End-to-End Mandarin Recognition based on Convolution Input

WANG Yanzhe , ZHANG LiMin, ZHANG Bingqiang, and Li Zhenyu

Institute of Information Fusion, Naval Aviation University, 264000 Yantai Shandong，China

Abstract. The cross-entropy criterion of mainstream neural network training is to classify and optimize each frame of acoustic data, while the continuous speech recognition uses the sequence-level transcription accuracy as a performance measure. In view of this difference, an end-to-end speech recognition system based on sequence level transcription is constructed in this paper. The model uses convolution neural network to deal with the input features, selects the best network structure, and performs two-dimensional convolution in the time and frequency domains. At the same time, neural network uses batch normalization technology to reduce generalization error and speed up training. Finally, the hyper-parameters in decoding process are optimized to improve the modelling effect. Experimental results show that the system performance is improved a lot，better than mainstream speech recognition systems.

1 Introduction

It is a complex task to construct modern automatic speech recognition system, which is based on strict design process, including input feature, acoustic model, language model and hidden Markov model. The long-term goal of ASR is to be able to deal with people's speech in different environments effectively, which is quite challenging for the traditional GMM-HMM hybrid model. A good acoustic model should be capable of simulating various acoustical changes in speech signals effectively in order to obtain robustness under different speech and environmental conditions.

In recent years, acoustic models based on deep neural networks (DNN) and convolutional neural networks (CNN) have achieved good results in some challenging large vocabulary speech recognition tasks [1][2].

However, DNNs have a disadvantage in that they are completely connected and cannot fully utilize the structural locality in the feature space. Similar to model objects in computer vision, acoustic signals may contain basic two-dimensional feature patterns. CNN can better explain small changes and disturbances in the feature space that may be caused by different speaking styles and environments.

In speech recognition, hybrid neural networks are trained to use cross-entropy criteria [2] as objective functions to classify individual frames of acoustic data, which is very different from the accuracy of transcription at the sequence level. This hybrid approach maybe suboptimal for solving the final task because different modules are trained with different standards so that extra hyperparametric adjustments are maybe required for each training stage.

Graves proposed a combination of deep, Long Short-Term Memory (LSTM) networks [3] and linking Connectionist Temporal Classification (CTC) objective functions [4] in [5], which can directly transcribe audio data into text without the need of inter-mediate voice representation. Moreover, there is no need for any pre-alignment between the input sequence and the target sequence, so end-to-end speech recognition represents the trend of future development.

A lot of work has been done in English speech recognition before, so this paper combines deep bidirectional Long Short-Term Memory (DBLSTM) networks with CTC to establish an end-to-end speech recognition model for Mandarin. Model input uses CNN, and neural networks use batch normalization (BN) technique. According to the phonetic features of Chinese Mandarin, characters are used as modelling units, combined with dictionary and speech models when decoding. Experimental results show that the recognition of Chinese Mandarin in resource-constrained conditions results in a good recognition rate.

2 End-to-end speech recognition model

Applying CNN in ASR can reduce spectrum changes and effectively model the spectral correlation in ASR acoustic features. Applying LSTM can learn the dependence information between long-term sequences. Finally, it can be combined with CTC to build a true end-to-end. The speech recognition model (Fig. 1) replaces the traditional method of relying on the HMM model.

2.1 LSTMP

The speech signal has the characteristics of long-range correlation. Standard feedforward networks typically only handle information in the frame's fixed-length sliding window, while LSTM can encode sequence history as their internal state, and can predict phonemes based on all the speech features observed prior to the current frame.

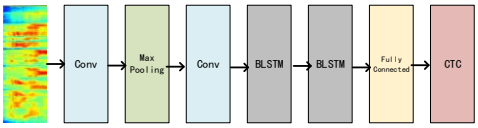

Figure 1. Schematic diagram of network structure

Because speech signal depends heavily on the front and back, the output is usually based on previous and subsequent states. So BLSTM should be used to solve the problem of using the future information of speech.

In order to solve the problem of training a DBLSTM that is slow and difficult to fit. H.Sak [6] combined the matrix decomposition with LSTM to propose a long and short memory network structure with a projection layer. A single projection vector with a lower size linear projection layer than the LSTM output is set behind the LSTM layer.

$$\boldsymbol{i}_t = \sigma(\boldsymbol{W}_{ix}\boldsymbol{x}_t + \boldsymbol{W}_{ip}\boldsymbol{p}_{t-1} + \boldsymbol{W}_{ic}\boldsymbol{c}_{t-1} + \boldsymbol{b}_i) \quad (1)$$

$$\boldsymbol{f}_t = \sigma(\boldsymbol{W}_{fx}\boldsymbol{x}_t + \boldsymbol{W}_{fp}\boldsymbol{p}_{t-1} + \boldsymbol{W}_{fc}\boldsymbol{c}_{t-1} + \boldsymbol{b}_f) \quad (2)$$

$$\boldsymbol{a}_t = g(\boldsymbol{W}_{cx}\boldsymbol{x}_t + \boldsymbol{W}_{cp}\boldsymbol{p}_{t-1} + \boldsymbol{b}_c) \quad (3)$$

$$\boldsymbol{c}_t = \boldsymbol{f}_t\boldsymbol{c}_{t-1} + \boldsymbol{i}_t\boldsymbol{a}_t \quad (4)$$

$$\boldsymbol{o}_t = \sigma(\boldsymbol{W}_{ox}\boldsymbol{x}_t + \boldsymbol{W}_{op}\boldsymbol{p}_{t-1} + \boldsymbol{W}_{oc}\boldsymbol{c}_t + \boldsymbol{b}_0) \quad (5)$$

$$\boldsymbol{p}_t = \boldsymbol{W}_{pm}(\boldsymbol{o}_t\boldsymbol{h}(\boldsymbol{c}_t)) \quad (6)$$

$$\boldsymbol{y}_t = soft\max(\boldsymbol{W}_{yp}\boldsymbol{p}_t + \boldsymbol{b}_y) \quad (7)$$

$\boldsymbol{i}$, $\boldsymbol{f}$, $\boldsymbol{o}$, $\boldsymbol{a}$ and $\boldsymbol{c}$ represents the input gate, the forget gate, the output gate, the unit input activation and the unit state vector, respectively. $\boldsymbol{W}$ and $\boldsymbol{b}$ represents the weight matrix and the offset vector, respectively. $\boldsymbol{W}_{ic}$, $\boldsymbol{W}_{fc}$ and $\boldsymbol{W}_{oc}$ is the diagonal weight matrix for each door and peephole connection.

2.2 CNN

Recently, CNN has been introduced into ASR. Because of its powerful modelling performance of sound signal, it has better results than DNN in many tasks [2].

CNN usually consists of one or more sets of convolutional layers and sampling layers. Each convolutional layer has a number of convolutional filters, which are convolved with lower activations to learn different local features in the speech signal. Then the maximum sampling method (Max-pooling) is generally used to down-sample the output activation at a fixed length, thus reducing the parameters and retaining most of the feature information to improve the performance.

In computer vision, when the convolutional layer convolves each image, the filters' weights are all shared. However, for speech signals, the local structures appearing in different frequency bands may behave in completely different ways, so the idea of limited-weight sharing of convolutional layers is introduced into ASR.

2.2.1 Full Weight Sharing

Assuming the input of Neural Network is $\boldsymbol{X} \in \boldsymbol{R}^{A\times B}$, where A is the number of features representing an input frequency band and B is the number of the input frequency bands. Let's assume that $\boldsymbol{x} = [\boldsymbol{x}_1\ \boldsymbol{x}_2 \ldots \boldsymbol{x}_B]$, where $\boldsymbol{x}_b$ is the feature vector representing band b. The activations of the convolution layer can be computed as:

$$h_{j,k} = \theta(\sum_{b=1}^{s} \boldsymbol{w}_{b,j}^T \boldsymbol{x}_{b+k-1} + a_j) \quad (8)$$

where $h_{j,k}$ is the convolution layer's output of the *jth* feature map on the *kth* convolution layer band, *s* is the filter size, $\boldsymbol{w}_{b,j}$ is a weight vector representing the *bth* band of the *jth* filter, a_j is the bias of the *jth* feature map. After the input and filters are weighted average, the output node value of the network is calculated from the nonlinear function θ.

The pooling function, which computes some statistics of the activations, is typically applied to the neurons along a window of frequency bands and generated from the same feature map in the convolution layer. Max pooling function simply computes the maximum value of the feature over the corresponding frequency bands. The max-pooling activations can be computed as :

$$p_{j,m} = \max_{k=1}^{r}(h_{j,(m-1)\times n+k}) \quad (9)$$

where $p_{j,m}$ is the pooling layer's output of the *jth* feature map and *mth* pooling layer band, *n* is the sub sampling factor, and *r* is the pooling size, which is the number of bands to be pooled together.

2.2.1 Limited Weight Sharing

Unlike FWS, LWS uses multiple different filters to convolve different frequency windows of the speech signal. The weight for input is not fully shared. The neurons in the pooling layer represent the sampling of specific feature maps obtained by different filters convolutions. It is as if the convolution and pooling layers are divided into many sections, each of which processes only a limited range of the input bands and generates only one output band in the pooling layer. Convolution and pooling are calculated as follows:

$$h_{j,k}^{(m)} = \theta(\sum_{b=1}^{s} \boldsymbol{w}_{b,j}^{(m)} \boldsymbol{x}_{(m-1)\times n+b+k-1}^T + a_j^{(m)}) \quad (10)$$

$$p_{j,m} = \max_{k=1}^{r}(h_{j,k}^{(m)}) \tag{11}$$

where $h_{j,k}^{(m)}$ is the value of the *kth* band of the *jth* feature map at the *mth* convolution layer section, and $p_{j,m}$ is the value of the *jth* feature of the *mth* pooling layer band.

2.3 Connectionist Temporal Classification

Traditional hybrid neural networks are suboptimal for ASR, because training using different standards at different stages may not be optimal for solving the final task. CTC can directly map speech features to labels without relying on the alignment between audio sequences and label sequences. The output of the softmax layer of RNN is generally used as the input of the CTC. An extra label is set to represent the blank, which is used to estimate the probability of not outputting the label at a specific moment, for a total of *K+1* labels.

Given an input sequence $\boldsymbol{X}$ whose length is T, the probability of the softmax layer outputs a label or blank index k at time t is:

$$P(k \mid t, \boldsymbol{x}) = \frac{\exp(y_t^k)}{\sum_{k'} \exp(y_t^{k'})} \tag{12}$$

$p(\boldsymbol{z}^l \mid \boldsymbol{x})$ is the product of the output probability of each time step:

$$p(\boldsymbol{z}^l \mid \boldsymbol{x}) = \prod_{t=1}^{T} P(k \mid t, \boldsymbol{x}) \tag{13}$$

$\boldsymbol{z}^l$ denotes all possible alignment of $\boldsymbol{x}$ and $\boldsymbol{l}$ in grid coding. Alignment includes label repetition and blank label insertion. There are several kinds of paths corresponding to an audio sequence, so we must first remove the repeated tags and blanks in the path, and map these paths to transcription. The label sequence $\boldsymbol{l}$ is represented by a set of all possible CTC paths mapped to $\boldsymbol{l}$, and the probability is the sum of all path probabilities:

$$p(\boldsymbol{l} \mid \boldsymbol{x}) = \sum_{p \in \phi(\boldsymbol{l})} p(\boldsymbol{z} \mid \boldsymbol{x}) \tag{14}$$

$\phi(\boldsymbol{l})$ is a many-to-one map representing the set of CTC paths corresponding to $\boldsymbol{l}$ because multiple CTC paths can correspond to the same tag sequence. And $p(\boldsymbol{l} \mid \boldsymbol{x})$ can be obtained by the forward backward algorithm.

The loss function of CTC is defined as the sum of the negative logarithmic probability marked by each training sample correctly, and the function is differentiable, so the CTC network can be trained by back-propagation algorithm:

$$\mathrm{L}_{CTC} = -\sum_{(\boldsymbol{x},\boldsymbol{l})} \ln p(\boldsymbol{z}^l \mid \boldsymbol{x}) = -\sum_{(\boldsymbol{x},\boldsymbol{l})} \mathrm{L}(\boldsymbol{x}, \boldsymbol{z}^l) \tag{15}$$

There are two main differences between CTC and the traditional label framework: 1) If the output is uncertain, additional blank label can be added to reduce the number of labels predicted by the network in one frame; 2) The training standard optimizes the logarithmic probability of the state sequence, rather than the input logarithmic likelihood.

3 Experimental Setup

The experiment uses the open source Mandarin voice corpus AISHELL-1 [7]. The corpus uses 3 recording devices for recording, and then selects high-fidelity microphone audio data and resamples to 16 kHz, 16 bits. In the corpus, the training set is 150 hours in total 120098 sentences, the test set is 5 hours and 7176 sentences, while the verification set is 10 hours in 14326 sentences which is used to cross-test the effect of training. The experiment used a character-level 5-gram language model with a size of 2.8GB, which was trained by Baidu's internally selected text collection [8].

The speech feature vector is generated by a Fourier-transform based filter-banks, which includes 40 coefficients distributed on a mel-scale and energy, along with their first and second temporal derivatives. The long-term characteristics of multi-frame series can effectively improve the performance of the model. Therefore, the long-term characteristics of a total of 11 frames consisting of 5 frames before and after the current frame are taken. All speech data are normalized by averaging over all training cases so that each coefficient or first derivative or second derivative all has zero mean and unit variance. According to the characteristics of Chinese character set, the network output is about 6,000 characters. Because of the instability of CTC training in the early stage, the probability of gradient anomaly in some long sentences is relatively high. The SortaGrad technique is used in the training. The initial values of the model parameters of the neural network are randomly selected from the uniform distribution range of [-0.1,0.1], and the initial learning rate is set to 5e-4.

In this paper, we use PReLU function. The Parametric Rectifier Linear Unit (PReLU) [9] is an extension of the ReLU in which the output of the model in the regions that input is negative is a linear function of the input with a slope of α. PReLU is formalized as:

$$\overset{\sqcap}{\boldsymbol{H}}_i = \begin{cases} \boldsymbol{H}_i, & if\ \boldsymbol{H}_i > 0 \\ \alpha \boldsymbol{H}_i, & otherwise \end{cases} \tag{16}$$

The extra parameter α is usually initialized to 0.1 and can be trained using backpropagation.

In applications like speech recognition, we usually have access to the entire sequences. However, those sequences may have variable length. To solve this problem, we apply a sequence-wise normalization, where we compute the mean and variance of each feature along both the time and batch axis using:

$$\bar{x}_k = \frac{1}{n}\sum_{i=1}^{m}\sum_{t=1}^{T} x_{i,t,k} \tag{17}$$

$$\sigma^2{}_k = \frac{1}{n}\sum_{i=1}^{m}\sum_{t=1}^{T}(x_{i,t,k} - \bar{x}_k)^2 \tag{18}$$

BN introduces additional learnable parameters γ and β for scaling and moving data respectively:

$$BN(x_k) = \gamma_k \hat{x}_k + \beta_k \tag{19}$$

And the batch data processed by BN is used as a new input.

4 Experiment Result

The machine used for the experiment is an Intel Xeon E5-2640 v4 CPU, a Nvidia Tesla M40 GPU, and 128 GB of memory. Each set of experiments was run 3 times and the average was taken as the final experimental result.

4.1 Different LSTM Input

In previous experiments, CTC can reduce the length of input and output sequences at a lower frame rate, reduce the computational cost during decoding and provide an improvement in the delay. Specifically, three 10ms frames are directly stacked into the LSTM. After processing a stack frame once, 3 frames can be skipped. At the same time, the acoustic score evaluation during decoding occurs once every 30ms to speed up the process.

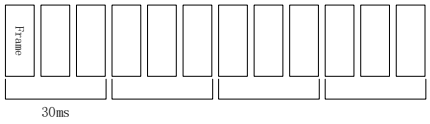

Figure 2. Stacking of input frames

The input of this model is first processed by CNN, each input is a single frame of 20ms, and then input to LSTM. In ASR, the signal characteristics in the low frequency region are very different from those in the high frequency region. Therefore, different spectral bands should use different filter weights so that it allows LFS to be used in convolution layers which weights span only a small local region of frequency. But in theory, given enough feature maps, FWS should be able to match the performance of LWS assuming the model is able learn which convolution feature maps to use and discard in the appropriate frequency bands.

So, this article applies FWS, uses one-dimensional and two-dimensional convolutions, respectively, where 1D is the convolution of the time axis and 2D is the convolution of the time and frequency axes. The stride of both is 1 and 2x1. The pooling layer is only added after the first convolution layer, only sampling the frequency, and its size is 1x3, while maximum sampling is used and does not overlap.

Results as shown in Table 1, there are three layers of LSTM with 1024 nodes in each layer and 3,386 nodes in the full connection layer. The recognition rate is word error rate (WER).

Table 1. Performance of different LSTM inputs

Input	Filters	Filter Size	WER	
			No BN	BN
Stack Frames	-	-	28.12 ± 0.02	25.59 ± 0.09
2 Layers-1D	512,512	5,5	24.53 ± 0.02	22.33 ± 0.07
2 Layers-2D	128,128	41x11 21x11	22.71 ± 0.11	20.68 ± 0.05

From Table 2, it can be seen that the two-dimensional convolution performance is better than one-dimensional convolution, and the recognition rate is improved by about 5.5% on average, and the final recognition result is improved. In low-resource corpus, limited data makes easier to under-fit and tends to favour common phoneme states with large probabilities, whereas two-dimensional convolutions can provide more efficient regularization. And the BN transformation reduces internal covariate shift by insulating a given layer from potentially uninteresting changes in the mean and variance of the layer's input which improves the final performance.

4.2 Pitch Features for Tonal Languages

In particular, the pitch features have been found to be beneficial for tonal languages (e.g., Mandarin, Cantonese and Vietnamese) [10]. On our Mandarin setup, we incorporate the pitch features into CTC model training. The pitch features are extracted using the method described in [10]. On each frame, appending the 3-dimensional pitch to the 40-dimensional filterbank features gives us a 43-dimensional feature vector. The experimental results in Table 2 show that the addition of tone features improves the performance of the system.

Table 2. Performance of adding pitch features

Model	Feature	WER
2 Layers-2D	filterbank	22.33 ± 0.05
	filterbank+pitch	21.66 ± 0.06

4.3 Decode

After the network trained by CTC, it needs an algorithm to decode it. The initial greedy search method which used above does not need to add any language information, selecting the maximum probability output of each time node to search for the best path.

In the process of CTC training, the acoustic model itself has the properties of the language model, and the decoding effect is good without adding the external language information, but the external language model trained with large text corpus is necessary to obtain the

best results. In the decoding process, the beam search is used to find the optimal transcription y that formula 20 can obtain maximum.

$$Q(y)=\log(p_{ctc}(y|x))+\alpha\log(p_{lm}(y))+\beta word\_count(y) \quad (19)$$

The weight α controls the relative contributions of the language model and the CTC network. The weight β encourages more words in the transcription. These parameters are tuned on a development set.

Table 3. Decoding Performance for 2 Layers-2D

Method	LM	WER
Greedy Search	-	21.66 ± 0.06
Beam Search	5-gram LM	15.99 ± 0.08

From Table 3, we can see that the performance of the original greedy search method is obviously worse than beam search, and the recognition rate of beam search method with LM is increased by about 30%. The reason is that greedy search doesn't add any language information to create transcriptions. When in a slightly more complex situation, it is prone to disorder of words.

In formula 21, the optimization of the two hyper-parameters has a significant effect on the final decoding effect, so a two-dimensional grid search is performed to adjust the optimal value based on the validation set.

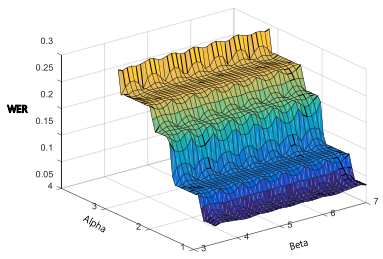

Figure 3. The 2D grid search for hyper-parameter

The grid search diagram for α and β is shown in figure 3. For LM, the global optimal value of WER is α=1.2、β=3.5, then decode the test set again.

Table 4. Decoding Performance for 2 Layers-2D

State	α , β	WER
Initial	2.6，5.0	15.99 ± 0.08
Tune	1.2，3.5	14.57 ± 0.03

The optimized α , β achieves better performance than the initial value on the test set, and the recognition rate is increased by 8.9%. It shows that the optimized super parameters can distribute weights more effectively and rationally, which is helpful to improve the performance of ASR.

Table 5. The Performance of Models

Model	WER
GMM-HMMs	21.73 ± 0.11
DNN-HMMs	14.82 ± 0.03
CNN-HMMs	14.16 ± 0.06

4 Conclusion

In this paper, an end-to-end speech recognition model for Mandarin is built. The input feature of the model uses convolution neural network instead of direct input and it has been modified and adjusted. We use the batch normalization and carry out two-dimensional convolution of input features in time domain and frequency domain. Finally, the effect of super-parameter optimization on the recognition effect is verified, improving the robustness of the model and achieves good results better than the mainstream hybrid ASR (Table 5).

References

1. Hinton, G., Deng, L., Yu, D., Dahl, G. E., ... & Kingsbury, B, *IEEE Signal Processing Magazine*, **29(6)**, 82-97, (2012)
2. Abdel-Hamid, O., Deng, L., & Yu, D, *Interspeech*, **Vol. 2013**, pp. 1173-5, (2013)
3. Hochreiter S, Schmidhuber J, *Neural computation*, **9(8)**, 1735-1780, (1997)
4. Graves A, Fernández S, Gomez F, *Proceedings of the 23rd international conference on Machine learning*, ACM, 369-376, (2006)
5. Graves A, Jaitly N, *International Conference on Machine Learning*, 1764-1772, (2014)
6. Sak H, Senior A, Beaufays F, *Computer Science*, 338-342, (2014)
7. Bu H, Du J, Na X, arXiv preprint arXiv: **1709.05522**, (2017)
8. Amodei D, Ananthanarayanan S, Anubhai R, *International Conference on Machine Learning*, 173-182, (2016)
9. He K, Zhang X, Ren S, *Proceedings of the IEEE international conference on computer vision*, 1026-1034, (2015)
10. Ghahremani P, BabaAli B, Povey D, *Acoustics, Speech and Signal Processing (ICASSP)*, 2494-2498, (2014)