International Journal of Computational Intelligence Systems
Vol. **13(1)**, 2020, *pp.* 341–351
DOI: https://doi.org/10.2991/ijcis.d.200316.001; ISSN: 1875-6891; eISSN: 1875-6883
https://www.atlantis-press.com/journals/ijcis/

Regular paper

End-to-End Sequence Labeling via Convolutional Recurrent Neural Network with a Connectionist Temporal Classification Layer

Xiaohui Huang[1,2,*], Lisheng Qiao[1], Wentao Yu[2], Jing Li[1], Yanzhou Ma[2]

[1] *College of Computer Science and Technology, University of Science and Technology of China, Hefei, Anhui, China*
[2] *Zhengzhou Information Science and Technology Institute, Zhengzhou, Henan, China*

ARTICLE INFO

Article History
Received 23 Sep 2019
Accepted 12 Mar 2020

Keywords
Sequence labeling
Convolutional recurrent neural network
Unified framework
End-to-end

ABSTRACT

Sequence labeling is a common machine-learning task which not only needs the most likely prediction of label for a local input but also seeks the most suitable annotation for the whole input sequence. So it requires the model that is able to handle both the local spatial features and temporal-dependence features effectively. Furthermore, it is common for the length of the label sequence to be much shorter than the input sequence in some tasks such as speech recognition and handwritten text recognition. In this paper, we propose a kind of novel deep neural network architecture which combines convolution, pooling and recurrent in a unified framework to construct the convolutional recurrent neural network (CRNN) for sequence labeling tasks with variable lengths of input and output. Specifically, we design a novel CRNN to achieve the joint extraction of local spatial features and long-distance temporal-dependence features in sequence, introduce pooling along time to achieve a transform of long input to short output which will also reduce he model's complexity, and adopt Connectionist Temporal Classification (CTC) layer to achieve an end-to-end pattern for sequence labeling. Experiments on phoneme sequence recognition and handwritten character sequence recognition have been conducted and the results show that our method achieves great performance while having a more simplified architecture with more efficient training and labeling procedure.

© 2020 *The Authors*. Published by Atlantis Press SARL.
This is an open access article distributed under the CC BY-NC 4.0 license (http://creativecommons.org/licenses/by-nc/4.0/).

1. INTRODUCTION

Deep neural network has been recently applied to sequence labeling which shows comparatively remarkable performance. Both recurrent neural network (RNN) and convolutional neural network (CNN) have been used for sequence labeling tasks such as speech recognition or handwritten text recognition [1–3]. As RNN is good at extracting temporal-dependence features and CNN is capable of extracting local spatial features, they have both achieved great success on some sequence labeling tasks [4,5]. In addition, Connectionist Temporal Classification (CTC) is a sequence labeling algorithm that provides and optimizes a probability distribution over variable length output sequence conditioned on longer input sequence [6,7]. Since CTC does not require forced alignment to pre-segment the input sequence, it is feasible to train an "end-to-end" system using RNN or CNN equipped with CTC.

Although the application of RNN and CNN on sequence labeling is impressive, they still have their own shortcomings. For example, it can be difficult to train RNN by back-propagation through time due to the iterative multiplication over time when the input feature sequence is very long, and the training is sometimes tricky due to the well-known problem of gradient vanishing/exploding [8]. Compared to RNN-based models, CNN has faster computational speed and is more efficient on training due to its good parallelism. However, the convolutional kernels in CNN are usually of fixed size on two dimensions referred as feature axis and time axis. When using such kernels, it is difficult to determine the kernel's size: small size may result in the loss of some critical information such as the spectral correlation or temporal correlation, whereas large size may result in enormous parameters or over-fitting. In order to extract temporal correlations within long distance in sequences, they have to stack multiple layers to construct very deep model [9] that is difficult to train, which will in turn result in more parameters and more requirements for storage space. At the same time, some RNN and CNN combined models are proposed in order to play better of their advantages [10–14]. However, most of them follow the cross-stacking architecture in which different components work respectively. This architecture is not only ineffective in joint modeling of local spatial features and temporal features but also more complicated on the training procedure.

In order to make best use of the advantages of CNN and RNN for sequence labeling, we propose a novel end-to-end sequence labeling framework based on CNN, RNN and CTC. Particularly, the motivation and the overall implementation of our method can be summarized as follows:

* *Corresponding author. Email: huangxia@mail.ustc.edu.cn*

Firstly, in order to extract the local spatial features and long-distance temporal-dependence features from input sequence effectively for sequence labeling, we introduce the convolutional structure into RNN to construct a unified convolutional recurrent neural network (CRNN) in which the recurrent connections are used to capture the long-distance temporal-dependence features and the convolutional connections are used to capture the local spatial features.

Secondly, in order to achieve a better transform of long input to short output, we introduce two-dimensional (2-D) pooling along feature and time into sequence labeling model to achieve further refinement of features before the final prediction of the label sequence. Specifically, we use pooling along time to transform multiple input vectors into just one output vector which could eliminate some redundant features and simplify the model.

Thirdly, in order to achieve end-to-end training, we use CTC loss to train our model. The CTC could provide and optimize a probability distribution over shorter output sequence conditioned on longer input sequence. Since it does not require forced alignments to pre-segment the input sequence. We can use it to achieve end-to-end training.

Finally, experiments on two tasks (phoneme recognition and handwritten character recognition on) will be conducted to validate the feasibility and efficiency of our method on sequence labeling tasks with shorter label sequence than input sequence.

2. RELATED WORK

It has been studied for a long time to apply deep neural networks to sequence labeling tasks. RNN and CNN have both ever been used for speech recognition or handwritten character recognition, and they gained dramatic improvement. The use of CTC makes it possible to train deep RNN or CNN in an end-to-end manner for sequence labeling, and these "end-to-end" models have achieved promising results on several tasks. For example, paper [9] constructed a model with multiple Bi-LSTM (Bidirectional Long Short-Term Memory, one type of RNN cell) layers and a CTC output layer which obtained the state-of-the-art results on TIMIT phoneme recognition when it was proposed. Paper [7] proposed a model based on CNN equipped with CTC for "end-to-end" training. It achieved 18.2% phoneme error rate (PER) on TIMIT core test set which was comparable to the state-of-the-art models based on RNN-CTC while getting a more efficient training procedure. Apart from the CTC based models, there is another type of "end-to-end" architecture which uses attention mechanism to perform alignment between input vector and the predicted label [15–18]. Attention-based neural network models implicitly realize encoding, attention and decoding for sequence labeling. Compared with CTC, attention-based models could make predictions conditioned on all the extracted features or previous predictions, and thus can learn the context information of the feature sequence or the label sequence. Alignment between the input feature and the desired label could also be learned automatically by attention mechanism.

However, as sequence labeling tasks not only needing the most likely label for a single input but also seeking the most suitable annotation for the whole input, it requires the model to handle both the local spatial features and temporal-dependence features in sequence for overall labeling. The works mentioned above are purely RNN- or CNN-based models which are inability to handle the both features. Therefore, they have to stack multiple layers to enhance their ability of extracting features. The result is that enormous parameters will be generated and need to be trained. Therefore, they have many shortcomings such as the poor efficiency of training, the ease of over-fitting or the overhead of computing. Although various approaches such as data/model parallelization across multiple GPUs and careful initialization for connections have been proposed to address the above issues, those models still suffer from intensive computation or complicated training procedure. Furthermore, the attention mechanism is too flexible in the sense that it allows extremely non-sequential alignments wherever the alignments are usually monotonic in sequence labeling tasks such as speech recognition or handwritten text recognition. So the cost of using an explicit alignment without monotonic constraints means that the alignment may become impaired.

In order to make best use of the advantages of RNN and CNN, some CNN and RNN combined models are proposed [19]. The most common architecture is designed as encoder–decoder in which CNN is used as an encoder to transform the input feature into an intermediate vector and RNN is used as a decoder to transform the intermediate vector to target labels. This architecture has been widely used in speech [20,21], handwritten text [22], image [23,24] and other related fields. Furthermore, these combined models have also been applied to the classification of sentiment texts [25], medical images [26,27] and other classification problems. Particularly, there has been another form of combined model in which the pooling layer widely used in CNN is equipped on the output of a recurrent layer. Literature [28] proposes a recurrent CNN for text classification and achieves a very good result. The authors first use a recurrent neural layer to capture contextual information as far as possible and then employ a max-pooling layer on the output of the recurrent layer aiming to automatically judge which word plays key roles for classification. In summary, the motivations of these combined models are all same, that is to combine the advantages of them to achieve joint feature extraction. But in fact, most of the models are just simple pipelined stacking of CNN and RNN in which the two subparts are actually separate components. Due to the pipelined stacking structure, error propagation may seriously restrict the joint extraction ability of the model. For example, when CNN completes the extraction of local spatial features, it may have missed some temporal features. As a result, the subsequent RNN will not be able to extract any valuable temporal association features. So the main problem of this architecture is the uncertainty to the joint extraction of features caused by the pipelined structure.

Furthermore, it is common for the length of the label sequence to be much shorter than the input sequence in some tasks such as speech recognition and handwritten text recognition. Traditional neural network-based models such as RNN, CNN or their combination usually predict a label for each input individual feature vector, and the output of each hidden layer follows the same pattern. This architecture undoubtedly contains large amount of redundancy. They not only have to perform intensive computational search to find the optimal label sequence from the grid, but also need to do a lot of merging to get the final label sequence.

In light of the success of these methods mentioned above, especially the success of CNNs' local connection, shared weights and

the mechanism of pooling, in this paper we propose a new neural network architecture which introduces local connection and shared weights into recurrent neuron and introduces 2-D pooling along feature and time axis to sequence labeling. The rest of this paper is organized as follows: the key points of our architecture are presented in Section 3. Experimental details and results are described in Section 4 and the final conclusions with a brief outlook on the future work are made in Section 5.

3. ARCHITECTURE OF THE PROPOSED MODEL

The proposed model which we refer as a CRNN starts with an input layer. It is regarded as a feature vector sequence extracted from origin input. The input feature sequence could be depicted as Equation (1).

$$X = \{x_1, x_1, \cdots, x_T\} \tag{1}$$

where x is a feature vector of size F, and X is a sequence of the feature vector, T is the length of X. The input feature sequence is treated as a 2-D matrix with height along the feature axis and width along the time axis. It will be feed into the network and processed in succession. The overall architecture of our end-to-end sequence labeling model can be depicted as Figure 1:

3.1. Convolutional Recurrent Neural Layer

The basic component of our method is convolutional recurrent neural layer which could be seen as an improved version of RNN. Unlike traditional RNN in which the input layer and the hidden layer are fully connected, here we adopt local connection and shared weights just like the inspiration of convolutional kernels. The neuron in the convolutional recurrent neural layer proposed here could achieve operation of both convolution and recurrent. We call this neuron as convolutional recurrent kernel.

Given an input sequence $x = \{x_1, \ldots, x_T\}$ with a shape of $[F, T]$ along feature axis and time axis, the convolutional recurrent kernel will first process the inputsequenceas equation group (2):

$$z_{f,t} = W_k \otimes x_{f,t} + b_k \tag{2}$$

$$y_{f,t} = H\left(z_{f,t}\right)$$

In Equation (2), t denotes the t-th time step, f denotes the f-th patch along feature axis, W_k is a 2-D matrix responding to a convolutional recurrent kernel and b_k is the kernel's bias parameter. W_k will make an element-wise multiplication with patch $x_{t,j}$ in the input sequence which has a same shape as W_f, then all values will be summed to just one value, and then the value will be added with bias parameter and feed into an activation function to form a final output $y_{f,t}$. So $y_{f,t}$ denotes the convolution output of a kernel (convolutional recurrent kernel) in patch f and time t. The data processing of a single neuron in convolutional recurrent layer can be showed in Figure 2:

In Figure 2, a single feature filter uses a perceptual region of certain size (called as patch, 2*2 region in the figure) to first perform one-dimensional convolution along the feature dimension, thereby generating a local feature map at a certain moment. Then it will slide along the time dimension to perform the same convolution operation on the sequence at the next moment. After that, recurrent connection of the filter will introduce the local map generated at the previous moment into new local feature map, and finally realize joint extraction of local spatial features and time-dependent features.

Obviously, there are multiple neurons in a convolutional recurrent layer, and each neuron will generate an instantaneous local feature map at a certain moment, so multiple neurons will generate multiple feature maps that need to be processed in the following process of time iteration. Therefore, we improved the process of subsequent iterations along time. Taking data processing of two convolutional recurrent neurons as an example, it can be depicted as Figure 3:

In Figure 3, two neurons (black and red kernel) will generate two local instantaneous feature maps at a certain moment. Before iteration along time, we use pooling mechanism which is commonly used in CNN to handle the processing along time of multiple feature graphs. Specifically, we only compute the average value by average-pooling on each feature map, and then all averaged values are concatenated to form a hidden state feature vector at the

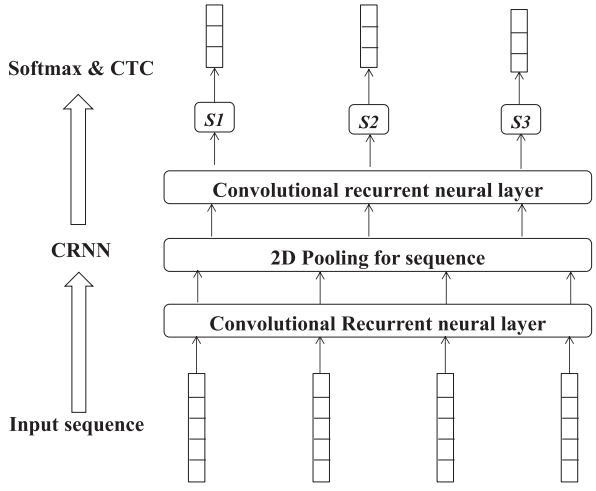

Figure 1 The architecture we proposed for end-to-end sequence labeling.

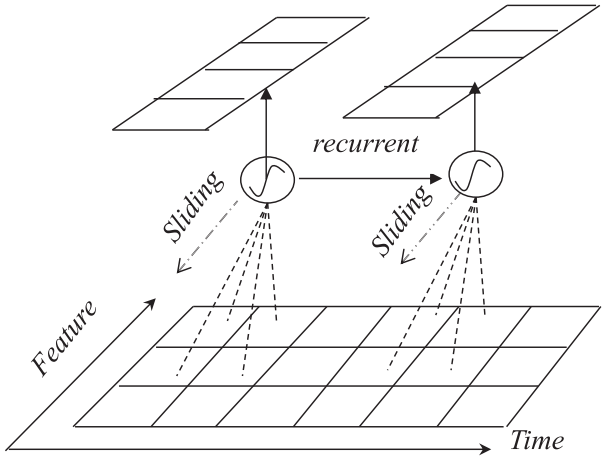

Figure 2 The convolutional process of a convolutional recurrent neuron.

current moment. Finally, this feature vector is weighted and then enters the convolutional recurrent neuron at the next moment. In this mode, the final dimension of the hidden state vector generated at the previous moment is still the number of neurons, so the number of recurrent connections is still N*N (N is the number of convolutional recurrent neurons). Then the network can directly process a sequence just like the original RNN. The calculation of the convolutional recurrent layer can be expressed as formula group (3):

$$\begin{aligned} map_{t,n} &= conv1\left(x_t, C_n\right) \\ y_{t+1,n} &= Rnncell\left(map_{t+1,n}, \left[pool\left(map_{t,1}\right), \cdots, pool\left(map_{t,N}\right)\right]\right) \\ y_{t+1} &= \left[y_{t+1,1}, \cdots, y_{t+1,N}\right] \end{aligned} \tag{3}$$

In addition, Gated Recurrent Unit (GRU) cell which has two gates to control the input and output has been proved to be more efficient for handling long-distance temporal correlations on many applications [29], so we also introduce GRU cell into the convolutional recurrent neuron called as convolutional GRU (CGRU). Since convolution has been introduced in front, its output is seen as input of the GRU cell. The computation in a convolutional recurrent layer based on CGRU cell can be illustrated as Figure 4.

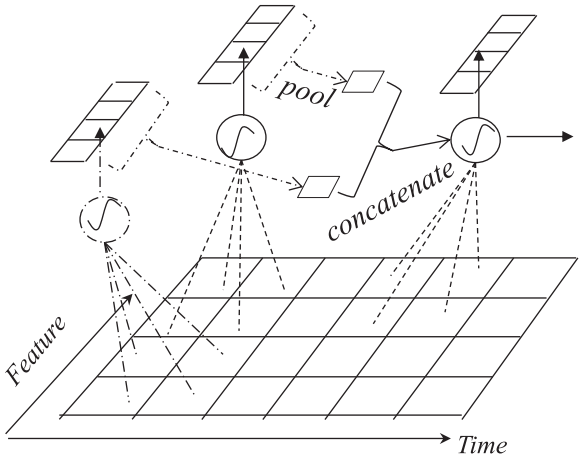

Figure 3 The recurrent processing of two convolutional recurrent neurons.

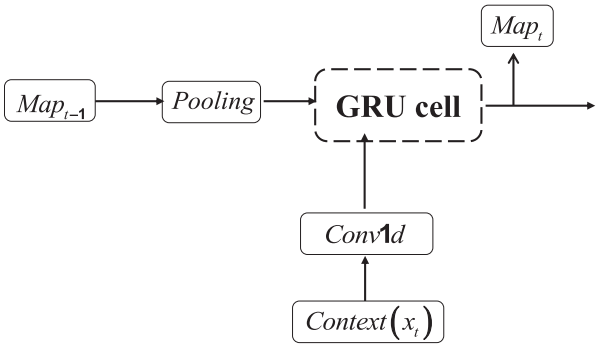

Figure 4 The data processing of the convolutional gated recurrent unit (CGRU) cell.

3.2. Bidirectional Convolutional Recurrent Network

Conventional recurrent layer is unidirectional and only able to make use of the previous context, but processing sequence from both directions is more suitable for sequence labeling tasks [30]. Here, bidirectional convolutional recurrent layer who using two separate hidden layers to iterative from forward and backward respectively is also adopted. The outputs of the two directions are processed as Equation (4) to form a new feature sequence:

$$h_t = w_{\vec{h}}\vec{h}_t + w_{\overleftarrow{h}}\overleftarrow{h}_t \tag{4}$$

Finally, it has been shown that stacking multiple neural network layers on the top of each other could achieve better performance on many sequence labeling tasks. This can be simply done by treating the output of a convolutional recurrent neural layer as the input of the higher layer. In our work, we also constructed deep network composed of several convolutional recurrent neural layer for sequence labeling, and the experimental results confirmed their superior performance.

3.3. 2-D Pooling on Sequence

As in conventional RNN, the length of the output sequence is always equal to the input sequence. While for some tasks, the length of the input is often much longer than the target output (such as speech recognition, handwritten recognition and so on), that is to say there are several input vectors responding to just one output label, so we introduce 2-D pooling into sequence labeling tasks which is commonly used in image process but rarely used for sequence labeling. The purpose of introducing 2-D pooling is to achieve reduction of the feature redundancy while obtaining a more simplified network. The 2-D pooling on sequence can be illustrated as Figure 5, and the effect of pooling along time can be illustrated as Figure 6.

Usually, max-pooling and average-pooling are the two most commonly used pooling methods. We tested both and found that max-pooling would have better performance. As can be seen from Figure 6, after pooling along time, the length of upper layers' output sequence could be shorter than its lower layer according to the value of the pooling stride, and the complexity of the whole network would also be reduced.

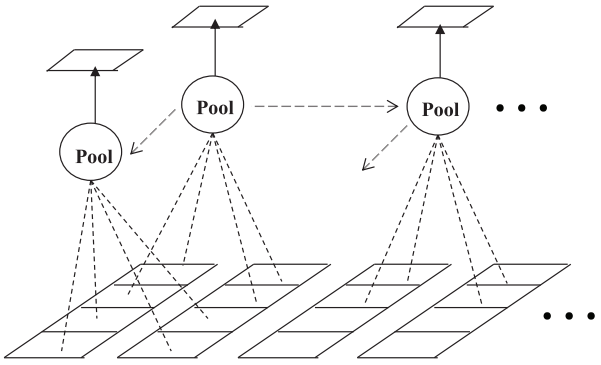

Figure 5 Illustration of the two-dimensional (2-D) pooling on sequence.

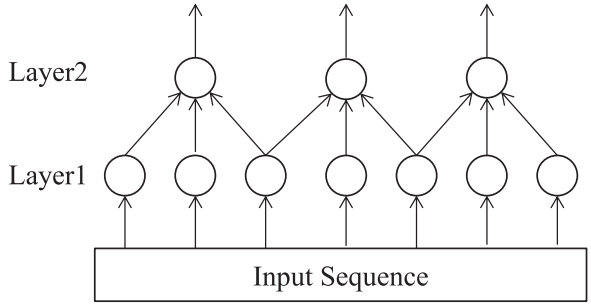

Figure 6 Illustration of the pooling effect along time in sequence.

3.4. CTC Output Layer

CTC output layer is equipped on the output of the last convolutional recurrent layer. It introduces a many-to-one mapping of latent variable sequences output by the network to a shorter sequence which serves as the final prediction. It will first add an extra blank output class "-" to the target label alphabet K to represent the probability of not outputting a symbol at a particular time step. Then a soft-max function is applied to the output of the network for each time step. Assuming the output sequence by the last recurrent layer is denoted as π, and it is composed of target label and blank label with the length of T which is the folding number of the last recurrent layer along time axis, the probability to generate this sequence is computed as Equation (5):

$$\Pr\left(k,t|x\right)=\frac{\exp\left(y_t^k\right)}{\sum_{k'}\exp\left(y_t^{k'}\right)},k\in K\cup blank \tag{5}$$
$$\Pr\left(\pi|x\right)=\prod_{t=1}^{T}\Pr\left(k,t|x\right),k\in K\cup blank$$

Any output sequence π generated from the last recurrent layer can now be transformed into a target sequence y using the many-to-one mapping function $\beta(\pi)$ which first merges the repetitions of consecutive non-blank labels to a single label and subsequently removes the blank labels. That is also to say there will be multiple sequences output by the last recurrent layer that could be mapped to the same target sequence y. The processing of mapping function $\beta(\pi)$ can be explained by examples as Equation (6):

$$y=\beta\left(\pi\right)\Leftrightarrow\left(a,b,c\right)=\begin{cases}\beta\left(a,b,-,-c\right)\\ \beta\left(-,a,b,-,c\right)\\ \beta\left(-,a,a,b,-,c,c\right)\\ \vdots\\ \beta\left(-,a,-,b,-,c,c\right)\end{cases} \tag{6}$$

Therefore, the probability of a target sequence y is the summation of probability over all possible sequences π that yield to y after applying the function β^{-1}:

$$\Pr\left(y|x\right)=\sum_{\pi\in\beta^{-1}(y)}\Pr\left(\pi|x\right) \tag{7}$$

Usually, negative log probability is used as the objective function to be minimized which is defined as Equation (8):

$$CTCLoss=-\log\Pr\left(y|x\right) \tag{8}$$

A dynamic programming algorithm similar to the forward algorithm for HMMs is used to compute the summarization in Equation (7) in an efficient way and the intermediate values of this dynamic programming can also be used to compute the gradient of Equation (8) with respect to the neural network outputs efficiently. When the model has been trained appropriately, it could be used to predict the target label sequence from the input feature sequence by beam search decoding algorithm [1].

Generally speaking, the main contributions and advantages of our method can be summarized as follows:

- We design a novel CRNN suitable for joint extraction of local spatial features and long-distance temporal-dependence features in sequence. The network is an integration of RNN and CNN in which the CNN's structure of local connection and shared weights are introduced into RNN. By combining the recurrent structure with convolution in a unified framework, the CRNN could successfully utilize the advantages of both RNN and CNN for feature extraction from sequence.
- We introduce pooling along time into sequence labeling tasks with longer input than output. This type of pooling could cause a reduction in length of the output over input sequence proportionally according to the stride along time. Actually, the operation can not only result in a great reduction of the complexity of the network but also achieve further refinement of features and then improve the efficiency of training and labeling.
- We construct an end-to-end model which integrates convolutional recurrent layer, pooling layer and CTC output layer into a unified architecture for sequence labeling. Experimental results on TIMIT, IAM and RIMES show that the architecture not only achieves a comparable result to the state-of-art methods, but also has a more simplified architecture with more effective training and labeling procedure.

4. EXPERIMENTS

Two benchmark sequence labeling tasks were performed to validate the performance of the proposed neural network architecture. The first one was phoneme sequence recognition on TIMIT speech corpus [31], and the other was handwritten character sequence recognition on IAM [32] and TIMES [33]. Details of the experimental tasks, datasets, models, baselines, training procedure and the final results are presented in the following parts:

4.1. Phoneme Sequence Recognition

TIMTI speech corpus: A corpus of phonemically and lexically transcribed speech of American English speakers. TIMIT corpus includes time-aligned phonetic and word transcriptions as well as a 16-bit, 16K Hz speech waveform file for each utterance. It contains a total of 6300 speech sentences with 10 sentences spoken by each of 630 speakers from 8 major dialect regions of the United States. We used the standard 462-speaker training set with all SA records removed to train the proposed model and use the 50-speaker development set as validation set for early stopping during training. Finally, the evaluation was performed on the core test

set (including 192 sentences spoken by 24 speakers). Each speech utterance is accompanied with a target phoneme sequence that composed of 61 kinds of phoneme labels [34].

For phoneme recognition, the input was a sequence of features that extracted from a raw audio. The raw audio was first framed with length of 25 ms and step of 10 ms, and then we extracted 40-dimensional log-mel-filter-bank coefficients from each frame to form a 40-dimensional feature vector which was commonly used for speech recognition. After that, each dimension of the feature vector was normalized to have zero mean and unit variance over the whole training, development and core test sets.

We validated the hyper-parameters through grid search. The starting point refers to the three-layer network structure proposed in paper [9]. We set up different groups to search the hyper-parameters jointly such as the number of layers, quantity of the filters in a layer, shape of the filters, stride of the pooling, etc. Search will be done along quantity both from less to more and from more to less. Each group of hyper-parameters will be trained and tested at least 10 times on TIMIT. During the training time, stochastic gradient descent with learning rate 10^{-4} was used to optimize the proposed model, and we adopted a mini-batch of 36 sequences each of which was padded zero to the longest sequence's length during training. The weights' initial values of the whole network were drawn uniformly from [−0.1, 0.1]. In addition, zero padding was applied on the input both along feature and time axis in order to reserve the original frequency and temporal information when convolution and pooling were being done.

PER [1] was adopted as the evaluation metric for the phoneme recognition experiments. Computing of the PER is consist of three steps: the first step is decoding (best path decoding [1]) the CTC output to the final phoneme sequence, and then Levenshtein edit distance between the decoded phoneme sequence and the ground truth sequence will be computed as the number of mistakes made by the model. At last the average number of mistakes over the length of the phoneme sequence is just the final PER. When searching the optimal hyper-parameters, PER would be evaluated on the validation set after each pass through the training set, and training would be stopped after 50 evaluations with no improvement. After each group of hyper-parameters was evaluated, the mean and standard deviations of all PERs on test dataset will be recorded. The hyper-parameters and weights giving the lowest PER on test set were selected as the final result. At last, the optimal network is composed of two stacked bidirectional convolutional recurrent layers with a pooling layer embedded. The output layer is a soft-max layer with a size of 62 (responding to 61 phoneme labels and a blank label). Details of the network are listed in Table 1:

Table 1 Details of our model for phoneme recognition.

Layer	Type	Configuration
Input	vector	Sequence of 40-dimension vector
Layer1	Bi-CRNN	Kernels (512), filter [20, 10], average-pooling
Layer2	2-D Pooling	Shape [4, 4], max-pooling
Layer3	Bi-CRNN	Kernels (512), filter [10, 5], average-pooling
Output	CTC	62 nodes

CRNN: convolutional recurrent neural network; 2-D: two-dimensional; CTC: Connectionist Temporal Classification.

In order to validate the superiority of the proposed network over other types of neural network on sequence labeling tasks, we constructed some typical neural network models equipped with CTC for comparison. They are common models purely based on RNN, CNN or their combination without any additional auxiliary components or processing (such as dictionaries, language model or other post processing). We explored their structures according to their PERs and the one achieve best PER were selected as baseline for comparison. At the same time, the epochs and PERs of each baseline model were recorded and their means and standard deviations were computed when the exploration were completed. At last, the statistical results are shown in Table 2.

Furthermore, we also investigated the training procedure on phoneme sequence labeling. The changing trends of PER corresponding to training epochs are observed and recorded. Finally, the comparison with other models is shown in Figure 7.

Besides, we also selected several works achieved state of art performance on TIMIT phoneme sequence recognition for references. They were typical models based on neural network and achieved state-of-art results when proposed. The results are shown in Table 3 and the main contributions of those works are listed in detail as below:

- BLSTM+5L+CTC [9]: the model combines a 5-layer LSTM with CTC to achieve end-to-end training.
- TRANS-3L-250H [9]: the model is composed of a 3-layer LSTM trained as the scheme of RNN transducer with random initial weights.

Table 2 Comparison with other typical neural network.

Model	Training Epochs: Mean [stdev]	Test PER: Mean [stdev]
LSTM + CTC	117 [4.9]	25.6% [0.24%]
Bi-LSTM + CTC	132 [5.5]	23.4% [0.19%]
CNN + CTC	146 [5.5]	23.3% [0.21%]
CNN + LSTM + CTC	121 [6.9]	22.1% [0.24%]
Ours (CRNN + CTC)	113 [3.0]	17.1% [0.09%]

CRNN: convolutional recurrent neural network; CNN: convolutional neural network; CTC: Connectionist Temporal Classification; PER: phoneme error rate.

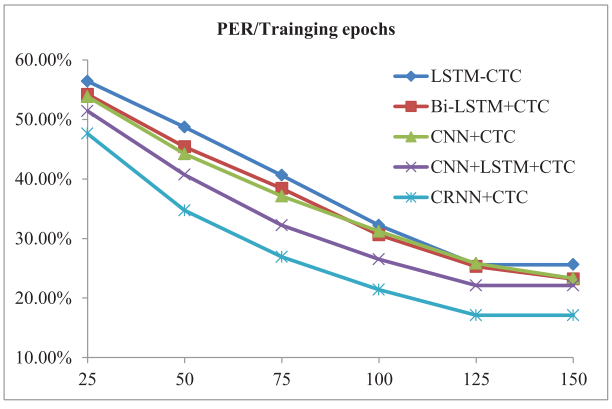

Figure 7 Comparison with other neural networks on training procedure.

- PreTrans-3L-250 [9]: the model is composed of a 3-layer LSTM trained as the scheme of RNN transducer with pre-trained next-step prediction network.
- CNN+CTC [7]: a CNN-based model which is composed of 10 convolutional layers, 3 fully-connected layers and a CTC layer.
- RCNN+MLP [20]: a recurrent CNN which can be viewed as a convolutional layer embedded with an RNN for speech processing.
- Attention model [16]: an attention-based RNN which could add location awareness to attention mechanism and avoid concentration on a single frame.
- Conv-in-Conv [35]: a CNN-based model which combines the frequency-domain convolution with time-domain convolution structure.

4.2. Handwritten Character Sequence Recognition

IAM handwritten database 3.0: The database contains of unconstrained English handwritten text in forms of page, sentence, line and word. It totally contains of 1539 pages of scanned text written by 657 different writers. All the pages and extracted sentences, text lines, words are available as PNG files with 256 gray levels. In our experiments, we use the text lines dataset with the commonly used partition [36] which includes writer-independent 6161 text lines as the training set for the networks' training, 940 text lines as the validation set for early stopping and 1861 text lines as testing set for evaluating the proposed models. The line images have an average height of 124 pixels and average width of 1751 pixels. The three sets are writer independent, that is to say a person who has contributed to any of the three sets did not contribute to any of other sets. Each of the handwritten text lines is accompanied with a target character sequence which is composed of 79 kinds of distinct target labels (capital letters, lower case letters, numbers, punctuations and whitespace).

RIMES Database: the dataset is a modern French handwritten text database used for ICDAR 2011 competition. It has 12, 723 scanned pages of mails handwritten by 1300 writers. It also contains 11333 training lines and 778 test lines. Since the original competition release does not include a separated validation partition, we sampled 10% of the training lines to create our validation set. Thus, the final division of the dataset into training, validation and test consists of 10203, 1130 and 778 lines, respectively. The original line images in the training set have an average width of 1658 pixels and an average height of 113 pixels. There are 99 different labels in this dataset.

For off-line handwritten character recognition, the convolutional recurrent layer will just slide over the processed text-line image from left to right (or right to left). In this way, we needn't any other feature engineering for the text-line image. The hyper-parameters of the network such as the number of layers, the quantity of the filters in a layer, the shape of filters, the stride of pooling are all searched as the same procedure of Section 4.1. The output layer is also soft-max layer with a size of 80 for IAM and a size of 100 for RIMES (responding to all distinct labels and a blank label in the two dataset, respectively). During the training time, the initial weights of the whole network were drawn uniformly from [−0.1, 0.1], and the complete system was trained with stochastic gradient descent using a learning rate of 10^{-4} and a momentum of 0.9. We adopted a mini-batch of 54 sequences each of which was padded to a unified shape during training. The character error rate (CER, computed just same as the PER) was evaluated on the validation set after each pass through the training set, and the training was stopped after 50 evaluations with no improvement. Hyper-parameters and weights giving the lowest error rate on the test set were selected as the final result.

According to experimental results, the final network is also composed of two stacked bidirectional convolutional recurrent layers with a pooling layer embedded. Details of the optimal network are listed in Table 4:

In order to validate the superiority of the proposed network for character sequence recognition over other neural network, four typical models purely based on RNN, CNN or their combination were constructed for comparisons. We recorded the values of training epoch and test CER when the model converges. The mean and standard deviation of each model's final training epoch and CER were computed as indictors. At last, the statistical results are shown in Tables 5 and 6.

Table 3 PER comparison with other state-of-art results.

Model	NP	AC	PER (%)
BLSTM + 5L + CTC [9]	5	-	18.4
TRANS-3L-250H [9]	3	Transducer	18.3
PreTrans-3L-250 [9]	3	Pre-Transducer	17.7
CNN + CTC [7]	13	-	18.2
RCNN + MLP [20]	6	FA	18.0
Attention model [16]	4	CF	17.6
Conv-in-Conv [35]	6	FA, LM	16.7
Ours (CRNN + CTC)	2	-	17.1

NP: number of layers with parameters; AC: auxiliary components; FA: forced alignment; CF: convolutional features; LM: language model; CRNN: convolutional recurrent neural network; CNN: convolutional neural network; CTC: Connectionist Temporal Classification; PER: phoneme error rate.

Table 4 Details of our model for character recognition.

Layer	Type	Configuration
Input	Vector	Sequence of image pix column
Layer1	Bi-CRNN	Kernels (512), filter [20, 10], average-pooling
Layer2	2-D pooling	Shape [6, 6], max-pooling
Layer3	Bi-CRNN	Kernels (512), filter [10, 5], average-pooling
Output	CTC	80 for IAM and 100 for RIMES

CRNN: convolutional recurrent neural network; 2-D: two-dimensional; CTC: Connectionist Temporal Classification.

Table 5 Comparison with other typical neural network components on IAM.

Model	Training Epochs: Mean [stdev]	Test CER: Mean [stdev]
LSTM + CTC	96 [4.2]	9.6% [0.08%]
Bi-LSTM + CTC	104 [4.3]	8.5% [0.12%]
CNN + CTC	113 [4.1]	8.4% [0.11%]
CNN + LSTM + CTC	118 [4.3]	7.9% [0.12%]
Ours (CRNN + CTC)	91 [2.4]	6.1% [0.08%]

CRNN: convolutional recurrent neural network; CNN: convolutional neural network; CTC: Connectionist Temporal Classification.

Furthermore, we also investigated the training procedure on handwritten character sequence labeling. Trends of CER corresponding to the training epochs were also observed and recorded. The result is shown in Figures 8 and 9.

Besides, we also select several results achieve state-of-art performance on IAM and RIMES text-line character sequence recognition for references. They are representative models based on neural network with some additional auxiliary components (such as attention, language model and so on). All results are shown in Tables 7 and 8, and the main contributions of these baselines are listed in detail as below:

- CNN-BLSTM+LSTM [3]: a CNN combined with a Seq2Seq model in which CNN is used to extract relevant features from word images and Seq2Seq model is used to predict the character sequence with the extracted features as input.
- CNN-BLSTM+LSTM [36]: a deep convolutional network with a RNN-based encoder–decoder network in which CNN is used as a feature extractor.
- CNN-BGRU+GRU [18]: an attention-based sequence-to-sequence model which is composed an encoder consisting of a CNN and a bidirectional GRU, an attention mechanism and a decoder formed by a one-directional GRU.
- CNN-1DLSTM-CTC [37]: a neural network architecture based on convolution and 1D-LSTM to perform line-level handwritten text recognition.
- STN-CNN-RNN [38]: A CNN–RNN hybrid model which is composed of spatial transformer network, residual convolutional blocks, bidirectional LSTM and is trained with synthetic data and domain specific image normalization and augmentation.
- CNN-BLSTM-ATT [39]: the model first combines a CNN with a RNN as a feature extractor to encode both the visual information as well as the temporal context between characters in the input image, and then uses a separate RNN to decode the character sequence.

Table 6 Comparison with other typical neural network components on RIMES.

Model	Training Epochs: Mean [stdev]	Test CER: Mean [stdev]
LSTM + CTC	88 [4.1]	5.9% [0.10%]
Bi-LSTM + CTC	94 [5.2]	4.7% [0.11%]
CNN + CTC	86 [5.9]	5.8% [0.14%]
CNN + LSTM + CTC	91 [5.1]	4.4% [0.18%]
Ours (CRNN + CTC)	72 [3.3]	3.4% [0.10%]

CRNN: convolutional recurrent neural network; CNN: convolutional neural network; CTC: Connectionist Temporal Classification.

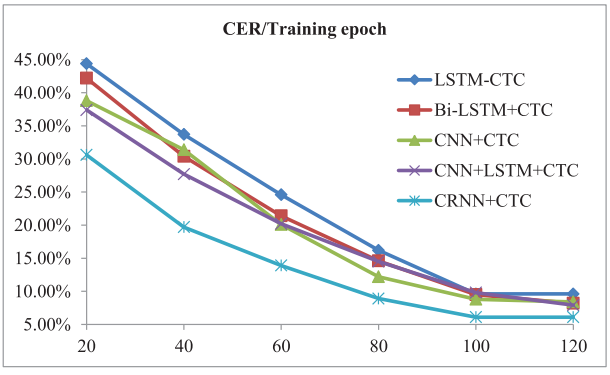

Figure 8 Comparison of training procedure with other neural networks on IAM.

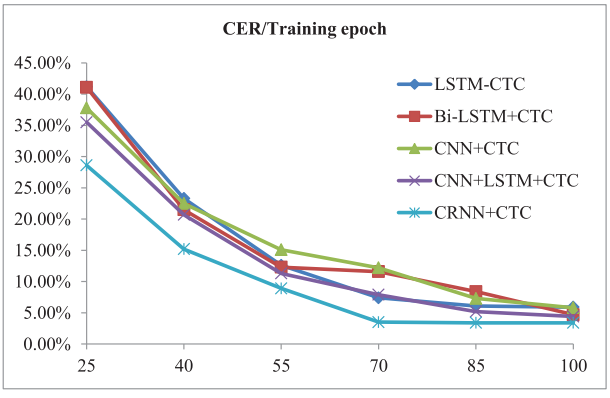

Figure 9 Comparison of training procedure with other neural networks on RIMES.

4.3. Effect of Pooling Along Time

In our method, pooling along time is employed to reduce the network's complexity, but what are the benefits of doing so? In order to explore the impact of this process, we test several pooling strides to verify the effect of pooling along time. The results are shown in Table 8.

In order to understand the changes more clearly, we show the trends through three charts as shown in Figures 10–12.

4.4. Analysis and Discussion

From Tables 2, 5 and 6 we can see that our method achieves the best average PER and CER on two tasks compared to other neural network- and CTC-based architecture. Certainly, we also see that it gets the least training epochs when achieving the best result. It is worth emphasizing that we have used the statistical results of many

Table 7 Comparison with other state of art results on IAM and RIMES.

Model	NP	AC	CER on IAM (%)	CER on RIMES (%)
CNN + LSTM [3]	7	LeNet	8.8	4.8
CNN + LSTM [36]	9	FA	8.1	3.5
CNN + GRU [18]	20	VGG	6.9	-
CNN + 1DLSTM [37]	4	LM	6.2	3.3
STN-CNN-RNN [38]	>5	DA	5.7	5.07
CRNN + ATT [39]	8	DA	4.8	-
Ours (CRNN + CTC)	2	-	6.1	3.4

NP: number of layers with parameters; LeNet: reference as paper [40]; VGG: reference as paper [41]; AC: auxiliary components; FA: forced alignment; LM: language model; DA: data augmentation; CRNN: convolutional recurrent neural network; CNN: convolutional neural network; CTC: Connectionist Temporal Classification; RNN: recurrent neural network; GRU: Gated Recurrent Unit.

Table 8 | Effect of pooling stride along time for PER and CER.

Pooling Stride Along Time	Test PER on TIMIT (%)	Test CER on IAM (%)	Test CER on RIMES (%)
1	20.3	10.60	6.2
2	18.5	8.6	5.1
3	18.1	8.4	4.5
4	17.1	7.8	4.1
5	18.0	7.9	3.6
6	18.4	6.1	3.4
7	19.1	9.0	5.1
8	22.8	11.8	7.6

PER: phoneme error rate.

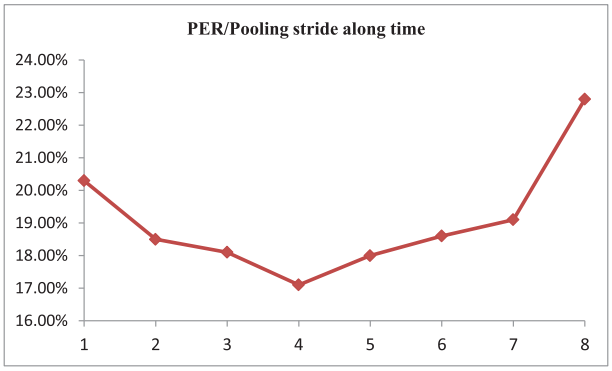

Figure 10 | Effect of pooling stride along time for phoneme error rate (PER) on TIMIT.

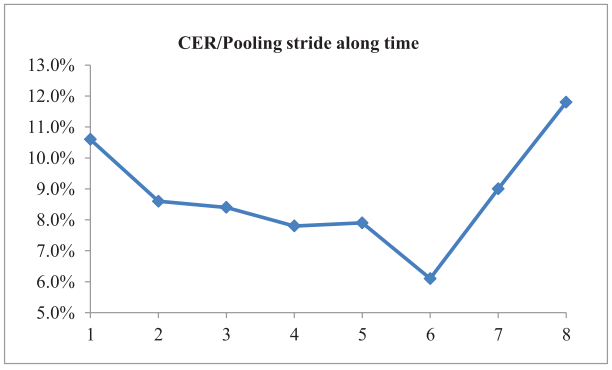

Figure 11 | Effect of the pooling stride along time for CER on IAM.

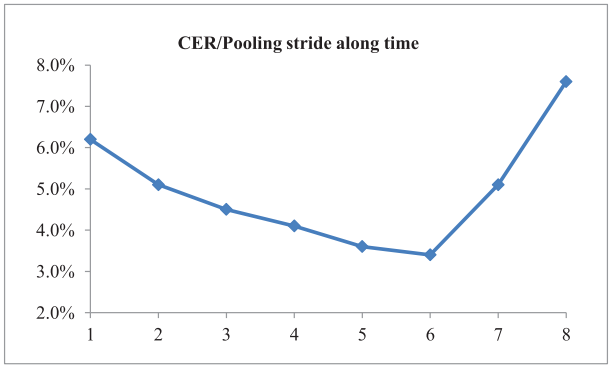

Figure 12 | Effect of the pooling stride along time for CER on RIMES.

epochs of training and test, and the final mean and standard deviation are both optimal. That is to say our method has more stable training procedure and labeling performance. This conclusion can also be seen from Figures 7–9 in which our method has a faster and more stable change curve than other models. So we can say that the proposed convolutional recurrent network is more suitable for sequence labeling tasks than purely RNN, CNN or their combination-based model.

From Table 3, we can see that we achieve a result second only to paper [35] for phoneme sequence recognition. A relatively close result to us is paper [16] which not only adopts attention mechanism but also uses special convolutional features and smooth focus. If without these external processing, it could only achieve 18.7% which is far from our method. At the same time, paper [35] uses a network-in-network configuration based on CNN, which is trained in two steps and needs to get frame-level labels for training through forced alignment, so it is more complicated than our method and it does not realize end-to-end sequence labeling. A very similar work is paper [20], which constructs recurrent convolution layer for speech recognition. But its core module is convolution layer embedded with RNN. So it is far different from our method, and it doesn't use pooling along time and CTC to achieve end-to-end training. Naturally, its method has a worse result than ours.

From Table 7, we can see that our method gets a third result on IAM and a second result on RIMES. The closest result to ours is paper [37] which achieves a slightly worse result on IAM and a slightly better result on RIMES than ours. The method adopts stacked CNN, LSTM and CTC for character sequence recognition, but our method has a better result than it. Obviously, paper [38] and [39] still achieve better results than ours. But both of them have multiple special layers with parameters including spatial transformer network, residual convolutional blocks, BLSTM, etc. So their architecture is much more complex than ours. Other than that, data augmentation, dropout and attention are all used in their methods. In contrast, our method is just a general architecture for end-to-end sequence labeling without any other auxiliary components. Even so, we still achieve quite good result.

From Table 8 and Figures 10–12 we can see that it is effective to use pooling along time on sequence labeling tasks with input longer than output, but the pooling stride has significant impact on the results. As the stride increases from 0 to 4, the PER is keeping declining, but when it exceeds 4, the PER increases sharply. Similar phenomena also appear in handwritten recognition task in which the inflection point is 6. The reason for this is that when the stride of the pooling along time is less than the number of elements in input sequence a single phoneme or character corresponding (such as a phoneme of 4 frames, a character of 6 pixel column), some redundant information can be reduced effectively, but if the stride is too large, it may cause serious loss of useful information and then lead to a performance degradation.

In summary, although our method is purely based on CRNN with CTC, we still achieve a good result on phoneme sequence recognition and handwritten character sequence recognition without any auxiliary component. Besides, the simplified architecture could also bring improvement to its training and annotation performance. It can be seen obviously form Figures 7–9 that our method has a more stable and faster training procedure than other models. In

addition, with pooling along time, the length of the output label vector sequence can be reduced proportionally, and the speed of the decoding from the label vector sequence will also be greatly improved because of the reduced search space. In this case, though we haven't got the best performance on the two tasks so far, we still exceed most of the special solutions with a more simplified architecture which could also bring faster training and decoding procedure. So we can say that our method is worth introducing here and is very promising for an improvement in the future with some special equipment.

In fact, the reason for the efficiency of our method can be explained from the following perspectives: first, in terms of feature extraction for sequence labeling tasks, the local spatial features and long-distance temporal-dependence features are both very important for annotation of a single element in the input sequence. Although CNN is good at modeling local spatial features and RNN is good at modeling long-distance temporal features, neither of them can achieve joint modeling of the two in an effective way. Fixed filter size of the CNN restricts its ability to capture the long-distance temporal correlation, and RNN is just good at extracting temporal correlation but less effective on local spatial features due to its own structural characteristics. At the same time, traditional combined architecture in a cross-stack manner is only able to achieve a compromise of the two types of features, but far from joint extraction. In contrast, CRNN could remedy their shortcomings effectively, and thus achieve a better performance. Second, in terms of sequence labeling tasks such as phoneme sequence recognition or character sequence recognition, although the actual input sequence contains hundreds of feature vectors, the target sequence is usually much shorter. That is to say there are several continuous feature vectors corresponding to a same label. Traditional deep models usually predict a label for each individual feature vectors, which undoubtedly contains redundancy. The use of pooling along time could transform multiple input vectors into just one output label which seems to be helpful to reduce the redundancy. Finally, in addition to the structural characteristics of the network, parameter tuning is still a critical step. In this paper, although the structure of the models applied to phoneme recognition and handwritten character recognition is similar, their parameters are not exactly the same (e.g. pooling stride). Therefore, how to tune the parameters is still an important problem in sequence labeling models based on neural networks.

5. CONCLUSION AND FUTURE WORK

In this work, we present a novel "end-to-end" sequence labeling framework which combines convolution structure and recurrent connections in a unified manner. The framework could use recurrent connections to perform long-distance temporal features modeling, use convolution structure to perform local spatial feature modeling, use 2-D pooling on sequence to reduce its local spatial spectral and short-distance temporal variation and use CTC to achieve "end-to-end" training. Experimental results show that our method achieves comparable performance to state-of-art results on phoneme sequence recognition and handwritten character recognition while having a more simplified architecture with more efficient training and labeling procedure. Unlike the traditional combined models, we integrate convolution and recurrent network in a unified manner and give full play to advantages of both. This is just suitable for sequence labeling tasks. In the future, we would like to apply the network to the two tasks with some special domain processes to construct a superior architecture in order to achieve state-of-art result.

CONFLICT OF INTEREST

The authors declare that they have no competing interests.

AUTHORS' CONTRIBUTIONS

The work was conceived and designed by Xiaohui Huang and Lisheng Qiao. Experiments were performed by Xiaohui Huang and the paper was written by Xiaohui Huang and Wentao Yu. Jing Li and Yanzhou Ma reviewed and edited the manuscript. All authors read and approved the manuscript.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for the constructive comments, and this work is sponsored by the National key research and development program of China (No. 2016YFB0201402).

REFERENCES

[1] A. Graves, S. Fernández, J. Schmidhuber, Connectionist temporal classification: labeling unsegmented sequence data with recurrent neural networks, in Proceedings of the 23rd International Conference on Machine Learning, ACM, Pittsburgh, PA, USA, 2006, pp. 369–376.

[2] J. Watanabe, S. Hori, T. Baskar, *et al.*, Language model integration based on memory control for sequence to sequence speech recognition, in IEEE International Conference on Acoustics, Speech, and Signal Processing, Brighton, UK, 2019, pp. 6191–6195.

[3] J. Sueiras, V. Ruiz, A. Sanchez, J. Velez, Offline continuous handwritten recognition using sequence to sequence neural networks, Neurocomputing. 289 (2018), 119–128.

[4] A. Graves, J. Schmidhuber, Multidimensional recurrent neural networks, in International Conference on Artificial Neural Networks, Porto, Portugal, 2007, pp. 549–558.

[5] A. Naseer, K. Zafar, Meta features-based scale invariant OCR decision making using LSTM-RNN, Comput. Math. Organ. Theor. 5 (2019), 165–183.

[6] Y. Zhang, M. Pezeshki, Towards end-to-end speech recognition with deep convolutional neural networks, in Interspeech 2016, San Francisco, CA, USA, 2016, pp. 410–414.

[7] M. Karafiát, M.K. Baskar, S. Watanabe, *et al.*, Analysis of multilingual sequence-to-sequence speech recognition systems, in Interspeech 2019, Graz, Austria, 2019.

[8] A. Graves, N. Jaitly, Towards end-to-end speech recognition with recurrent neural networks, in Proceedings of the 31st International Conference on Machine Learning (ICML-14), Beijing, China, 2014, vol. 32, pp. 1764–1772.

[9] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (1997), 1735–1780.

[10] A. Graves, A.R. Mohamed, G. Hinton, Speech recognition with deep recurrent neural networks, in 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP2013), Vancouver, Canada, 2013, pp. 6645–6649.

[11] H. El Bahi, A. Zatni, Text recognition in document images obtained by a smartphone based on deep convolutional and recurrent neural network, Multimed. Tools Appl. 78 (2019), 26453–26481.
[12] Q. Lu, Y. Xu, R. Yang, N. Li, C. Wang, Serial and parallel recurrent convolutional neural networks for biomedical named entity recognition, Database Syst. Adv. Appl. 11448 (2019), 439–443.
[13] D. de Benito-Gorron, A. Lozano-Diez, D.T. Toledano, *et al.*, Exploring convolutional, recurrent, and hybrid deep neural networks for speech and music detection in a large audio dataset, EURASIP J. Audio Speech Music Process. 2019 (2019).
[14] Y. Wang, F. Liu, K. Zhang, G. Hou, Z. Sun, T. Tan, LFNet: a novel bidirectional recurrent convolutional neural network for light-field image super-resolution, IEEE Trans. Image Process. 27 (2018), 4274–4286.
[15] D. Bahdanau, J. Chorowski, D. Serdyuk, *et al.*, End-to-end attention-based large vocabulary speech recognition, in 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP2016), Shanghai, China, 2016, pp. 4945–4949.
[16] J.K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, Y. Bengio, Attention-based models for speech recognition, in Advances in Neural Information Processing Systems 28 (NIPS 2015), Montreal, Canada, 2015, pp. 577–585.
[17] B. Li, T. Liu, Z. Zhao, X. Du, Attention-based recurrent neural network for sequence labeling, Web and Big Data. 10987 (2018), 340–348.
[18] L. Kang, J.I. Toledo, P. Riba, M. Villegas, *et al.*, Convolve, attend and spell: an attention-based sequence-to-sequence model for handwritten word recognition, in International Conference on Pattern Recognition, Beijing, China, 2018, vol. 11269, pp. 459–472.
[19] T.N. Sainath, O. Vinyals, A. Senior, H. Sak, Convolutional, long short-term memory, fully connected deep neural networks, in 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brisbane, Australia, 2015, pp. 4580–4584.
[20] Y. Zhao, X. Jin, X. Hu, Recurrent convolutional neural network for speech processing, in IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP2017), New Orleans, LA, USA, 2017.
[21] Z. Zhang, D. Robinson, J. Tepper, Detecting hate speech on twitter using a convolution-GRU based deep neural network, in The 15th European Semantic Web Conference (ESWC 2018), Heraklion, Greece, 2018, vol. 10843, pp. 745–760.
[22] B. Shi, X. Bai, C. Yao, 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. 36 (2017), 2298–2304.
[23] O. Vinyals, A. Toshev, S. Bengio, *et al.*, Show and tell: lessons learned from the 2015 MSCOCO image captioning challenge, IEEE Trans. Pattern Anal. Mach. Intell. 39 (2016), 652–663.
[24] F. Zhou, R. Hang, Q. Liu, X. Yuan, Integrating convolutional neural network and gated recurrent unit for hyper-spectral image spectral-spatial classification, in Pattern Recognition and Computer Vision, Guangzhou, China, 2018, pp. 409–420.
[25] X. Wang, W. Jiang, Z. Luo, Combination of convolutional and recurrent neural network for sentiment analysis of short texts, in The 26th international conference on Computational Linguistics, Osaka, Japan, 2016, pp. 2428–2437. https://www.aminer.cn/pub/58d83051d649053542fe99f1
[26] M.Z. Alom, C. Yakopcic, M.S. Nasrin, *et al.*, Breast cancer classification from histopathological images with inception recurrent residual convolutional neural network, J. Digit. Imaging. 32 (2019), 605–617.
[27] P. Xie, G. Wang, C. Zhang, *et al.*, Bidirectional recurrent neural network and convolutional neural network (BiRCNN) for ECG beat classification, in The 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC2018), Honolulu, HI, USA, 2018, pp. 2555–2558.
[28] S. Lai, L. Xu, K. Liu, J. Zhao, Recurrent convolutional neural networks for text classification, in Proceedings of the Twenty-Ninth AAAI Conference on Artificial Inteligence, Austin, TX, USA, 2015, vol. 333, pp. 2267–2273.
[29] K. Cho, B. Van Merrienboer, C. Gulcehre, *et al.*, Learning phrase representations using RNN encoder-decoder for statistical machine translation, in Proceeding of the Conference on Empirical Methods Natural Language Process, Doha, Qatar, 2014, pp. 1724–1734.
[30] M. Schuster, K.K. Paliwal, Bidirectional recurrent neural networks, IEEE Trans. Signal Process. 45 (1997), 2673–2681.
[31] J.S. Garofolo, L.F. Lamel, W.M. Fisher, J.G. Fiscus, D.S. Pallett, DARPA TIMIT Acoustic-Phonemetic Continuous Speech Corpus CD-ROM.NIST Speech Disc 1-1.1, NASA STI/Recon Technical Report, vol. 93, 1993.
[32] U. Marti, H. Bunke, The IAM-database: an English sentence database for off-line handwritten recognition, J. Doc. Anal. Recognit. 5 (2002), 39–46.
[33] E. Grosicki, H. El-Abed, ICDAR 2011-french handwriting recognition competition, in International Conference on Document Analysis and Recognition, Beijing, China, 2011, pp. 1459–1463.
[34] K.F. Li, H.W. Hon, Speaker-independent phoneme recognition using hidden markov models, J. Acoustical Soc. Am. 84 (1988), 62.
[35] L. Toth, Combining time- and frequency-domain convolution in convolutional neural network-based phone recognition, in IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Florence, Italy, 2014, pp. 190–194.
[36] A. Chowdhury, L. Vig, An efficient end-to-end neural model for handwritten text recognition, arXiv: 1807.07965 [cs.CL], 2018.
[37] J. Puigcerver, Are multidimensional recurrent layers really necessary for handwritten text recognition?, in IEEE 2017 IAPR 14th International Conference on Document Analysis and Recognition (ICDAR 2017), Kyoto, Japan, 2017.
[38] K. Dutta, P. Krishnan, M. Mathew, C.V. Jawahar, Improving CNN-RNN hybrid networks for handwriting recognition, in 16th International Conference on Frontiers in Handwriting Recognition (ICFHR), Niagara, NY, USA, 2018.
[39] J. Michael, R. Labahn, T. Grüning, *et al.*, Evaluating sequence-to-sequence models for handwritten text recognition, arXiv: 1903.07377v2 [cs.CV], 2019.
[40] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE. 86 (1998), 2278–2324.
[41] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, in The 3rd International Conference on Learning Representations (ICLR2015), San Diego, CA, USA, 2015. arXiv:1409.1556[cs.CV].