Study the Preprocessing Effect on RNN Based Online Uyghur Handwriting Word Recognition

Wujiahemaiti Simayi, Mayire Ibrayim, and Askar Hamdulla(✉)

Institute of Information Science and Engineering,
Xinjiang University, Urumqi, China
askar@xju.edu.cn

Abstract. There is little work done on unconstrained handwritten Uyghur word recognition by implementing deep neural networks. This paper carries out a comparative study to see the preprocessing effect on training a neural network based online handwriting Uyghur word recognition system. Bidirectional recurrent neural network with connectionist temporal classification is implemented for unconstrained handwriting word recognition experiments on a dataset of 23400 Uyghur word samples. The results are directly obtained from model output without any lexicon or language model. Experiments showed that proper preprocessing can improve the training speed very effectively. The comparative study conducted in this paper can be good reference for later studies.

Keywords: Online handwriting recognition · Preprocessing · Input representation · Recurrent neural networks · Connectionist temporal classification · Uyghur words

1 Introduction

Online handwriting recognition is conducted on the traced pen-tip movement trajectory information [1]. The first-hand online handwritten samples are usually coarse trajectories that need proper preprocessing. Although, deep neural networks have been showing their strength to learn from raw input [2], good preprocessing can alleviate the need for very large number training data and improve model generalization [3]. Shortening the handwritten trajectory is helpful to speed up network training at least. This paper conducts comparative experiments to see the effect of preprocessing on model training process on Uyghur online handwritten word samples, based on recurrent neural network and connectionist temporal classification-CTC [7].

Uyghur is a typical alphabetic script which of 32 basic character/letter types. Each character type has several specific character forms which are selectively used based on the character position within a word. There are 128 character forms in Uyghur alphabet. Most handwriting word recognition studies so far have been being conducted using a holistic approach or with help of certain lexicon [4]. The applied model in this paper

H. Song et al. (Eds.): SIMUtools 2019, LNICST 295, pp. 448–458, 2019.
https://doi.org/10.1007/978-3-030-32216-8_43

maps handwritten word trajectory into string of characters in Uyghur alphabet directly without external lexicon help.

Study on Artificial intelligent systems has been gaining more and more attention in many fields [11, 12]. The handwriting word recognition experiments in this paper will be a reference for later study and development of intelligent systems in this field, too. The remainder of this paper is organized as follows. Section 2 introduces the performed preprocessing methods in detail. Section 3 describes the applied model for online handwriting word recognition. Preprocessing effects and experiment results are given in Sect. 4 a brief conclusion is drawn in Sect. 5.

2 Preprocessing

Figure 1 illustrates some online handwritten Uyghur word samples where temporal neighbor strokes are drawn using different colors and annotated at the beginning of each stroke. We can see that randomness and invasion to writing regulation in all handwritten samples.

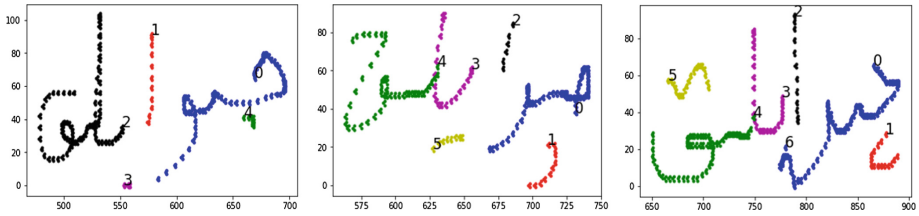


Fig. 1. Different handwritten word samples for a word

Preprocessing is hoped to decrease the disturbing content in raw trajectory and improve the input representation thus can alleviate the need for large volume of data [3]. The preprocessing techniques applied in this paper includes redundant removing, point insertion, sampling, and turn-point selection. Figure 2 shows the workflow of preprocessing operations implemented in our experiments.

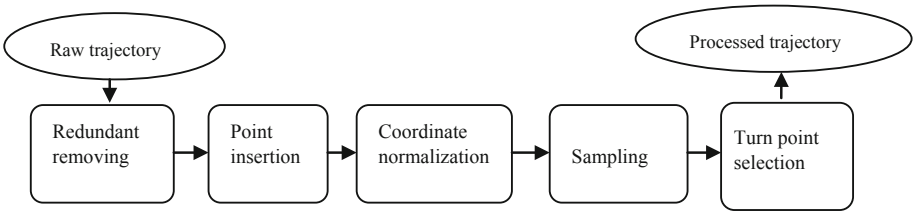


Fig. 2. Preprocessing workflow on handwritten trajectory

2.1 Remove Redundant Points

A handy threshold based redundant removing technique is implemented on each stroke of handwritten word trajectory. The removing thresholds are set based on average neighbor point distance in the stroke trajectory. Each neighbor point couples are visited and treated according to removing conditions. The distance between neighbor points are calculated using Eq. (1).

$$D = |P_i - P_{i-1}| = \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2} \tag{1}$$

where $P_i(x_i, y_i)$ and $P_{i-1}(x_{i-1}, y_{i-1})$ are point coordinates of a point and its previous neighbor, D refers the distance between the neighbor points.

If the distance from a point to its previous neighbor is greater than 3 times of average neighbor-point distance, this point is treated as noise and removed. 1/2 of the average neighbor point distance is used as duplication removing threshold. If the distance from a point to its previous neighbor is smaller than the duplication removing threshold, this point is removed from the trajectory.

2.2 Point Insertion

Point insertion is made between the neighbor points which have larger distance than threshold value. For avoiding generated extra points between strokes, point insertion to sample trajectory is applied on stroke level only. The insertion threshold is set by 0.01 times of the larger criteria of width or height of sample shape, so it is varied sample to sample.

$$N = \frac{|P_1 - P_2|}{thr\_d} \tag{2}$$

$$x_i = x1 + \frac{\Delta x}{N} * i \tag{3}$$

$$y_i = y1 + \frac{\Delta y}{N} * i \tag{4}$$

In Fig. 3, $P_1(x_1, y_1)$ and $P_2(x_2, y_2)$ are the neighbor points that need point insertion between them. Δx and Δy are the horizontal and vertical distances calculated by their coordinates. Euclidian distance between them is obtained using Eq. (1). N is The number of points to be inserted and thr_d is the distance threshold for point insertion, see Eq. (2). The coordinates of each insertion point is set according to Eqs. (3) and (4) where (x_i, y_i) are the coordinate values of the i^{th} inserted point.

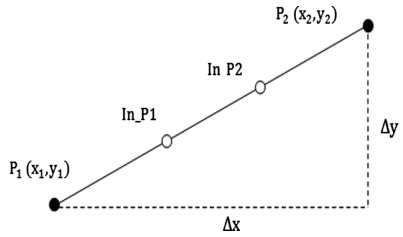


Fig. 3. Inserting points

2.3 Coordinate Normalization

The recorded point coordinates are normalized to be within certain interval by simple min-max normalization using Eqs. (5)–(8). Points in trajectory are moved to be zero started and within (0, 1) values both horizontal and vertical coordinates.

$$W = max(X) - min(X) \tag{5}$$

$$H = max(Y) - min(Y) \tag{6}$$

$$x_i = \frac{X_i - min\,X}{max(W, H)} \tag{7}$$

$$y_i = \frac{Y_i - min(Y)}{max(W, H)} \tag{8}$$

where (X_i, Y_i) and (x_i, y_i) are point coordinate values before and after normalization; X and Y represents the sets of horizontal and vertical coordinates of a sample trajectory, respectively. The normalizing factor (denominator) uses the maximum criteria of shape width W and height H, thus the aspect ratio of the sample shape is kept.

2.4 Sampling

In order to make even handwritten trajectory, an equal distance based sampling is implemented. Stroke based sampling with distance threshold is applied on each stroke to avoid missing delayed strokes which are crucial to distinguish characters and words. Specifically, if the distance between a point and its previous neighbor is smaller than threshold distance, this point is discarded; otherwise, it is selected and kept as sampled point.

2.5 Turn Point Selection

Turn-points are very much informative that they express substantial direction change of pen-tip movement during handwriting. Using only turn-points greatly decreases the trajectory length than using all trajectory points. Again, selecting turn points is performed on stroke level in order to avoid losing character distinguishing marks.

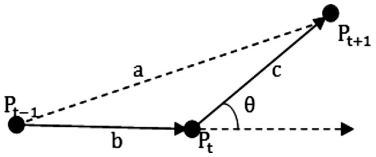
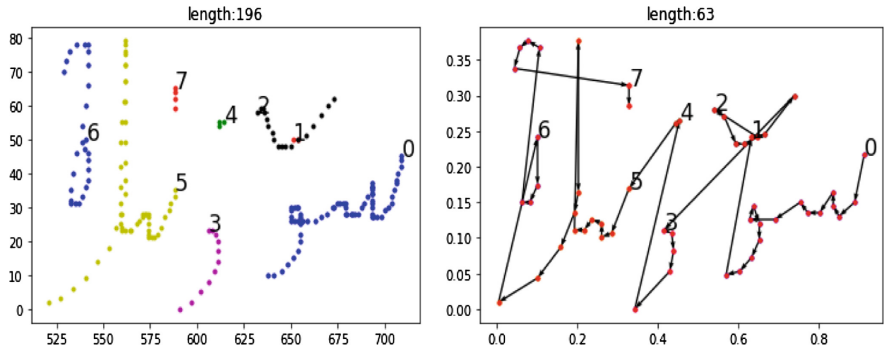


Fig. 4. Pen-tip direction change

Direction change θ at a point in trajectory is obtained by its previous and next neighbor points using Eq. (9), where P_{t-1} and P_{t+1} are the previous and next neighbor points of point P_t; And the Euclidian distances between them are noted as a, b, c as illustrated in Fig. 4. A point in trajectory is detected as critical turn point if direction change θ exceeds threshold of $\pi/12$.

$$\theta = \pi - arccos\left(\frac{b^2 + c^2 + a^2}{2bc}\right) \tag{9}$$

Figure 5 compares the visual effect of the handwritten word samples before and after preprocessing. The original trajectory has very large values and a great number of points, as shown in Fig. 5(a). After preprocessing, the coordinate values are squeezed to be within (0,1). And the trajectory length has got reduced significantly while still keeping readability, as in Fig. 5(b).



(a) Before preprocessing (b) After preprocessing

Fig. 5. Visual effect of preprocessing on handwritten trajectory

3 Unconstrained Handwriting Word Recognition System

3.1 Input Representation

Two input representation are used in the comparative training experiments in this paper. First one simply uses two dimensional [x,y] values. The second representation

uses six dimensions which include point coordinates (x,y), pen-tip movement direction $(\Delta x, \Delta y)$, two dimensional pen-tip state of up or down [6]. Pen-tip movement direction $(\Delta x, \Delta y)$ is easily got by calculating the differences of neighbor point positions by Eqs. (10) and (11). Then pen-state is known as pen-down if a point stays in the same stroke with his previous neighbor and marks as [1, 0], otherwise marked as [0,1] which notes pen-up state.

$$\Delta x = x_t - x_{t-1} \tag{10}$$

$$\Delta y = y_t - y_{t-1} \tag{11}$$

where (x_t, y_t) and (x_{t-1}, y_{t-1}) are the coordinates of current and adjacent previous points within trajectory.

3.2 System Architecture

A recurrent neural network based unconstrained handwriting word recognition system with LSTM cells in recurrent layers, as shown in Fig. 6(a), is applied to recognize online handwritten word trajectories in this paper.

The bidirectional recurrent layer in the system has two sub-layers (forward and backward) which read the input sequence in opposite directions, either right-left of left-right [6]. Each cell in a recurrent layer controls input, output and state values to the next state with gate mechanism. The outputs of sub-layers, noted $Y_{forward}$ and $Y_{backward}$, are concatenated get overall output of the bidirectional recurrent layers as in Eqs. (12)–(14) and Fig. 6(b).

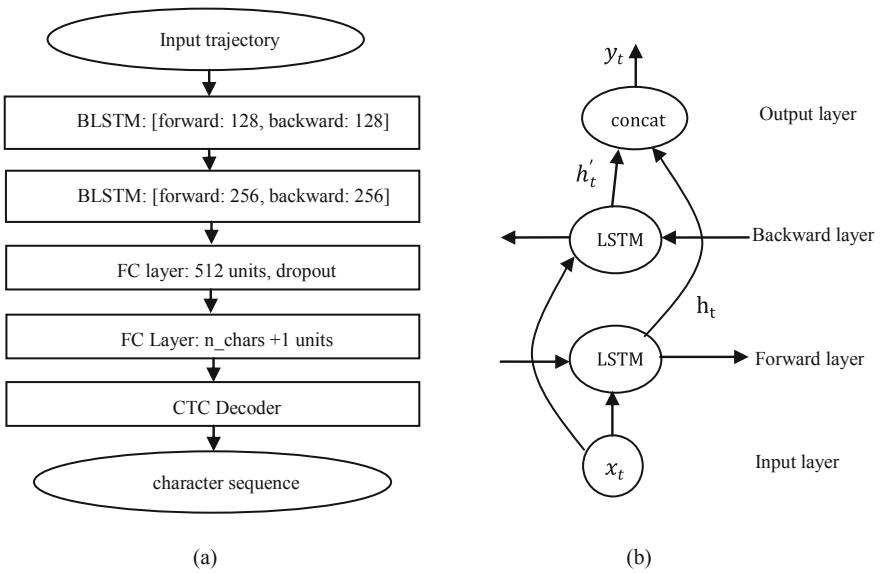


Fig. 6. Architecture of unconstrained handwritten word recognition system (a) Model Architecture (b) Bidirectional LSTM layer

$$Y_{forward} = h_t = [y_{l1}, y_{l2}, \ldots . y_{lN}] \tag{12}$$

$$Y_{backword} = h_t^{'} = [y_{r1}, y_{r2}, \ldots . y_{rN}] \tag{13}$$

$$Y = concat\left(Y_{forward}, Y_{backward}\right) \tag{14}$$

Where y_{rN} and y_{lN} represents the output of the N^{th} node of in right-left and left-right sub-recurrent layers. $Y_{forward}$ and $Y_{backward}$ are the outputs of the two inverse sub-layers and Y is the of the bi-directional recurrent layer. The fully connection layers learn the further generalized features from recurrent layers.

This paper uses connectionist temporal classification to make character string outputs directly from the handwriting trajectory input to realize unconstrained handwriting word recognition system [7]. Since the ground truth word transcriptions are based on specific character shapes, the last fully connection layer is equipped with 128+1 units. Then CTC calculates the most possible character sequence as output.

4 Dataset and Configuration

4.1 Dataset

A total of 23400 online handwritten Uyghur word samples have been collected from 26 writers for 900 word classes. Each writer is asked to write all word classes continuously on handwriting tablet in order to make the collected samples more natural and challenging. Each handwritten word sample in the established dataset contains the recorded pen-tip coordinates on handwriting tablet, associated Unicode based ground-truth word transcription and some general information such as trajectory length, total number of strokes etc. The samples from 22 writers are arranged to be in train set, while the ones of remained 4 writers are used to be the test set. The train set has 19800 samples and test set contains 3600 samples for 900 word classes, respectively.

4.2 Configurations

Training experiments are performed on TitanX GPU with 12G RAM. During training, a small portion of train samples (10 batches) is used as validation set and not participated in model parameter adaptation. Character error rate CER and character accurate rate CAR [8] are used as main evaluation metrics by Eqs. (15) and (16).

$$CER = \frac{D_e + S_e + l_e}{N_t} \tag{15}$$

$$CAR = 1 - \frac{D_e + S_e + l_e}{N_t} \tag{16}$$

where N_t is number of total characters in reference text; D_e, S_e, I_e denote the substitution, deletion and insertion errors, respectively.

All variables of the network are initialized by variance scaling initialize [9]. Stochastic gradient descent with Adam optimizer is applied for all experiments [10]. Initial learning rate is set 0.001 and decreased by half when no improvement seen in continues 3 epochs. No regularization except dropout is implemented with drop-rate of 0.5. Training is stopped after 10 epochs failed to make any progress on validation set. In order to save the training time, only 10 batches of the train and validation sets are used to navigate the model performance during training with CER metric.

5 Experiments and Discussion

5.1 Experiments

In order to see the effect of preprocessing on training process, the same training configurations are implemented for all comparative experiments. In the first group of experiments, two dimensional input representations respectively by raw, normalized and preprocessed trajectory values are used as input. The second group of experiments uses six dimensional input representations respectively based on raw, normalized and preprocessed trajectories, again. In the context of these experiments1, the raw, coordinate normalized and preprocessed trajectories are noted by Raw-[x,y], Norm-[x,y] and Prep-[x,y], respectively. The six dimensional inputs are also noted by Raw-dim6, Norm-dim6 and Prep-dim6 accordingly in Table 1 and Fig. 7.

5.2 Discussion on Results

Speed. The speed improvement in RNN training is very much preferred, because RNNs are usually slow for their recurrent connections. In training, the shortened trajectories such as Prep-[x,y] and Prep-dim6 representations have benefited almost 3 times speed-up over the ones with original trajectory lengths. As given in Table 1, the shortened representations only take around 5 min per training epoch while representations with original lengths need longer than 15 min. Recognition on short input trajectories is also faster than long sequences, too. The Prep-[x,y] and Prep-dim6 representations needed around 0.028 s to recognize a sample by average. Recognition time for a sample from the raw or the normalized only (Norm) representations took averagely around 0.09 s.

Performance in Training. In both group of comparative experiments, the training on raw trajectories experienced very unpleasant journey and ended up with severe divergence. In the following discussion, we only give figures of training process on the normalized and preprocessed trajectories.

Table 1. Comparison handwriting recognition systems

Input	Mean Seq_len	Model Size(M)	No.ep	T/ep (min)	Tr_CER (%)	Te_CER (%)	Te_CAR (%)	Av-recT (s)
Raw-[x,y]	221	16.2M	73	~15.6	35.73	40.54	59.46	0.092
Norm-[x,y]	211	16.2M	105	~15.6	11.81	21.94	79.06	0.092
Prep-[x,y]	67	16.2M	66	~5.1	15.99	23.59	76.41	0.025
Raw-dim6	221	16.25M	80	~16	–	–	–	0.095
Norm-dim6	211	16.2M	72	~16	**6.92**	**19.32**	**93.06**	0.095
Prep-dim6	58	16.25M	100	~5.3	11.61	23.48	76.52	0.028

Where Tr_CER and Te_CER are the character error rates on train and test sets, respectively, Te_CAR means character accurate rate on test set; No.ep and T/ep the number of epochs that training stopped and average time spent per epoch. Av-recT means average recognition time per sample

Two Dimensional Representation. Since raw coordinate values of points are very large, training on Raw-[x,y] representation was hard and suffered by incredibly big fluctuation. In contrast, Normalized coordinate values experienced very good training performance despite some zig-zags on error curve in Fig. 7(a). Prep-[x,y] based Training performed a very fast improvement of model performance at the beginning 30 epochs for, see Fig. 7(b). There is no obvious overfitting found during training but some degree of fluctuation accompanied from the beginning to the end. However, it is very obvious that the shorted trajectory loses much information from the original handwritten trajectory. Therefore, Norm-[x,y] inputs which kept all information from the original trajectory produced better performance than Prep-[x,y] representation.

Six Dimensional Representation. Raw-dim6 representation couldn't make pleasing performance both on train and validation sets, again. The normalized coordinate values from Norm-dim6 produced almost steady decline of error during training as plotted in Fig. 7(c). The training and validation curves of the six dimensional representation indicate better model generality than of two dimensional ones. The difference of training and validation errors is small until the training prepared to stop. Norm-dim6 representation obtained the highest recognition performance on whole train and test sets comparing with other comparative experiments in this paper, see Table 1. Comparing with Norm-[x,y] representation, more information in Norm-dim6 made faster convergence, too. Prep-dim6 showed the fastest convergence on train set among the experiments that 40 epochs reached train error rate of 5% (CER). After 30 epochs of training, see Fig. 7(d), training saw some degree of overfitting although model showed almost steady performance in later epochs. This may indicate that smaller models will be preferred for Prep-dim6 representation. Among all experiments, The Norm-dim6 representation found itself best fit for the model capacity and reached a good generalization of 19.32% CER on test set which is best recognition result among the compared training experiments in this paper and can be further improved with proper training configurations.

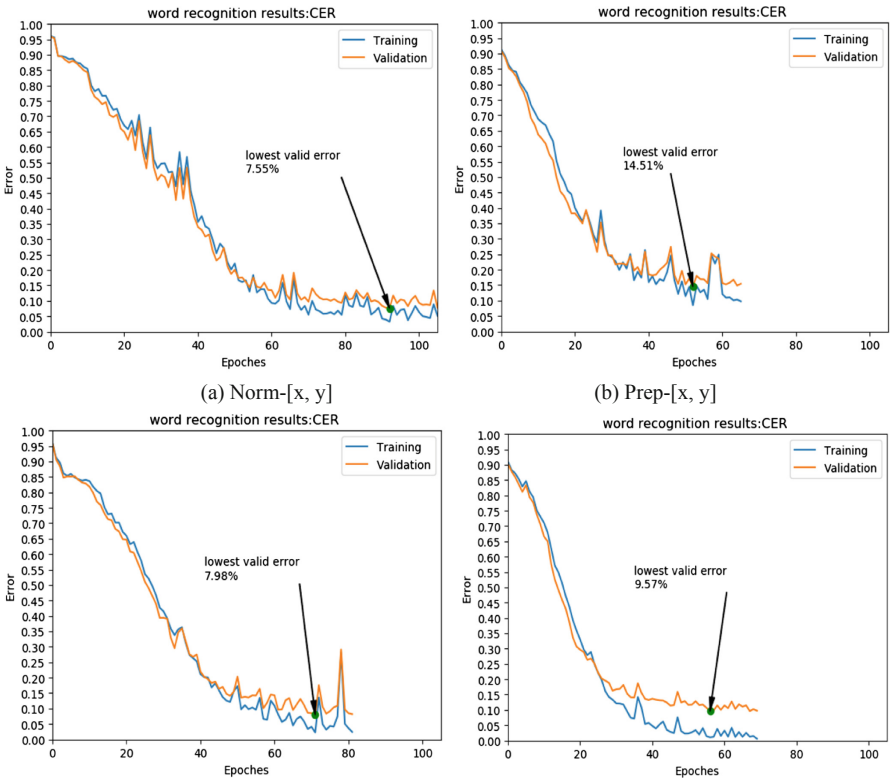


Fig. 7. Training records for online handwritten word recognition (The results are based on 10 batches from train and validation sets)

6 Conclusion

In both groups of comparative training experiments using two and six dimensional input representations, the pre-processed inputs are found much faster to complete an epoch of training than using raw trajectory. Experiments showed that raw trajectories with large range of coordinate values are hard to be trained. Excessive preprocessing will lose trajectory information and may cause harm for model training. More informative input representation shows better training behavior than just using trajectory coordinates. The normalized trajectory coordinates with more features produced the best performance for training and generalization. Investigating the different deep neural network structures is the main content of our next work.

Acknowledgment. This work is supported by National Science Foundation of China (NSFC) under grant number 61462081 and 61263038. The first author is very much grateful to the National Laboratory of Pattern Recognition of CASIA for providing the experimental environment.

References

1. Liu, C.L., Yin, F., Wang, D.H., Wang, Q.F.: Online and offline handwritten Chinese character recognition: benchmarking on new databases. Pattern Recogn. **46**(1), 155–162 (2013)
2. Graves, A., Liwicki, M., Bunke, H., Schmidhuber, J., Fernández, S.: Unconstrained on-line handwriting recognition with recurrent neural networks. In: Conference on Neural Information Processing Systems, pp. 458–464. DBLP, Vancouver (2007)
3. Liu, C.L.: Handwritten Chinese character recognition: effects of shape normalization and feature extraction. In: Doermann, D., Jaeger, S. (eds.) SACH 2006. LNCS, vol. 4768, pp. 104–128. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78199-8_7
4. Simayi, W., Ibrayim, M., Tursun, D., Hamdulla, A.: A survey on the classifiers in on-line handwritten Uyghur character recognition system. Int. J. Hybrid Inf. Technol. **9**(3), 189–198 (2016)
5. Chammas, E., Mokbel, C., Likforman-Sulem, L.: Handwriting recognition of historical documents with few labeled data. In: 2018 13th IAPR International Workshop on Document Analysis Systems (DAS), 43–48. IEEE (2018)
6. Zhang, X.Y., Yin, F., Zhang, Y.M., Liu, C.L., Bengio, Y.: Drawing and recognizing Chinese characters with recurrent neural network. IEEE Trans. Pattern Anal. Mach. Intell. **40**(4), 849–862 (2018)
7. Graves, A., Fernández, S., Gomez, F, Schmidhuber, J.: Connectionist temporal classification: labeling unsegmented sequence data with recurrent neural networks. In: Proceedings of the 23rd International Conference on Machine Learning, Pittsburgh, Pennsylvania, USA, pp. 369–376. ACM, New York (2006)
8. Su, T.H., Zhang, T.W., Guan, D.J., Huang, H.J.: Off-line recognition of realistic Chinese handwriting using segmentation-free strategy. Pattern Recogn. **42**(1), 167–182 (2009)
9. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1026–1034. IEEE, Santiago (2015)
10. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. In: The 3rd International Conference for Learning Representations. http://arxiv.org/abs/1412.6980, San Diego (2015)
11. Jiang, D., Huo, L., Lv, Z., et al.: A joint multi-criteria utility-based network selection approach for vehicle-to-infrastructure networking. IEEE Trans. Intell. Transp. Syst. **19**(10), 3305–3319 (2018)
12. Jiang, D., Li, W., Lv, H.: An energy-efficient cooperative multicast routing in multi-hop wireless networks for smart medical applications. Neurocomputing **2017**(220), 160–169 (2017)