output dimension is kept constant. Combined with the dimension of the batch size B , the output dimension of the encoder is $B \times 540 \times 8 \times 22$. Then, the output is transposed to obtain a context feature whose dimension is $176 \times B \times 540$. The context feature is used as input for the value and key of the multi-head self-attention module in the decoder.

Multi-branch feature fusion module The purpose of the multi-branch feature fusion module is to extract features from different receptive fields. A feature is input from the previous layer and passes through three branches of 1×1 , 3×3 , and 5×5 convolutions, respectively, as depicted in

Table 1 The configuration of the forward CNN

Layer name	Parameter
Conv	$k : 3 \times 3, s : 1, p : 1, d : 540$
Batch norm	$d : 540$
ReLU	—
Conv	$k : 3 \times 3, s : 1, p : 1, d : 540$
Batch norm	$d : 540$
ReLU	—

k is the kernel size; s is the stride; p is the padding; d is the model dimension

Fig. 4. After that, the feature map of each branch is split into three pieces by channel direction, mimicking the feature map key (K), query (Q), and value (V) in the Transformer model. We maintain the nomenclature of the feature map value (V) same and name the remaining two feature maps spatial (S) and channel (C) because we perform spatial and channel attention on these two features independently. For text images, it is required to extract features from different receptive fields for fusion because the scale sizes of characters are different. The feature map S , C , and V are input to an attention module, and an $180 \times 8 \times 22$ feature map is output. Note that the dimensions of the feature map S , C , and V are all $180 \times 8 \times 22$. By doing this, the feature maps obtained from the three branches in the channel direction are concatenated to form an $540 \times 8 \times 22$ feature map.

Two-stage attention We propose a simple and effective attention module combining spatial attention and channel attention, called SC (‘spatial & channel’) attention, as shown on the right side of Fig. 4. Given a feature map $S \in \mathbb{R}^{180 \times 8 \times 22}$, it is compressed along the channel dimension to 1 using 1×1 convolutions, and then scale it to $[0 - 1]$ using softmax activation function to generate a spatial weight $W_s \in \mathbb{R}^{1 \times 8 \times 22}$. Similarly, given a feature map $C \in \mathbb{R}^{180 \times 8 \times 22}$, it is compressed along spatial dimension to 1 using 8×22 convolutions, and then scale it to $[0 - 1]$ using softmax activation function to generate a channel weight $W_c \in \mathbb{R}^{180 \times 1 \times 1}$. Intuitively, it is better to use the sigmoid activation function because each feature is independent. However, in our experiments, we find that using the sigmoid activation function cause all the weights to be 1, i.e., the attention module is invalid. Instead, a softmax activation function is used to constrain the weights. The feature map V and the channel weight W_c are element-wise multiplied, and the result is element-wise multiplied with the spatial

weight W_s to obtain an $8 \times 22 \times 180$ feature map. The whole attention module can be summarized as:

$$V' = W_s \otimes (W_c \otimes V), \quad (2)$$

where \otimes indicates the element-wise multiplication operation. This SC attention can effectively enhance the overall text region in an image and provide more refined features for the decoder. The spatial and channel attention in the SC attention module process in parallel without precedence, different from [47]. The SC attention module, combined with the self-attention module at the decoder, results in a two-stage attention approach.

3.2 Decoder

The original Transformer decoder is used [17] as our decoder. In the training phase, the input to the decoder is the whole text, beginning with the <SOS> (start of sentence) symbol, but the <EOS> (end of sentence) symbol is not used because there is no requirement to predict the next character of the <EOS> symbol. The Transformer decoder masks all characters following the character at the current time step, enabling parallel training. Extracting embedding features of the input text by the word embedding. The word embedding aims to map a character in the text space to another numerical vector space. In this study, the dimension of the numerical vector space is 540. Since the decoder only uses linear layers, which do not have the ability to obtain the positional information of the text, a positional encoding must be used to utilize the position information of the input text. The same positional encoding as in [17] is used:

$$\begin{aligned} PE_{(pos, 2i)} &= \sin(pos/1000^{2i/d_{model}}) \\ PE_{(pos, 2i+1)} &= \cos(pos/1000^{2i/d_{model}}), \end{aligned} \quad (3)$$

where pos denotes the position and i denotes the dimension. This positional encoding uses sine and cosine functions of different frequencies. The output of the positional encoding and the, so they can be directly summed. The summed feature are input to the decoder, which has N_d layers. A decoder layer has three residual blocks, as shown on the right side of Fig. 3. The first residual block contains a masked multi-head self-attention module, which uses masked mechanism to ensure that subsequent sequences can be masked at the current time step to achieve parallel training. The second residual block contains a multi-head attention module, unlike the self-attention module, where the V and K are from the output of the encoder, i.e., the context feature. Both types of multi-head attention modules mentioned above have a hyperparameter N_h , which is the number of heads. The third residual block contains linear layers, which guarantee a representation capability. All attention modules and linear layers in the

decoder are regularized using Dropout [48]. In contrast, the encoder does not adopt dropout because dropout does not apply to CNNs. Unlike the encoder, which adopts batch normalization [49], the decoder uses layer normalization [50]. Layer normalization is used after each residual block, which is also different from the encoder.

3.3 Prediction

The output of the decoder is passed through a linear layer to obtain logits. After that, the scores are obtained by the softmax activate function. A cross entropy (CE) loss is used to calculate the loss between each time step's output and the ground truth. The final loss is the sum of the losses of all time steps:

$$L_{total} = -\frac{1}{B} \sum_{t=1}^T \sum_{i=1}^B \sum_{j=1}^C p_{tij} \log q_{tij}, \quad (4)$$

where q_i is a score for sample i , p_i is the ground truth for the sample i , B and C denote the number of samples in a batch and the number of classes respectively, and T is the time steps.

The search algorithm is not required during the training and validation stages since the ground truth is known. In the test stage, the character sequence of the current time step is used to predict the next character. The beam search algorithm is used for prediction. This algorithm has a hyperparameter of beam size K . In each time step, the top- K maximum probability sequences is used as candidate sequences until the <EOS> symbol is encountered or the maximum character length is reached.

4 Experiments

4.1 Implementation details

Our method is implemented on Pytorch library [51]. The whole training and test are in RTX 3090 GPU with 24GB memory, and the batch size is 128. The ADADELTA [4] optimizer with $\rho = 0.95$ and $\epsilon = 1 \times 10^{-8}$ is used to train the model 5 epochs. Character classes include 0-9, a-z, <SOS> (start of sentence), <EOS> (end of sentence), and <PAD> (padding), for a total of 39 classes. Masking the <SOS> symbol when calculating L_{total} facilitates training. The beam size K is 2.

4.2 Dataset

We train the model on the synthetic dataset and test it on the real-world dataset, as was the practice with previous methods. Two synthetic datasets are used: Synth90K [32]

and SynthText [3]. Seven real-world datasets include: IIIT 5K-Words (IIIT5K) [56], Street View Text (SVT) [57], ICDAR2003 (IC03) [58], ICDAR2013 (IC13) [59], ICDAR2015 (IC15) [60], Street View Text Perspective (SVTP) [61] and CUTE80 (CUTE) [62]. For testing, only the test set from the above real-world datasets are used.

Synth90K consists of 9 million word box images covering 90k English words. **SynthText** is another synthetically generated dataset that consists of 8-million-word text images.

IIIT5k is obtained from Google Image Search and contains 5000 cropped word images from scene text images and digital images, of which 2000 images are used for training and the remaining 3000 images are used for testing. **SVT** contains 350 images, 100 of which are used for training and 250 for testing. Some of the images in this dataset are heavily corrupted by blur, noise, and low resolution. **IC03** contains 1156 training images and 1110 test images. The images that are less than 3 characters or contain non-alphanumeric characters are removed, leaving 867 images. We follow the configuration in [4] using two versions of this dataset: 860 images and 867 images. **IC13** has 848 images for training and 1095 images for testing. The test images containing non-alphanumeric characters are removed, leaving 1015 images. Like IC03, IC13 is widely used in two versions: 857 images and 1015 images. **IC15** consists of 4468 training images and 2077 test images captured by Google Glasses. Two different versions (1811 images and 2077 images) are used for testing. The 1811-images version removes non-alphanumeric character images and some extremely irregular or low-quality images. **SVTP** is cut from Google Street View and contains 645 test images. **CUTE** is a dataset for curved text detection, which contains 80 images. The text regions are cropped according to the labels, resulting in a STR dataset of 288 images.

4.3 Comparison with other method

In this section, we report comparing our method with other methods on several test datasets. The test datasets can be divided into regular text datasets (IIIT5k, SVT, IC03, and IC13) and irregular text datasets (IC15, SVTP, and CUTE). Table 2. shows the results of comparison between our method and some previous methods. The recognition results using lexicon are not compared because lexicon cannot be obtained in practical applications. The methods in Table 2 are all based on deep learning, while the traditional methods have not been compared because of unfairness. As can be seen from Table 2, our method is competitive with other methods. On the IIIT5k dataset, the recognition accuracy of our method is about 2.8% lower than that of [35]. We

Table 2 Comparisons of our method with other methods on several public datasets

Method	IIIT5K 3000	SVT 647	IC03 860	867	IC13 857	1015	IC15 1811	2077	SVTP 645	CUTE 288
CRNN [20]	78.2	80.8	89.4	—	—	86.7	—	—	—	—
R ² AM [45]	78.4	80.7	88.7	—	—	90.0	—	—	—	—
RARE [6]	81.9	81.9	90.1	—	88.6	—	—	—	71.8	59.2
STAR-Net [40]	83.3	83.6	89.9	—	—	89.1	—	—	73.5	—
FAN [52]	87.4	85.9	—	94.2	—	93.3	70.6	—	—	—
ATR [53]	—	—	—	—	—	—	—	—	75.8	69.3
AON [22]	87.0	82.8	—	91.5	—	—	—	68.2	73.0	76.8
Char-Net [54]	83.6	84.4	91.5	—	90.8	—	—	60.0	73.5	—
ESIR [9]	93.3	90.2	—	—	—	91.3	—	76.9	79.6	83.3
MORAN [55]	91.2	88.3	—	95.0	—	92.4	—	68.8	76.1	77.4
TRBA [4]	87.9	87.5	<u>94.9</u>	94.4	<u>93.6</u>	92.3	77.6	71.8	79.2	74.0
DAN-2D [33]	<u>94.3</u>	89.2	—	95.0	—	93.9	—	74.5	80.0	84.4
TextScanner+90k [31]	93.9	90.1	—	—	—	92.9	—	<u>79.4</u>	<u>84.3</u>	83.3
Yang et al. [35]	94.7	88.9	—	—	—	93.2	<u>79.5</u>	77.1	80.9	<u>85.4</u>
ViTSTR [38]	88.4	87.7	94.7	94.3	93.2	92.4	78.5	72.6	81.8	81.3
PIEED [36]	93.8	90.0	—	—	—	<u>94.0</u>	—	79.9	79.4	<u>85.4</u>
TRAN [10]	<u>94.3</u>	<u>90.7</u>	—	<u>95.3</u>	—	93.6	—	77.7	85.0	81.6
Ours	91.9	91.3	96.5	95.8	96.1	94.2	80.6	78.0	83.4	88.8

The metric is the classification accuracy (%). In each column, the highest accuracy is marked in bold font and the second highest accuracy is shown with an underline

analyzed misclassified images in IIIT5K, some of which contain text with unconventional fonts. The model of [35] is also based on Transformer, which uses a simple CNN on the encoder while not using the attention module. Probably, for this reason, our method has higher recognition accuracy than [35] on all the datasets except IIIT5k. ViTSTR is a simple single stage model based on Vision Transformer (ViT) [19]. The inference speed of ViTSTR is faster than other equivalent parameter-level methods (see Table 3) because it only uses one Transformer encoder layer and

is a non-autoregressive model. However, its recognition accuracy is reduced.

Table 3 shows the comparison with other methods in terms of the number of parameters, inference speed, MACs (multiply-accumulate operations) and average accuracy. For a fair comparison, we re-tested our method and some mainstream methods in the same environment: Intel Core i7-8700 CPU, Nvidia RTX 2080Ti GPU, 64 GB memory, and Pytorch v1.9.1 deep learning library. Furthermore, the methods in Table 3 all adopt the code environment of [4].

Table 3 Comparison with other methods in the following metrics: the number of parameters, inference speed, MACs and average accuracy

Method	Parameters $1 \times 10^6 \downarrow$	Inference speed ms/image \downarrow	MACs $1 \times 10^9 \downarrow$	Average accuracy % \uparrow
CRNN [20]	<u>8.5</u>	3.1	0.69	78.9
R ² AM [45]	2.9	12.5	<u>0.89</u>	78.8
RARE [6]	10.8	11.2	0.92	82.4
STAR-Net [40]	48.7	<u>6.0</u>	5.30	83.3
TRBA [4]	49.6	13.0	5.37	84.0
Yang et al. [35]	75.1	61.5	23.67	<u>87.8</u>
ViTSTR [38]	85.8	7.3	8.72	85.2
Ours	87.2	37.6	12.61	88.4

Average accuracy refers to the average result on all real datasets

The best-performing items in each column are bolded

The underline entries in represent the second best performance in each column

Table 4 Comparison with other methods on training speed

Method	Forward time per batch (ms) ↓	Backward time per batch (ms) ↓	Total time per batch (ms) ↓
CRNN [20]	13.95	26.69	40.64
R ² AM [45]	40.43	83.43	123.86
RARE [6]	<u>29.31</u>	<u>49.17</u>	<u>78.48</u>
STAR-Net [40]	80.73	149.73	230.46
TRBA [4]	92.18	162.99	255.17
Yang et al. [35]	168.63	258.42	427.05
ViTSTR [38]	79.36	134.12	213.48
Ours	80.54	141.98	221.52

The best-performing items in each column are bolded

The underline entries in represent the second best performance in each column

Although our method is not dominant in the number of parameters, its accuracy is the highest among the compared methods. On the other hand, our method's inference speed and MACs are within an acceptable range, enabling real-time inference. Based on the above two points, our method can be used in some scenarios that require high real-time performance and accuracy but is more tolerant of storage space. From Table 3, we find that the inference speed of

non-autoregressive models [20, 38, 40] is much faster than that of autoregressive models ([4, 6, 35, 45] and our method), but the accuracy of the former is lower than that of the latter, which is in line with general cognition. The autoregressive model loops using the decoder to get the next output from the previous output until a <EOS> symbol is encountered. In contrast, the non-autoregressive model outputs all text results at once.

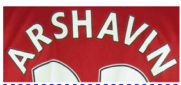






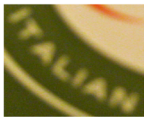
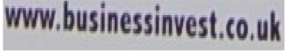
					
CRNN	frshavin	fanies	s_____	fer_____	foreet
STAR-Net	rshavia	fandet	sconieg_	hstale__	fokeet
TRBA	rehavin	famier	scotting	instore	foreet
ViTSTR	rshavin	famill	scottism	historic	foreet
Ours	arshavin	family	scottish	historic	forget
GT	arshavin	family	scottish	historic	forget
	(a)	(b)	(c)	(d)	(e)
					
CRNN	a_____	for__	thr	nombinsinstol_____	
STAR-Net	5_____	e_____	ceriam	uonubusinessnmestrouk	
TRBA	i_____	the____	pallan	unnousinesimestions_	
ViTSTR	a_____	aenoo_	rrcian	whywbusinessivetrcl	
Ours	can_	certain	tracian	nonadvinessiveness____	
GT	club	salmon	italian	wwwbusinessinvestcouk	
	(f)	(g)	(h)	(i)	

Fig. 5 (a)-(e): Examples where our method outperforms several competitive methods. (f)-(i): Failure cases of our method and several competitive methods. The wrong characters are marked in red

Table 5 The performance with or without attention in the spatial and channel

Spatial attention	Channel attention	SVT	CT
		89.7	86.1
✓		90.1	86.4
	✓	90.2	86.8
✓	✓	91.3	88.8

The best-performing items in each column are bolded

We retrained our method and some mainstream methods under the same environment as Table 3 and reported the training speed in Table 4. The batch size is 128. Note that the training time (forward time + backward time) per batch of our method is competitive among autoregressive models ([4, 6, 35, 45] and our method) because the mask mechanism allows the Transformer decoder to input the ground truth text in parallel during training. In contrast, RNN-attention based models [4, 6, 45] require multiple iterations of the decoder in the training and inference stages.

Qualitative results are provided to illustrate the performance of our method (ResNet-Transformer Decoder) compared to several competitive methods, including CRNN (VGG-BiLSTM-CTC), STAR-Net (TPS-ResNet-BiLSTM-CTC), TRBA (TPS-ResNet-BiLSTM-Attention), and ViT-STR (Transformer Encoder). The top of Fig. 5 shows the correct recognition results of our method, and it can be seen that our method outperforms other methods in irregular text images. Our method has certain advantages on some lower-resolution images (see Fig. 5(b)). In addition, our method is more robust when recognizing some images with error-prone characters (see Fig. 5(e)). CRNN is a method without the text rectification module (TPS) and the attention module, so its performance on irregular text is poor. STAR-Net has slightly better results than CRNN because it uses a TPS module, but its results are worse than those of methods with an attention module, such as TRBA. This empirical evidence also illustrates the importance of the attention mechanism in irregular text recognition. Both our method and ViTSTR are based on Transformer, which has a self-attention module. This shows that the self-attention module has an advantage over RNN-attention based methods in

Table 6 The performance with different convolution kernel size combinations

1×1	3×3	5×5	SVT	CT
✓			89.9	86.4
	✓		90.1	87.5
		✓	90.4	87.5
✓	✓	✓	91.3	88.8

The best-performing items in each column are bolded

STR. The bottom of Fig. 5 shows some typical images that are incorrectly recognized by both our method and the previously mentioned methods. Recognizing images where the text is vertical (see Fig. 5(f) and (g)) is a difficult task because the number of such images in the dataset is small, and many methods are designed for irregular images, where the text is not tilted at a large angle. As seen in Fig. 5(i), our method performs poorly on images that contain long text. This is also a common problem with other methods. The main reason for this problem is that few such images are in the dataset.

4.4 Ablation study

Our method has many hyperparameters, such as the number of encoder layers and the number of self-attention heads, affecting the recognition results. In this subsection, ablation experiments are performed to better select the locally optimal hyperparameters and better understand our STR model. When exploring a hyperparameter, the other hyperparameters need to be fixed.

Impact of SC attention The SC attention aims to enhance the text region on the encoder. As shown in Table 5, different combinations of spatial and channel attention are used to verify the effectiveness of the SC attention. In our experiments, the parameters of the *S* or *C* branch are not updated to demonstrate the role of the one remaining attention branch. The experimental results show that using spatial attention or channel attention alone is not as effective as combining the two. Figure 6 visualizes the spatial weights of each branch. The visualization of the channel weights

Fig. 6 Visualization results of SC attention. We visualize the spatial weights on each branch, while do not show the channel weights

Table 7 Performance of different encoder layer numbers

Number of encoder layers	SVT	CT
1	91.3	88.8
2	91.3	87.5
3	88.4	86.8
4	89.1	86.8

The best-performing items in each column are bolded

has no visual meaning, so it is not shown. The spatial attention of each branch can notice different regions in the text image. Therefore, their results can be fused for roughly localizing the entire text region. Some channels in the feature map are strongly associated with the classification results, while some are redundant. The channel attention enhances channels relevant to the results while suppressing redundant ones.

Impact of multi-branch feature fusion module GoogLeNet inspires us to build a multi-branch feature fusion module, which is beneficial to fuse features from different scales. Table 6 illustrates the effectiveness of this module, where the highest recognition accuracy is obtained when the features with different receptive field sizes are fused. We verify the recognition effect when using only one of the 1×1 , 3×3 and 5×5 convolutions. To achieve this, all three branches are set to the same convolution. When using 5×5 convolutions is the most accurate among the three, which is in line with our expectation because a large kernel size can “see” more information of the image. Further, the recognition effect will be improved when the multi-scale features from different convolutions are fused.

Number of encoder and decoder layers The number of the encoder layers or the decoder layers affects the recognition accuracy. Too little will affect the feature extraction, too much will create redundancy. We set the number of encoder layers to 1, 2, 3, and 4, respectively, and keep the number of the decoder layers as 4, as shown in Table 7. The results show that the highest recognition accuracy is achieved when the number of the encoder layers is 1. Our method already has ResNet34 as a feature extractor. The main purpose of

Table 8 Performance of different decoder layer numbers

Number of decoder layers	SVT	CT
2	91.0	87.5
3	89.6	88.5
4	91.3	88.8
5	90.5	86.4

The best-performing items in each column are bolded

Table 9 The performance with different attention head numbers in the decoder

Number of decoder attention heads	SVT	CT
3	89.9	86.4
6	90.5	87.5
9	91.3	88.8
12	91.3	88.5

The best-performing items in each column are bolded

the encoder layer is a coarse enhancement for text regions, so the number of encoder layers should not be too large. Table 8 is about experiments on different numbers of the decoder layers. We set the number of decoder layers to 2, 3, 4, and 5, respectively, while keeping the number of the encoder layers as 1. Our method achieve the best recognition when the number of decoder layers is 4. We do not choose the number of decoder layers greater than 5 because doing this would lead to a large model and reduce the inference speed.

Number of decoder attention heads At the decoder, the multi-head attention module enables the model to focus on the information from different representation subspaces at different positions [17]. A different number of attention heads will affect information extraction. Too little will lead to incomplete information extraction, while too many will lead to redundancy. Because the multi-head attention module is performed in a 540-dimensional subspace, the number of attention heads must be divisible by 540. As is shown in Table 9, our method achieves the highest recognition when the number of attention heads is 9. When the number of attention heads is 3, the recognition effect decreases significantly.

5 Conclusion

In this work, we explore the full potential of the Transformer framework in STR. We propose a Transformer-based encoder-decoder framework for STR. The difference between this framework and the RNN-based STR method is that our framework is more friendly to training and can accelerate training. The encoder of this framework keeps the same structure as the original Transformer encoder. However, it uses the convolutional layers to replace linear ones since the input is an image. The proposed attention module and multi-head attention module form a two-stage attention method. In the first stage, we designed a multi-branch feature fusion module at the encoder and combined it with an attention module. The multi-branch feature fusion module can extract more robust features from

different receptive fields. The proposed attention module can coarsely localize the position of the whole text in the image, preparing for the multi-head self-attention at the decoder. In the second stage, the multi-head self-attention module is used to pinpoint the position of each character in the text image. Finally, we test our approach on several public datasets, showing that our approach is highly competitive. However, the recognition results on long character sequence and low-quality images are not satisfactory; this will be the focus of our future work.

References

1. Yim M, Kim Y, Cho H-C, Park S (2021) Synthtigger: synthetic text image generator towards better text recognition models. In: International conference on document analysis and recognition (ICDAR). Springer, pp 109–124
2. Liao M, Song B, Long S, He M, Yao C, Bai X (2020) Synthtext3d: synthesizing scene text images from 3d virtual worlds. *Sci China Inform Sci* 63(2):1–14
3. Gupta A, Vedaldi A, Zisserman A (2016) Synthetic data for text localisation in natural images. In: Conference on computer vision and pattern recognition (CVPR), pp 2315–2324
4. Baek J, Kim G, Lee J, Park S, Han D, Yun S, Oh SJ, Lee H (2019) What is wrong with scene text recognition model comparisons? dataset and model analysis. In: International conference on computer vision (ICCV), pp 4715–4723
5. Wang W, Xie E, Liu X, Wang W, Liang D, Shen C, Bai X (2020) Scene text image super-resolution in the wild. In: European conference on computer vision (ECCV). Springer, pp 650–666
6. Shi B, Wang X, Lyu P, Yao C, Bai X (2016) Robust scene text recognition with automatic rectification. In: Conference on computer vision and pattern recognition (CVPR), pp 4168–4176
7. Jaderberg M, Simonyan K, Zisserman A, Kavukcuoglu K (2015) Spatial transformer networks. In: Advances in neural information processing systems (neurIPS), pp 2017–2025
8. Xu C, Wang Y, Bai F, Guan J, Zhou S (2022) Robustly recognizing irregular scene text by rectifying principle irregularities. In: Proceedings of the IEEE/CVF winter conference on applications of computer vision, pp 3061–3068
9. Zhan F, Lu S (2019) Esir: end-to-end scene text recognition via iterative image rectification. In: Conference on computer vision and pattern recognition (CVPR), pp 2059–2068
10. Wu L, Xu Y, Hou J, Chen CP, Liu C-L (2022) A two-level rectification attention network for scene text recognition. *IEEE Trans Multimed*
11. Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image recognition. In: International conference on learning representations (ICLR)
12. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Conference on computer vision and pattern recognition (CVPR), pp 770–778
13. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A (2015) Going deeper with convolutions. In: Conference on computer vision and pattern recognition (CVPR), pp 1–9
14. Cho K, van Merriënboer B, Gulcehre C, Bougares F, Schwenk H, Bengio Y (2014) Learning phrase representations using rnn encoder-decoder for statistical machine translation. In: Conference on empirical methods in natural language processing (EMNLP)
15. Hochreiter S, Schmidhuber J (1997) Long short-term memory. *Neural Computat* 9(8):1735–1780
16. Graves A, Liwicki M, Fernández S, Bertolami R, Bunke H, Schmidhuber J (2008) A novel connectionist system for unconstrained handwriting recognition. *IEEE Trans Pattern Anal Mach Intell (PAMI)* 31(5):855–868
17. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I (2017) Attention is all you need. In: Advances in neural information processing systems (neurIPS)
18. Touvron H, Cord M, Douze M, Massa F, Sablayrolles A, Jégou H (2021) Training data-efficient image transformers & distillation through attention. In: International conference on machine learning. PMLR, pp 10347–10357
19. Dosovitskiy A, Beyer L, Kolesnikov A, Weissenborn D, Zhai X, Unterthiner T, Dehghani M, Minderer M, Heigold G, Gelly S et al (2020) An image is worth 16x16 words: transformers for image recognition at scale. In: International conference on learning representations (ICLR)
20. Shi B, Bai X, Yao C (2016) An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE Trans Pattern Anal Mach Intell (PAMI)* 39(11):2298–2304
21. Graves A, Fernández S, Gomez F, Schmidhuber J (2006) Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In: International conference on machine learning (ICML), pp 369–376
22. Cheng Z, Xu Y, Bai F, Niu Y, Pu S, Zhou S (2018) Aon: towards arbitrarily-oriented text recognition. In: Conference on computer vision and pattern recognition (CVPR), pp 5571–5579
23. Li H, Wang P, Shen C, Zhang G (2019) Show, attend and read: a simple and strong baseline for irregular text recognition. In: AAAI conference on artificial intelligence (AAAI), vol 33, pp 8610–8617
24. Dalal N, Triggs B (2005) Histograms of oriented gradients for human detection. In: Conference on computer vision and pattern recognition (CVPR). Ieee, vol 1, pp 886–893
25. Lowe DG (2004) Distinctive image features from scale-invariant keypoints. *Int J Comput Vis (IJCV)* 60(2):91–110
26. Bay H, Ess A, Tuytelaars T, Van Gool L (2008) Speeded-up robust features (surf). *Comput Vis Image Understand* 110(3):346–359
27. Su B, Lu S (2014) Accurate scene text recognition based on recurrent neural network. In: Asian conference on computer vision (ACCV). Springer, pp 35–48
28. Bissacco A, Cummins M, Netzer Y, Neven H (2013) Photoocr: reading text in uncontrolled conditions. In: International conference on computer vision (ICCV), pp 785–792
29. Jaderberg M, Simonyan K, Vedaldi A, Zisserman A (2015) Deep structured output learning for unconstrained text recognition. In: ICLR
30. Liao M, Zhang J, Wan Z, Xie F, Liang J, Lyu P, Yao C, Bai X (2019) Scene text recognition from two-dimensional perspective. In: AAAI conference on artificial intelligence (AAAI), vol 33, pp 8714–8721
31. Wan Z, He M, Chen H, Bai X, Yao C (2020) Textscanner: reading characters in order for robust scene text recognition. In: AAAI conference on artificial intelligence (AAAI), vol 34, pp 12120–12127
32. Jaderberg M, Simonyan K, Vedaldi A, Zisserman A (2014) Synthetic data and artificial neural networks for natural scene text recognition. In: Workshop on deep learning, NIPS
33. Wang T, Zhu Y, Jin L, Luo C, Chen X, Wu Y, Wang Q, Cai M (2020) Decoupled attention network for text recognition. In: AAAI conference on artificial intelligence (AAAI), vol 34, pp 12216–12224

34. Yu D, Li X, Zhang C, Liu T, Han J, Liu J, Ding E (2020) Towards accurate scene text recognition with semantic reasoning networks. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp 12113–12122
35. Yang L, Wang P, Li H, Li Z, Zhang Y (2020) A holistic representation guided attention network for scene text recognition. *Neurocomputing* 414:67–75
36. Ma X, He K, Zhang D, Li D (2021) Pieed: position information enhanced encoder-decoder framework for scene text recognition. *Appl Intell*:1–10
37. Sheng F, Chen Z, Xu B (2019) Nrtr: a no-recurrence sequence-to-sequence model for scene text recognition. In: *International conference on document analysis and recognition (ICDAR)*. IEEE, pp 781–786
38. Atienza R (2021) Vision transformer for fast and efficient scene text recognition. In: *International conference on document analysis and recognition*. Springer, pp 319–334
39. Lu N, Yu W, Qi X, Chen Y, Gong P, Xiao R, Bai X (2021) Master: multi-aspect non-local network for scene text recognition. *Pattern Recogn* 117:107980
40. Liu W, Chen C, Wong K-YK, Su Z, Han J (2016) Star-net: a spatial attention residue network for scene text recognition. In: *British machine vision conference (BMVC)*, vol 2, p 7
41. Qi X, Chen Y, Xiao R, Li C-G, Zou Q, Cui S (2019) A novel joint character categorization and localization approach for character-level scene text recognition. In: *2019 International conference on document analysis and recognition workshops (ICDARW)*. IEEE, vol 5, pp 83–90
42. Gao Y, Chen Y, Wang J, Tang M, Lu H (2019) Reading scene text with fully convolutional sequence modeling. *Neurocomputing* 339:161–170
43. Hu W, Cai X, Hou J, Yi S, Lin Z (2020) Gtc: guided training of ctc towards efficient and accurate scene text recognition. In: *Proceedings of the AAAI conference on artificial intelligence*, vol 34, pp 11005–11012
44. Bahdanau D, Cho KH, Bengio Y (2015) Neural machine translation by jointly learning to align and translate. In: *3rd International conference on learning representations, ICLR 2015*
45. Lee C-Y, Osindero S (2016) Recursive recurrent nets with attention modeling for ocr in the wild. In: *Conference on computer vision and pattern recognition (CVPR)*, pp 2231–2239
46. Xiong R, Yang Y, He D, Zheng K, Zheng S, Xing C, Zhang H, Lan Y, Wang L, Liu T (2020) On layer normalization in the transformer architecture. In: *International conference on machine learning (ICML)*. PMLR, pp 10524–10533
47. Woo S, Park J, Lee J-Y, Kweon IS (2018) Cbam: convolutional block attention module. In: *European conference on computer vision (ECCV)*, pp 3–19
48. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. *J Mach Learn Res* 15(1):1929–1958
49. Ioffe S, Szegedy C (2015) Batch normalization: accelerating deep network training by reducing internal covariate shift. In: *International conference on machine learning*. PMLR, pp 448–456
50. Xu J, Sun X, Zhang Z, Zhao G, Lin J (2019) Understanding and improving layer normalization. In: *Advances in neural information processing systems (neurIPS)*, vol 32
51. Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, Gimelshein N, Antiga L et al (2019) Pytorch: an imperative style, high-performance deep learning library. In: *Advances in neural information processing systems (neurIPS)*, pp 8026–8037
52. Cheng Z, Bai F, Xu Y, Zheng G, Pu S, Zhou S (2017) Focusing attention: towards accurate text recognition in natural images. In: *International conference on computer vision (ICCV)*, pp 5076–5084
53. Yang X, He D, Zhou Z, Kifer D, Giles CL (2017) Learning to read irregular text with attention mechanisms. In: *International joint conference on artificial intelligence (IJCAI)*, vol 1, p 3
54. Liu W, Chen C, Wong K-Y (2018) Char-net: a character-aware neural network for distorted scene text recognition. In: *AAAI conference on artificial intelligence (AAAI)*, vol 32
55. Luo C, Jin L, Sun Z (2019) Moran: a multi-object rectified attention network for scene text recognition. *Pattern Recogn* 90:109–118
56. Mishra A, Alahari K, Jawahar C (2012) Scene text recognition using higher order language priors. In: *British machine vision conference (BMVC)*. BMVA
57. Wang K, Babenko B, Belongie S (2011) End-to-end scene text recognition. In: *International conference on computer vision (ICCV)*. IEEE, pp 1457–1464
58. 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 Recognit (IJDAR)* 7(2-3):105–122
59. Karatzas D, Shafait F, Uchida S, Iwamura M, Bigorda LGI, Mestre SR, Mas J, Mota DF, Almazan JA, De Las Heras LP (2013) Icdar 2013 robust reading competition. In: *International conference on document analysis and recognition (ICDAR)*. IEEE, pp 1484–1493
60. Karatzas D, Gomez-Bigorda L, Nicolaou A, Ghosh S, Bagdanov A, Iwamura M, Matas J, Neumann L, Chandrasekhar VR, Lu S et al (2015) Icdar 2015 competition on robust reading. In: *International conference on document analysis and recognition (ICDAR)*. IEEE, pp 1156–1160
61. Phan TQ, Shivakumara P, Tian S, Tan CL (2013) Recognizing text with perspective distortion in natural scenes. In: *International conference on computer vision (ICCV)*, pp 569–576
62. Risnumawan A, Shivakumara P, Chan CS, Tan CL (2014) A robust arbitrary text detection system for natural scene images. *Expert Syst Appl* 41(18):8027–8048

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.



Shifeng Xia received the M.S. degree in Software Engineering from Nanchang Hangkong University, Nanchang, China, in 2020. He is currently pursuing the PH.D. degree in Nanjing University of Aeronautics and Astronautics. His research interests include deep learning and computer vision.



Jinqiao Kou received the M.S. degree in Detection Technology and Automation Device from Beihang University, Beijing, China, in 2002. He is currently working at the Beijing Institute of Computer Technology and Applications. His research interests are system architecture and emerging technologies.



Tianxiang Yin received the B.S. degree in Computer Science and Technology from Tiangong University, Tianjin, China in 2013. He is currently pursuing the PH.D. degree in Nanjing University of Aeronautics and Astronautics. His research interests include deep learning and computer vision.



Ningzhong Liu received the B.S. degree in Computer Engineering from Nanjing University of Science and Technology, Nanjing, China in 1998, and received the Ph.D. degree in Pattern Recognition and Intelligent Systems from Nanjing University of Science and Technology in 2003. He is currently a professor of the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China. His research interests include artificial intelligence and machine learning.