Speech2Stroke: Generate Chinese Character Strokes Directly from Speech

Yinhui Zhang(✉), Wei Xi, Zhao Yang, Sitao Men, Rui Jiang, Yuxin Yang, and Jizhong Zhao

School of Computer Science and Technology, Xi'an Jiaotong University, Xi'an, China
manli0826@gmail.com, weixi.cs@gmail.com, zhaoyang9425@gmail.com, sitao.men@gmail.com, ruijiang.jerry@gmail.com, yangdx6@gmail.com, zjz@xjtu.edu.cn

Abstract. Chinese character is composed of spatial arrangement of strokes. A portion of these strokes combines to form phonetic component, which provides a clue to the pronunciation of the entire character, the others combine to form semantic component, which indicates semantic level information for speech context. How closely the connection between the internal strokes of Chinese characters and speech? In this paper, we propose Speech2Stroke, a end-to-end model that exploits the phonetic and morphologic level information of pictographic words. Specifically, we generate strokes directly from the speech by Speech2Stroke. The performance of Speech2Stroke is evaluated by the specific stroke error rate(SER). The SER of the optimal model can achieve 20.61%. Through the experiments and analysis, we show that our model has the ability to capture the alignment between audio and the internal structures of pictographic characters.

Keywords: Deep learning · Stroke of Chinese character · Pictographic word

1 Introduction

When Chinese people encounter a previously unseen character without any pinyin or phoneme, but one can still pronounce the character according to its graphical shape, and guess the meaning. There is a closely connection between speech and internal graphical components, part of which is a direct result of the mechanisms of Chinese formation system.

The traditional six writings were first described by Xu Shen in his dictionary Shuowen Jiezi in the ancient Han dynasty [13], which consists of six kinds of principles of Chinese formation. One of six writings, pictophonetic compound, is

This work was supported by NSFC Grant No. 61832008, 61772413, 61802299, and 61672424.

H. Gao et al. (Eds.): CollaborateCom 2020, LNICST 349, pp. 83–94, 2021.
https://doi.org/10.1007/978-3-030-67537-0_6

by far the most numerous characters. Pictophonetic characters are composed of at least two components, characters with the same phonetic components share similar pronunciation. As shown in Table 1, all these simplified Chinese characters have the same right-hand side part, and their pronunciation is extremely similar except the tone.

Chinese characters have been regarded as a logographic language, the logograms of Chinese characters convey fruitful information of their meanings. Recent researches have proved that internal structural features are beneficial in the field of natural language processing (NLP): radicals, also as semantic components in most cases, are useful in learning Chinese word embedding and language understanding task[19,20]. Cao et al.[4] decompose the character into units of smaller granularity, and propose stroke n-grams for Chinese word embedding. Su and Lee et al. [16] use a convolutional auto-encoder to extract character feature from bitmap to represent the character glyphs, which shows that glyph embedding enhance word analogy and word similarity tasks. Meng et al. [12] utilize the Tianzige-CNN structures to extract the semantic glyph-vectors with historical Chinese scripts, which is proved to improve a wide range of Chinese NLP tasks due to the rich pictographic information in Chinese character. Unfortunately, the aforementioned methods only focus on the signific component of character, however, they neglect the phonetic component.

In this work, given the rich phonetic components in Chinese characters, we ask a question: can machine learn the relationship between speech and internal structures of Chinese characters? In our proposed model, we use a similar idea with [4], which decomposes the character into a sequence of strokes. With such stroke order information, we design a model that can exploit the sequential correction with audio input. Specificially, we use a neural network model that takes a short speech signal as input and predicts a stroke vector representing the internal structure. To train our model, we use the self-built stroke order data set, comprised of 32 different types of Chinese strokes, which has more diversity compare to [4]. Our model simply exploits the natural advantages of Chinese speech and internal structures of character, without requiring any other phoneme, e.g., pinyin.

The contributions of this paper are as follows: 1) To the best of our knowledge, our work is the first to explore a model for generating the internal structural strokes directly from the mandarin speech. 2) a new task learning for multi-task learning or transfer learning in mandarin speech recognition and speech synthesis.

The rest of this paper is organized as follows. In Sect. 2, we begin by introducing the architecture of Speech2Stroke model in detail. We describe the experimental setup in Sect. 3. We evaluate our models on the test set, and compare the different architectures and decoding methods in Sect. 4. The paper is concluded in Sect. 5.

Table 1. Pictophonetic characters

Character	Radical	Phonetic	Pronunciation
油 (grease)	氵 (liquid)	由	yóu
柚 (pomelo)	木 (plants)	由	yóu
铀 (uranium)	钅 (metal)	由	yóu
釉 (glaze)	釆 (distinguish)	由	yòu

2 Speech2Stroke Model

We provide an overview of our Speech2Stroke model. As shown in Fig. 1, our proposed model consists of two main pipelines: 1) character decompose, which takes Chinese characters as input, and produces a sequence of stroke orders that correspond to the associated character; 2) a speech encoder, which takes a spectrogram of power normalized audio clips as input, and predicts a stroke sequence. During the training stage, we train our speech encoder with Connectionist Temporal Classification (CTC) loss function in an end-to-end way. During predicting, we decode the stroke order from the output probability of speech encoder. In the following sub-sections, we describe each of these components in detail.

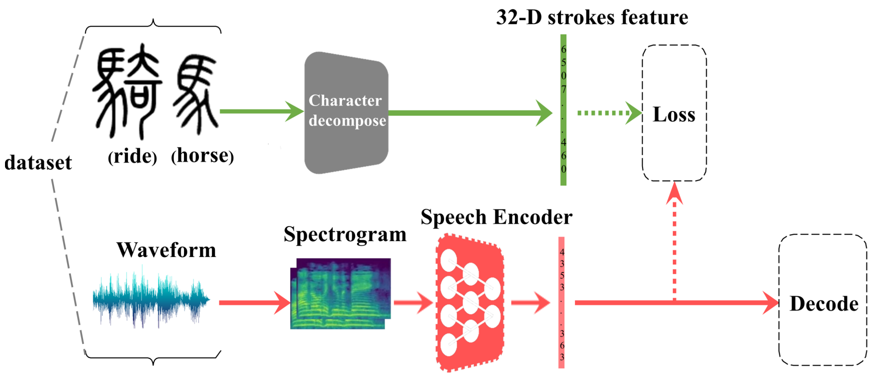


Fig. 1. The overall architecture of Speech2Stroke model

2.1 Character Decompose

A Chinese word is composed of a sequence of characters, and a Chinese character is composed of several graphical components. The characters with the same component share a similar pronunciation. The shape of the graphical components can be modeled by the stroke orders, which have finer granularity than radical. In our work, the strokes are categorized into 32 different types. As shown in Fig. 2, we use various strokes to depict the curved structure of character, therefore the structure information of character can be extracted completely from the stroke sequences, and we mark each stroke an integer ID from 1 to 32 respectively. The procedure of decomposing a word into IDs is shown in Fig. 3. First, we divide the simplified word into characters; then we retrieve the stroke sequence from every character, and add a blank space into the sequence of the strokes after finish a character. [1] provides an official guideline about the order for strokes involved in each simplified Chinese character; finally, we use integer IDs to represent the stroke sequence.

2.2 Speech Encoder

Let us define the sequence of acoustic feature vectors as $X = [X_1, X_2, X_3, ..., X_T]$, and $Y =$ [㇔, ㇐, ㇈,......] denotes a stroke sequence. The goal of speech encoder is to convert an input sequence X into a transcription Y, the most likely stroke sequence Y^* is given by:

$$Y^* = \arg\max_{Y} p(Y|X) \tag{1}$$

Our speech encoder is a neural network fed with spectrogram of power. At each output time-step t, the speech encoder predicts the probability, $p(y_t|x)$, where y_t is the stroke of Chinese or the blank symbol at time-step t. The overall model architecture is similar to DeepSpeech2 [2]. As shown in Fig. 4, the network is composed of convolutional neural network, bidirectional recurrent neural network, and fully connected network. In the bottom of speech decoder, the architecture that we experiment with consists of two convolutional layers. We consider the spectrogram as a picture and apply time-and-frequency domain 2-D convolution operations [15], which can capture the frequency and temporal features together. Convolution in frequency can model spectral variance due to speaker variability more compendiously, since it can preserve more of the vocal characteristics. Following the convolutional layers are one or more bidirectional recurrent layers, since recurrent network can exploit the sequence dependence of time series. It is a common sense in speech and language processing that using a more complex recurrent unit can allow the network to remember more time dependence, therefore the Long Short-Term Memory (LSTM) units [9] and the Gated Recurrent Units (GRU) [6] are applied respectively in bidirectional recurrent layers, though it is computationally expensive to train them. After the bidirectional recurrent layers, we stack one fully connected layer which can adjust the output dimension conveniently. The last layer is a softmax layer that computes a probability distribution over the strokes. The parameters of the speech encoder are updated through the backpropagation through time algorithm [18] . The cost function of training is CTC loss function. In the following section, we will detail the CTC loss function and how to decode the best transcription from the output probability distribution.

Stroke Name	Dot	Horizontal	Horizontal-hook	Horizontal-left-falling	Horizontal-left-falling-bend-hook	Horizontal-slant-hook	Horizontal-turning	Horizontal-turning-vertical-hook
Shape, ID	㇔,1	一,2	㇖,3	㇇,4	㇌,5	㇠,6	㇕,7	㇆,8
Stroke Name	Horizontal-turning-rise	Horizontal-turning-bend	Horizontal-turning-bend-hook	Horizontal-turning-turning	Horizontal-turning-turning-left-falling	Horizontal-turning-turning-turning	Horizontal-turning-turning-turning-hook	Right-falling
Shape, ID	㇊,9	㇍,10	㇈,11	㇅,12	㇋,13	㇎,14	㇡,15	㇏,16
Stroke Name	Left-falling	Left-falling-dot	Left-falling-turning	Vertical	Vertical-hook	Vertical-rise	Vertical-bend	Vertical-bend-horizontal-hook
Shape, ID	㇓,17	㇛,18	㇜,19	㇑,20	㇚,21	㇙,22	㇄,23	㇟,24
Stroke Name	Vertical-turning	Vertical-turning-left-falling	Vertical-turning-turning	Vertical-turning-turning-hook	Rise	Bend-hook	Recline-hook	Slant-hook
Shape, ID	㇗,25	ㄣ,26	㇞,27	㇉,28	㇀,29	㇁,30	㇃,31	㇂,32

Fig. 2. Shapes of Chinese strokes

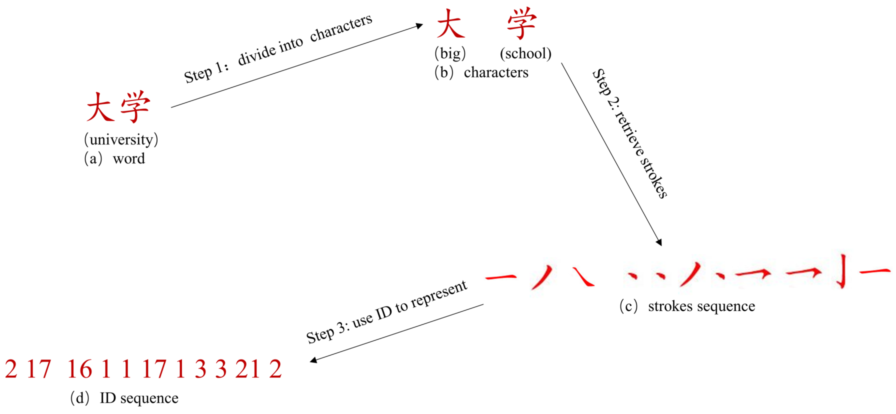


Fig. 3. Procedures of the generation of strokes from a word

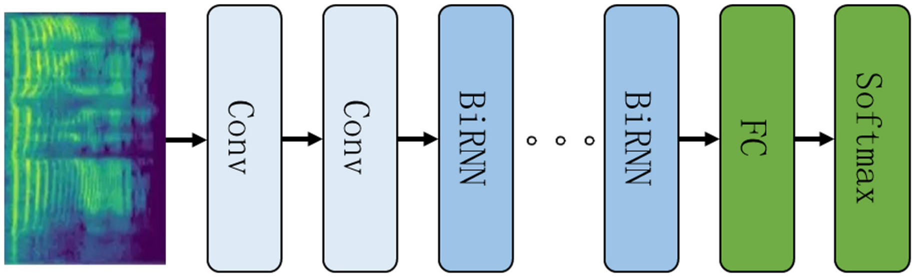


Fig. 4. Architecture of speech encoder

2.3 Connectionist Temporal Classification

Connectionist Temporal Classification (CTC) [7] is a popular method for end-to-end sequence learning, which usually applies in an RNN-based model to optimize predictions of the transcription sequence, it enables the end-to-end model training without pre-defined alignment information of training dataset. In the theory of CTC, let us denote x as the spectrogram frame sequence and y as the label sequence. CTC proposed a dynamic constrained procedure that adds a blank symbol ϕ as an additional label, the blank symbol can model the situation of pause and adjacent repeated labels in label sequence. The output of softmax of speech encoder has the same time-steps as the input sequence x, since the length of y should be shorter than the length of x due to the blank symbol. A CTC path π is defined as the concatenation of softmax observed labels at all time-steps. For the propose of mapping the path π to the target sequence y, CTC defines a many-to-one mapping operation β. β firstly collapses repeating tokens, then removes all blank symbols from the sequence processed in the previous step. For example, a sequence of 9 labels such as "aϕ ppϕ pϕ le", the sequence turns to "apple" after processing the operation β. The conditional probability of a given target sequence is defined as the sum of probabilities of all feasible paths through β, which is the definition of CTC loss function. The formula is defined as follows:

$$L_{CTC} = -\ln P(\mathbf{y}|\mathbf{x}) = -\ln \sum_{\pi \in B^{-1}(\mathbf{y})} P(\pi|\mathbf{x}) \tag{2}$$

In order to calculate the $P(\pi|x)$, CTC makes an assumption that every time-step output is conditionally independent of previous outputs, the probability of π is decomposed into the product of posteriors from each frame as:

$$P(\pi|\mathbf{x}) = \prod_{t=1}^{T} P\left(\pi_t|\mathbf{x}\right) \tag{3}$$

CTC uses a dynamic programming algorithm to efficiently sum up all the feasible paths. Therefore, the CTC technique provides practicable guidelines for training the end-to-end model by directly optimizing the CTC loss function.

2.4 Decoding Approaches

During inference, we use the decoding algorithms to compute the final label sequence from the output probabilities of CTC. In this work, we use two decoding approaches: greedy decoding and beam search decoding [8].

The idea of greedy decoding method is simple, assuming that the probability between time steps is conditionally independent. Therefore, at each time step, the maximum probability output value is selected as the prediction result. Due to the many-to-one mapping operation β, the result of greedy decoding method is an approximate solution, but it is very fast, which reduces the exponential time complexity to the length-dependent linear time complexity.

In this work, we also use beam search decoding algorithm, which is one of the popular decoding techniques for CTC decoding. Actually the beam search decoding is an improvement on greedy decoding. Beam search is a heuristic search algorithm that explores a breadth-first search tree by expanding the most promising node in limited sets. When the beam width is equal to 1, beam search decoding degrades to greedy decoding. The greater the beam width, the fewer states of search tree are pruned. Therefore, we can improve the performance of beam search decoding by increasing the beam width, but the side effects are increasing decoding time and memory consumption. In practice, it usually needs to find a trade-off between the beam width and time consumption. Due to the beam search sacrifices completeness by pruning, the solution of beam search is also not optimal.

3 Experimental Setup

3.1 Dataset

Our experiments are conducted on self-build Chinese strokes dataset and AISHELL-1 [3] open source mandarin speech corpus. Based on our observation, this is no publicly available simplified Chinese strokes dataset that has 32 different types of strokes, so we create the dataset by crawling webpage and manual correction.

The simplified Chinese strokes dataset contains 21964 Chinese characters and the stroke order of the corresponding Chinese characters. In addition, we find some interesting things that the length of the stroke order of character conforms to the normal distribution, and the lengths of most Chinese characters are distributed between 20–30.

We use the AISHELL-1 corpus dataset which comprises approximately 178 hours of recording from 400 speakers of different accent areas in China. All utterances of AISHELL-1 are encoded at the 16 kHz sampling rate with 16-bit resolution. All experiments use spectrogram of power as input features, computed with a 20ms window and shifted every 10ms. To reduce frequency leakage, we use a Hamming window function to reduce frequency leakage during the speech signal pre-processing.

3.2 Evaluation Metric

Since there is no standard metric for evaluating the stroke-based model, in order to compute the similarity between label strokes order sequence and prediction sequence, we propose the Stroke Error Rate (SER) which is inspired by the Word Error Rate (WER) of measuring the performance of Automatic Speech Recognition (ASR). A standard definition of the SER is as follows:

$$SER = \frac{I_s + S_s + D_s}{N_s} \tag{4}$$

where N_s is the number of strokes in the reference stroke order text, S_s is the number of strokes substituted, D_s is the number of strokes deleted and I_s is the number of strokes inserted required to transform the output text into the target. Fortunately, the numerator of SER is computed with the minimum number of operations required to convert the output prediction text into reference, which is known as Levenshtein distance [11]. The larger the number, the more different between the two stroke text.

3.3 Implementation Details

We reconstruct the speech dataset by replacing the label text with stroke sequences using the method of Sect. 2.1, the reconstructed dataset is used for all of our experiments in this work.

The architecture of convolutional layers of speech encoder is the same for all training and testing experiments, the convolutional block consists of a convolutional layer, batch normalization(BN) [10] layer and Hardtanh activation function layer, the hyperparameters of the first convolutional layer are as follows: the spectrogram is treated as one channel picture by convolutional operations, the number of out channels is 32, size of filters is 41×11 with a stride of 2×2, and the size of padding is 20×5. Similarly, the hyperparameters of the second layer are as follows: the numbers of input channels and output channels are 32, size of filters is 21×11 with a stride of 2×1, and the size of padding is 10×5. In the Bi-RNN blocks, we test different types of recurrent networks, including common RNN, GRU, and LSTM. In addition, we explore the impact of different layers on model performance. Each layer of Bi-RNN blocks has 800 cells. Batch normalization is also applied on each RNN layer. The last one fully connected layer has 34 hidden units, which represents the number of labels in the training set.

All the models are implemented with the PyTorch library [14], we use stochastic gradient descent (SGD) with a momentum of 0.9 [17] to train our model with CTC loss function described in Sect. 2.3, and set the batch size to 32 due to the GPU memory limitations. The initial learning rate is at 3e–4, The learning rate during training adopts 1/t annealing strategy after finishing every epoch, where t is set at 1.1. To accelerate the online learning, we train our model with SortaGrad [2] learning strategy. we also apply early stopping [5] to halt the training of model when the performance is degraded. During inference, we evaluate the test set using the checkpoint that produced the best SER on validation set.

4 Results

4.1 Evaluation of Various Architectures

Table 2 gives the SER obtained with our model using different types of recurrent networks in the Bi-RNN block. All results are obtained by beam search decoding method with a fixed beam size of 32. The convolutional layers and fully connected layers are fixed in these experiments. This is because CNN is used to extract the features of spectrogram, and the last layer, a fully connected network, is used for dimensional adaptation. The core of model is the Bi-RNN blocks, which model the temporal connection. As shown in Table 2, experiment E2 indicates that the GRU-based model has better performance than other RNN types, and we notice that there is an extremely big gap between E0 and E1, E2. it is revealed that the RNN-based model cannot handle the long time dependence, then the problem of vanishing gradient occurs, which makes it impossible to update the weights and ultimately leads to poor generalization.

Table 2. SER of Speech2Stroke with various RNN types

Exp ID	Architecture	SER %
E0	8-layer, 5 Bi-RNN	52.915
E1	8-layer, 5 Bi-LSTM	26.183
E2	8-layer, 5 Bi-GRU	20.609

Moreover, Table 3 gives the SER obtained with our model using different layers of recurrent networks in the Bi-RNN block. As Table 2 indicates GRU-based model performs best, we fix the RNN type as GRU and change the number of layers. The results of Table 3 show that experiment D1 has the lowest SER, when increasing the number of layers from 5 to 7, the performance of D2 is degraded, which seems to be a symptom of overfitting.

Table 3. SER of Speech2Stroke with various RNN layers

Exp ID	Architecture	SER %
D0	6-layer, 3 Bi-RNN	27.456
D1	8-layer, 5 Bi-LSTM	20.609
D2	10-layer, 7 Bi-GRU	22.216

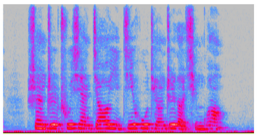
4.2 Evaluation of Various Decoding Methods

Table 4 shows the effects of the decoding beam width and the decoding method on the SER for the test set. The architecture of experiment D1 is applied to this evaluation. As we see, experiment C0 with SER of 20.857%, which natively chooses the most likely label at each time step, as expected, beam search decoding method gives better results than greedy decoding. On the other hand, greedy decoding is much faster than any beam search decoding, 6.4 times faster than, C1, the fastest beam search based experiment. We can observe extra benefits when further increasing the beam width. But the price is increasing time consumption. In practice, we need to find a balance between them in different scenarios.

Table 4. SER and decoding time consuming of Speech2Stroke with various decoding methods. BS-32 denotes beam search with width of 32

Exp ID	Decoding method	SER %	Time (ms/item)
C0	Greedy Decoding	20.857	17
C1	BS-32	20.609	109
C2	BS-128	20.573	458
C3	BS-512	20.547	4677
C4	BS-1024	20.549	8308

Feature input :



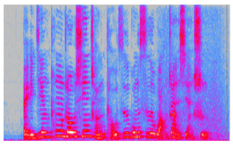
Words label : 北京通州新城投资公司网站显示
（Beijing Tongzhou Xincheng Investment Company website display）

Prediction strokes: " ㇑ ㇐ ㇀ ㇒ ㇟ ㇔ ㇐ ㇑ ㇕ ㇐ ㇚ ㇒ ㇔ ㇔ ㇑ ㇇ ㇐ ㇑ ㇔ ㇇ ㇔ ㇑ ㇆ ㇐ ㇐ ㇑ ㇔ ㇋ ㇏ ㇔ ㇒ ㇔ ㇑ ㇔ ㇑ ㇔ ㇐ ㇔ ㇒ ㇐ ㇐ ㇚ ㇒ ㇔ ㇒ ㇒ ㇐ ㇑ ㇐ ㇐ ㇒ ㇏ ㇐ ㇚ ㇀ ㇒ ㇅ ㇇ ㇏ ㇔ ㇀ ㇒ ㇇ ㇒ ㇏ ㇑ ㇆ ㇒ ㇔ ㇒ ㇏ ㇜ ㇔ ㇕ ㇐ ㇑ ㇕ ㇐ ㇑ ㇆ ㇒ ㇔ ㇒ ㇔ ㇔ ㇑ ㇐ ㇑ ㇑ ㇕ ㇐ ㇐ ㇑ ㇑ ㇔ ㇒ ㇐ ㇐ ㇐ ㇚ ㇒ ㇔ "

Manual stroke decompilation: 北京卞通州新天投资公司网什显示

Fig. 5. Qualitative results of case 1

Feature input :



Words label : 甚至出现交易几乎停滞的情况
（There is even a situation where the transaction is almost stalled）

Prediction strokes: " ㇐ ㇑ ㇑ ㇐ ㇐ ㇐ ㇒ ㇔ ㇗ ㇐ ㇜ ㇔ ㇐ ㇑ ㇐ ㇑ ㇑ ㇗ ㇑ ㇐ ㇐ ㇑ ㇀ ㇑ ㇕ ㇒ ㇟ ㇔ ㇐ ㇐ ㇒ ㇔ ㇒ ㇏ ㇕ ㇐ ㇐ ㇒ ㇆ ㇒ ㇒ ㇈ ㇒ ㇔ ㇐ ㇒ ㇒ ㇚ ㇒ ㇐ ㇐ ㇑ ㇕ ㇐ ㇔ ㇐ ㇏ ㇒ ㇐ ㇔ ㇏ ㇒ ㇑ ㇕ ㇐ ㇐ ㇒ ㇆ ㇔ ㇔ ㇔ ㇑ ㇐ ㇐ ㇑ ㇐ ㇑ ㇆ ㇐ ㇐ ㇔ ㇑ ㇟"

Manual stroke decompilation: 甚至出现交易乙乎适天的情也

Fig. 6. Qualitative results of case 2

4.3 Qualitative Analysis

To better comprehend the quality of the connection between speech and strokes, we conduct qualitative analysis by illustrating some case studies of decoding results. As shown in Figs. 5 and 6, we randomly select two examples in the test set and input the features into Speech2Stroke model, then the model outputs the sequence of strokes. Due to it is complex to figure out the difference between strokes, we manual decompile the strokes to characters. The results of Figs. 5 and 6 show that our model can capture the connection between strokes and speech. Another interesting observation is that words can be determined without an additional language model, which is an advantage of exploiting the internal structural information. As an example of Fig. 5, "投资" (investment) is a pictophonetic word, whose internal structures can indicate the pronunciation and context. Therefore, it can determine the "公司" (company), this is because investment is semantically related to company.

5 Conclusion

In this paper, we propose a novel study of generating strokes of Chinese characters from the audio of speech. The experiments have proved that the internal structures of Chinese characters are strongly connected to speech. We believe that generating strokes gives a more comprehensive view of speech-strokes correlation and can exploit the new tasks with internal structure in Chinese speech research. In the future, we plan to explore the pix-based internal structural information from speech.

References

1. Standardization of stroke order for modern Chinese homepage (2005). http://www.moe.gov.cn/s78/A19/yxs_left/moe_810/s230/201001/t20100115_75615.html, Accessed 2 May 2020
2. Amodei, D., et al.: Deep speech 2: end-to-end speech recognition in English and Mandarin. In: International Conference on Machine Learning, pp. 173–182 (2016)
3. Bu, H., Du, J., Na, X., Wu, B., Zheng, H.: Aishell-1: an open-source mandarin speech corpus and a speech recognition baseline. In: 2017 20th Conference of the Oriental Chapter of the International Coordinating Committee on Speech Databases and Speech I/O Systems and Assessment (O-COCOSDA), pp. 1–5. IEEE (2017)
4. Cao, S., Lu, W., Zhou, J., Li, X.: cw2vec: learning Chinese word embeddings with stroke n-gram information. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
5. Caruana, R., Lawrence, S., Giles, C.L.: Overfitting in neural nets: backpropagation, conjugate gradient, and early stopping. In: Advances in Neural Information Processing Systems, pp. 402–408 (2001)
6. Cho, K., et al.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078 (2014)
7. Graves, A., Fernández, S., Gomez, F., Schmidhuber, J.: Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In: Proceedings of the 23rd International Conference on Machine Learning, pp. 369–376 (2006)
8. Hannun, A.Y., Maas, A.L., Jurafsky, D., Ng, A.Y.: First-pass large vocabulary continuous speech recognition using bi-directional recurrent DNNs. arXiv preprint arXiv:1408.2873 (2014)
9. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
10. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167 (2015)
11. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions, and reversals. Soviet Phys. Doklady **10**, 707–710 (1966)
12. Meng, Y., Wet al.: Glyce: glyph-vectors for Chinese character representations. In: Advances in Neural Information Processing Systems, pp. 2742–2753 (2019)
13. Chinese. BLS. Macmillan Education UK, London (1999). https://doi.org/10.1007/978-1-349-27306-5_9
14. Paszke, A., et al.: Pytorch: an imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems, pp. 8024–8035 (2019)
15. Sainath, T.N., Mohamed, A.R., Kingsbury, B., Ramabhadran, B.: Deep convolutional neural networks for LVCSR. In: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 8614–8618. IEEE (2013)

16. Su, T.R., Lee, H.Y.: Learning Chinese word representations from glyphs of characters. In: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pp. 264–273 (2017)
17. Sutskever, I., Martens, J., Dahl, G., Hinton, G.: On the importance of initialization and momentum in deep learning. In: International Conference on Machine Learning, pp. 1139–1147 (2013)
18. Werbos, P.J.: Backpropagation through time: what it does and how to do it. Proc. IEEE **78**(10), 1550–1560 (1990)
19. Yin, R., Wang, Q., Li, P., Li, R., Wang, B.: Multi-granularity Chinese word embedding. In: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pp. 981–986 (2016)
20. Yu, J., Jian, X., Xin, H., Song, Y.: Joint embeddings of Chinese words, characters, and fine-grained subcharacter components. In: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pp. 286–291 (2017)