RECURRENT CONVOLUTIONAL NEURAL NETWORK FOR SPEECH PROCESSING

Yue Zhao, Xingyu Jin

Department of Electronic Engineering, TNList,
Tsinghua University, Beijing 100084, China

Xiaolin Hu

Department of Computer Science and Technology,
TNList, Tsinghua University, Beijing 100084, China

ABSTRACT

Different neural networks have exhibited excellent performance on various speech processing tasks, and they usually have specific advantages and disadvantages. We propose to use a recently developed deep learning model, recurrent convolutional neural network (RCNN), for speech processing, which inherits some merits of recurrent neural network (RNN) and convolutional neural network (CNN). The core module can be viewed as a convolutional layer embedded with an RNN, which enables the model to capture both temporal and frequency dependance in the spectrogram of the speech in an efficient way. The model is tested on speech corpus TIMIT for phoneme recognition and IEMOCAP for emotion recognition. Experimental results show that the model is competitive with previous methods in terms of accuracy and efficiency.

1. INTRODUCTION

Speech processing has been studied for decades. It has long been dominated by the Gaussian Mixture Models (GMM) - Hidden Markov Model (HMM) [17] structure until the resurgence of deep neural network (DNN) [20]. The first DNN successfully applied to speech recognition refers to the multi-layer perceptron (MLP) (when trained in an unsupervised way it is called deep belief network [13]). MLP-HMM systems significantly improved the performance of speech recognition on both small datasets [20] and large-scale datasets [7]. In recent years, recurrent neural networks (RNN) such as the long short-term memory (LSTM) and gated recurrent units (GRU) have achieved even better results in speech recognition. However, RNNs are generally hard to train because they cannot take full advantage of current highly optimized parallel computing facilities such as GPU. Convolutional neural network (CNN) is another class of popular deep learning model, but it has not exhibited significant improvement over other models in speech processing.

Recently, Liang et al. [18, 19] proposed an integrated model of RNN and CNN, called Recurrent Convolutional Neural Network (RCNN), and successfully applied it to object recognition and scene labeling. In view of the embedded RNN structure, it is expected to function well in modeling speech because speech is a typical type of sequential data, in which the information is temporally related. This is the primary motivation of the present work. We want to know whether this particular structure is suitable for speech-related applications. The experimental results on two speech processing datasets show that RCNN is efficient and effective, indicating that it is a good alternative in related applications[1].

[1]The source codes can be downloaded at: https://github.com/zhaoyue-zephyrus/RecurrentConvNet-for-Speech.

2. RELATED WORK

CNN has been widely used in computer vision. Intuitively, it is also applicable to speech recognition since the audio signal can be converted via short-time fourier transform (STFT) into a spectrogram which can be viewed as a 2-dimension *image* indexed by the time-axis and frequency-axis. Despite some positive results, it has long been argued that CNNs overkill the variation along time-scale by pooling within a temporal window, resulting in deep fully-connected neural network's dominance in modeling time variation [17]. Abdel-Hamid et al. introduced a *limited-weight-sharing* convolutional scheme [1, 2] and found that using convolution along the frequency axis or time axis increased recognition accuracy, but the improvement was less significant along the time axis. To alleviate the problem, a bottleneck network was constructed in place of the pooling layer [29]. Furthermore, Tóth in [28] proposed treating time-domain and frequency-domain separately and achieved the best performance on the TIMIT dataset by constructing such a hierarchical convolutional network.

Inspired by the temporal characteristics of speech, RNN, which tries to predict the current frame based on feature information collected from previous frames, has long been used in speech recognition tasks [23]. Due to its capability of modeling sequential data, RNN can be combined with HMM, or even replace HMM. In the latter case, the model can be trained "end-to-end", and for this purpose, the connectionist temporal classification (CTC) [9] and RNN Transducer [8] were proposed to deal with the specific evaluation metric for sequence labeling. Two special RNN models, Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) are now widely used [10, 6] in speech recognition. These methods have showed good results in many tasks. One of their limitations refers to the difficulty in training, and in practice, their performance relies heavily on pre-training.

Several recent works attempted to combine CNN and RNN for speech recognition. Amodei et al. proposed a CNN-RNN hybrid model for Large Vocabulary Continuous Speech Recognition (LVCSR) [4]. Sainath et al. proposed an architecture, which unifies CNN, LSTM, and MLP [25]. In the two models, however, the CNN module and RNN module are separated. A similar combination method was proposed for text classification [16]. Recently, Liang et al. proposed a deep learning model in which RNN and CNN were tightly coupled [18, 19]. The hallmark of the model is that there exist intra-layer recurrent connections among units in the convolutional layer of CNN. This model was used in experimentes on static images, but has not been tested on speech data.

3. METHODS

The core module inside RCNN is the Recurrent Convolutional Layer (RCL), whose state evolves over discrete time steps. Recall that a generic RNN usually has a feed-forward input $\mathbf{x}(t)$ and a hidden state $\mathbf{h}(t)$ which depends not only on the input but also the hidden state in the previous time step:

$$\mathbf{h}(t) = \mathcal{F}(\mathbf{x}(t), \mathbf{h}(t-1), \theta)$$

where the function $\mathcal{F}$ describes the dynamic characteristics of the RNN, with parameter θ. In conventional RNN, $\mathcal{F}$ is realized by a fully connected weight matrix and a nonlinear activation function $\sigma(x)$:

$$\mathbf{h}(t) = \sigma(W_{xh}\mathbf{x}(t) + W_{hh}\mathbf{h}(t-1) + b_h).$$

In RCL, the connections are local and share the same weights across the spectrogram, i.e., RCL is realized by convolution. Denote the feedforward input at position (i, j) by $\mathbf{x}^{(t)}$, and the state of the hidden layer to be $\mathbf{h}^{(t)}$, then

$$\mathbf{h}^{(t)}(i,j) = \sigma\Big(\sum_{i'=-s}^{s}\sum_{j'=-s}^{s} \mathbf{w}_k^f(i',j')\mathbf{x}^{(t)}(i-i', i-j') + \sum_{i'=-s}^{s}\sum_{j'=-s}^{s} \mathbf{w}_k^r(i',j')\mathbf{h}^{(t-1)}(i-i', j-j') + b\Big)$$

where $\mathbf{w}_k^f$ and $\mathbf{w}_k^r$ are the k-th feed-forward kernel and recurrent convolutional kernel, respectively. Both kernels are shared at different time steps. $\sigma(x) = f_n(g(x))$ is a composition of two nonlinear functions. The inner one $g(x)$ can be either a conventional sigmoid function $g(x) = 1/(1 + e^{-x})$ or a rectified linear unit (ReLU) [21] $g(x) = \max\{x, 0\}$. A model with ReLU usually converges faster and tends to achieve better performance compared to using the sigmoid function. However, the faster convergence brings the problem of "exploding gradient", which calls for smaller learning rate and necessary normalization. The outer function $f_n(\cdot)$ denotes an appropriate normalization function. The batch-normalization method [14] is adopted here. Specifically, $f_n(x_i; \gamma, \beta) = \gamma\hat{x}_i + \beta$, where γ and β are trainable parameters, and

$$\begin{aligned}\hat{x}_i &= \tfrac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \\ \mu_{\mathcal{B}} &= \tfrac{1}{m}\textstyle\sum_{i=1}^{m} x_i \\ \sigma_{\mathcal{B}}^2 &= \tfrac{1}{m}\textstyle\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2.\end{aligned}$$

In the equations above, x_i denotes the input feature to be normalized, $\hat{x}_i$ denotes the normalized feature, ϵ denotes a small constant (10^{-5} in our experiment), $\mu_{\mathcal{B}}$ denotes the mini-batch mean and $\sigma_{\mathcal{B}}^2$ denotes the mini-batch variance.

In implementation, an RCL is unfolded for T time steps into a multi-layer sub-network. See Fig. 1 for an example with $T = 3$. The receptive field (RF) of each unit expands with larger T , so that more context information is captured by the unit. The depth of the subnetwork increases with larger T, while keeping the number of parameters constant due to weight sharing across time steps.

It is assumed that the input to an RCL $\mathbf{x}(t)$ is the same across time t, which is denoted by $\mathbf{x}_0$. It is equivalently the output of the previous layer. This assumption means that the feed-forward part contributes equally at every time step.

To understand the essence of the RCL, it is useful to clarify the concept of time step in RCL. It is not identical to the time associated with the sequential data, and instead it refers to an *iteration* during processing the data. This is in sharp contrast with conventional RNN, whose time step is identical to the time present in the data. In conventional RNN, the current state is updated according to the previous state, while an RCL processes information from neighboring time slots and frequency banks at each iteration. In this sense, RCNN

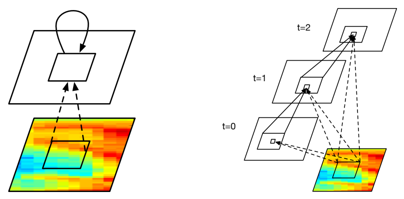

Fig. 1. Illustration of a single RCL (left) and its unfolded version with $T = 3$ (right). The colored parallelograms in the bottom represent the input spectrograms. The upper ones stand for the hidden states. The dotted lines denote feed-forward connections and the solid lines denote recurrent connections.

shares the advantage of multi-dimensional recurrent neural network (MDRNN) [11], that is, modeling on recurrent relationship along all possible dimensions not merely temporal dimension.

By stacking several RCLs, and optionally interleaved with pooling layers and other layers, a deep RCNN can be constructed. This resembles how CNN is constructed based on convolutional layers and other layers. In both computer vision applications and speech recognition applications, it has been found that adding several fully connected layers (i.e., an MLP) on the top will boost the performance of CNN [26, 24]. Inspired by this, we focused here on the RCNN-MLP architecture in experiments.

4. EXPERIMENTS

Two speech processing tasks, phoneme recognition and emotion classification, were considered in our experiments. In phoneme recognition, RCNN was used to predict senones directly. In emotion classification, RCNN is used for feature extraction, while the classification is fulfilled by a support vector machine (SVM). All experiments were carried out on a NVIDIA GeForce GTX Titan Black GPU.

4.1. Phoneme recognition

4.1.1. Dataset

TIMIT recorded 630 speakers, each reading ten phonetically rich sentences. The TIMIT corpus was manually segmented and annotated using 61 phonemes. Excluding the SA sentences that all speakers read, the training and test sets consist of 3696 and 1344 utterances, respectively. 192 utterances among the complete test set are prescribed to be a core test set. We also report our result on a pre-defined development set, a subset of 400 utterances from the test set. 10% (369) utterances were randomly drawn from the training set for automatically adjusting learning rate during training.

4.1.2. Experimental setup

The raw speech data recorded at a sample rate of 16 kHz was first pre-processed via a short-time Fourier Transform with 40 filter-banks distributed on a mel-scale, into 25 ms-long frames, at a stride of 10 ms. The first and second temporal derivatives were also included, which were concatenated to be a 120-dimension feature vector for each frame. The coefficients were then normalized so that they had a mean of 0 and variation of 1 over the training set.

model	layer	kernel size[2]	stride	# of channels	batch norm
CL+pooling	CL	(15,8)	(1,1)	128	-
	pooling	(2,2)	(2,2)	-	-
CL+CL+pooling	CL1	(15,8)	(1,1)	128	no
	CL2	(7,3)	(1,1)	128	no
	pooling	(2,1)	(2,1)		-
RCL(1)+pooling	RCL forward	(12,2)	(4,1)	128	yes
	RCL recurrent	(7,9)	(1,1)	128	yes
	pooling	(4,1)	(4,1)	-	-
RCL(2)+pooling	RCL forward	(11,2)	(1,1)	128	yes
	RCL recurrent	(15,5)	(1,1)	128	yes
	pooling	(15,1)	(15,1)	-	-
RCL(3)+pooling	RCL forward	(11,3)	(1,1)	128	yes
	RCL recurrent	(11,3)	(1,1)	128	yes
	pooling	(15,1)	(15,1)	-	-
RCL(2)+CL	RCL forward	(10,2)	(2,1)	128	yes
	RCL recurrent	(9,5)	(1,1)	128	yes
	CL	(16,2)	(1,1)	256	no

Table 1. Description of the first part of the models before the 3-layer MLP used for TIMIT phoneme recognition

For a frame at time t, a patch ranging from $t-\Delta$ to $t+\Delta$ on the time axis was extracted, which included all filter-bank coefficients on the frequency axis. The spectrogram fed into a model as input had three channels consisting of static coefficients and coefficients of the first and second temporal derivatives, respectively. Therefore, the input patch was of size $(2\times\Delta+1)\times 40\times 3$. In all experiments, $\Delta=5$. The network was trained using the stochastic gradient descent (SGD) with automatic adjusting of the learning rate. The mini-batch size was 200. The initial learning rate was set to be 0.02 per batch and was annealed to half of its original value if the accuracy on the validation set stopped increasing. The momentum was 0.9.

To generate the frame-level label, a conventional context-dependent (CD) HMM of 1954 senones was used with the assistance of the Kaldi toolkit [22]. The phoneme label outputs were mapped to the usual set of 39 labels for evaluation. The ultimate result was based on phoneme error rate (PER).

We tested different models on CNTK [3], all of which used the same 3-layer MLP (each layer had 2048 units) in the end. Then the difference between different models would mainly come from the difference in other layers. Table 1 lists settings of those different layers in different models, where CL denotes convolutional layer. The number in parentheses for each RCL denotes the number of unfolding time steps T. All CLs and RCLs used ReLU while all MLP layers used sigmoid function as activation function. Besides the models described in Table 1, a 4-layer MLP was also tested with 2048 units in every layer.

4.1.3. Results

The results of these models are listed in Table 2. Considering the same 3-layer MLP among the models, the first model in the table can be called MLP, the second and the third can be called CNN, and the models with RCL can be called RCNN. From MLP, CNN to RCNN, a progressive decrease in PER was observed. The comparison between the 1-layer and 2-layer CNNs (the second and the third models in Table 2) confirmed the fact that for small-scale corpus, stacking more convolutional layers may be harmful and convinced us to use a single RCL layer in the RCNN models. We found that unfolding more time steps may yield lower PER (compare the fourth and fifth models in Table 2) but there was a limit. To achieve an appropriate size of

[2]The tuple (N_f, N_t) corresponds to the frequency- and time- axis, and so is the stride size.

	dev set[3]	core test set
4-layer MLP	19.9%	22.0%
CL+pooling+3-layer MLP	18.4%	20.0%
CL1+CL2+pooling+3-layer MLP	19.2%	20.5%
RCL(1)+pooling+3-layer MLP	18.3%	20.3%
RCL(2)+pooling+3-layer MLP	17.3%	19.2%
RCL(3)+pooling+3-layer MLP	17.5%	19.3%
RCL(2)+CL+3-layer MLP	**17.0%**	**18.0%**
DBN [20]	-	20.7%
CNN (limited weight sharing) [1]	-	20.5%
bottleneck CNN [27]	16.1%	18.6%
3-layer LSTM + HMM [30][4]	17.7%	18.8%
3-layer LSTM + pre-trained transducers [10]	-	17.7%
Attention model [6]	15.8%	17.6%
time- and frequency- domain convolution [28]	14.2%	17.6%
time- and frequency- domain convolution (with dropout) [28]	**13.9%**	**16.7%**

Table 2. Results of different models on TIMIT phoneme recognition.

	train	decode
RCNN	2012 samples per second	1.721 utterances per second
LSTM	275 samples per second	0.944 utterances per second

Table 3. Comparison of speed between RCNN and LSTM

input for posterior MLP, a pooling layer of stride 15 was used in most models, which may render a loss of temporal information. By replacing the pooling layer with a CL, PER decreased to 18.0%.

We compared the results with existing models in the literature (see the lower part of Table 2). Our best model outperformed most of the ANN-HMM hybrid models. The exception was a network-in-network configuration [28], which was trained in two steps. Compared with the recently developed RNN-based end-to-end models, such as the RNN transducer [10] and attention model [6], our model was also competitive. However, our model converged faster and did not use pre-training. The best model in [10], which achieved 17.7% PER, was based on a pre-trained transducer and was trained for 144 epochs. For RCNN, however, the model was trained within 20 $\sim$ 30 epochs from scratch.

To compare the speed of RCL module with LSTM module, we trained a 3-layer LSTM with 1024 cells per layer using CNTK on our GPU server. The structure was borrowed from [30] (the non-highway version). An HMM, instead of a pretrained transducer, was used on top of the LSTM. The mini-batch size was set to 200, the same as for RCNN. Results showed that RCNN was faster both in training and decoding phases (Table 3). We attribute the difference to heavily optimized convolution operation in CNTK.

4.2. Emotion recognition

4.2.1. Dataset

The IEMOCAP dataset [5], short for The Interactive Emotional Dyadic Motion Capture, consists of approximately 12 hours of audio-visual data and was annotated by multiple annotators. Since different

[3]The selection for the development set may vary with different authors.

[4]The original paper works on another dataset. We use the structure here only to compare the speed between RCNN and LSTM so the performance may not be tuned perfectly. See in 4.1 for further discussion.

annotators may give different judgments, labels with the majority of annotators were used in order to avoid ambiguity. We only considered utterances with labels from the following five emotions: excitement, frustration, happiness, neutral and surprise. Among the 5300 utterances left after the filtering process, a proportion of 80% was randomly selected for training and the remaining were used for test.

4.2.2. *Experimental setup*

The input to the models consisted of 25-frame segments and the corresponding labels. Three models were constructed. The first model was an MLP with 3 hidden layers. The second was a CNN consisting of a CL with a 2-hidden-layer MLP. The third was an RCNN consisting of an RCL with a 2-hidden-layer MLP. Each fully connected hidden layer had 256 units with the ReLU activation function. For RCL, the feed-forward kernel size (N_f, N_t) was (9, 9) and the recurrent kernel size (N_f, N_t) was (7, 5), and the number of unfolding time steps was 2, such that the RCL unit could see the whole input. The number of channels was 64 for both feed-forward and recurrent part. For comparison, the CL had 128 convolutional kernel with $(N_f, N_t) = (9, 9)$ along with a max pooling layer of size $(2, 2)$ and stride $(2, 2)$ such that the number of learnable weights were comparable. Nonlinearity was realized by ReLU and local response normalization (LRN) [15] with hyper-parameters $\alpha = 10^{-3}, \beta = 0.75, k = 1, n = 9$.

After *segment-level* optimization, the segment-level features were extracted and merged into an *utterance-level* feature for utterance-level classification according to a previous study [12]. Let $\mathbf{f}_s^{(l)} = [f_s^{(l)}(1), \cdots, f_s^{(l)}(D)]$ denote the D-dimensional feature extracted from the l-th layer for the s-th segment. For an utterance with segment $\mathcal{S} = \{1, \cdots, S\}$, the utterance-level feature $\mathbf{f}^{(l)}$ is defined as

$$\mathbf{f}_1^{(l)} = \max_{s\in\mathcal{S}} \mathbf{f}_s^{(l)} \quad \mathbf{f}_2^{(l)} = \min_{s\in\mathcal{S}} \mathbf{f}_s^{(l)} \quad \mathbf{f}_3^{(l)} = \frac{1}{S}\sum_{s=1}^{S} \mathbf{f}_s^{(l)}$$

$$\mathbf{f}_4^{(l)} = \frac{1}{S}\left[\cdots, f_4^{(l)}(i), \cdots\right] = \left[\cdots, \frac{1}{S}\sum_{s=1}^{S} \mathbb{1}\left[f_s^{(l)}(i) \geq \theta\right], \cdots\right]$$

The first three vectors of feature can be viewed as a pooling in the manner of maximizing, minimizing, and averaging along all segments of a single utterance. The last vector is the percentage of segments whose activation on each neuron in the feature map is above a certain threshold.

After the fixed-length utterance-level features were obtained, an SVM classifier was trained to predict the utterance-level labels. Since the utterances were not evenly distributed among emotion categories, both weighed and unweighed accuracy were calculated for evaluation, as in [12]. The weighed accuracy is the accuracy on the whole test set, with every utterance's contribution being the same, while the unweighed accuracy is the averaged accuracy over each emotion class, which better reflects overall accuracy in the presence of an imbalanced class.

$$\text{Weighed Accuracy} = \frac{\text{\# of correct utterances}}{\text{\# of utterances}}$$

$$\text{Unweighed Accuracy} = \frac{1}{5}\sum_{i=1}^{5} \frac{\text{\# of correct utterances for emotion } i}{\text{\# of utterances for emotion } i}$$

4.2.3. *Results*

We compared the classification accuracy using segment-level features from different layers and found that those from the last hidden layer performed best. All results reported in Table 4 are based on those features. Besides weighted accuracy and unweighted accuracy, Table 4 also shows the frame-wise test accuracy during segment-level training. Among the three models, RCNN performed the best in terms of both weighed and unweighed accuracy. Compared with models in the literature, RCNN has achieved competitive results. Note that it is claimed in [12] that using spectral features rendered unsatisfactory performance and, as a result, MFCC plus pitch-based features were used in [12]. Our results indicate that using RCNN, the spectral features, the relatively lower level features, can also achieve good results.

	frame-wise accuracy	weighed accuracy	unweighed accuracy
3-layer MLP	41.4%	48.5%	39.9%
CL+2-layer MLP	43.1%	53.4%	41.6%
RCL+2-layer MLP	43.5%	**53.6%**	42.8%
(MFCC + pitch) MLP+SVM [12]	-	∼ 50%	∼**45%**[5]
Log Spec + CNN [31]	-	-	35.98%
Log Spec + PCA whiten + CNN [31]	-	-	40.02%

Table 4. Speech emotion recognition results on IEMOCAP

5. CONCLUSIONS AND FUTURE WORK

Recently, a deep learning model called recurrent convolutional neural network (RCNN) was proposed for performing computer vision tasks. In this work, we proposed to use this model for speech processing. Experimental results on two benchmark datasets showed that it was competitive with existing models. We conclude that this structure can be used for processing both image and speech information efficiently, which may inspire more generic and efficient cross-modal deep learning models in the future.

Acknowledgement: This work was supported in part by the National Basic Research Program (973 Program) of China under Grant 2013CB329403, in part by the National Natural Science Foundation of China under Grant 61273023, Grant 91420201, Grant 61332007, and Grant 61621136008, and in part by the German Research Foundation(DFG) under Grant TRR-169. Zhao and Jin were also supported by the National Training Program of Innovation and Entrepreneurship for Undergraduates under Grant 201510003036.

6. REFERENCES

[1] O. Abdel-Hamid, L. Deng, and D. Yu. Exploring convolutional neural network structures and optimization techniques for speech recognition. In *Interspeech*, pages 3366–3370, 2013.

[2] O. Abdel-Hamid, A.-r. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu. Convolutional neural networks for speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(10):1533–1545, 2014.

[3] A. Agarwal, E. Akchurin, C. Basoglu, G. Chen, S. Cyphers, J. Droppo, A. Eversole, B. Guenter, M. Hillebrand, R. Hoens, et al. An introduction to computational networks and the computational network toolkit. Technical report, Tech. Rep. MSR-TR-2014-112, August 2014.[Online]. Available: http://research. microsoft. com/apps/pubs/default. aspx, 2014.

[5]The results are read from a bar chart (Figure 3) of [12].

[4] D. Amodei, R. Anubhai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, J. Chen, M. Chrzanowski, A. Coates, G. Diamos, et al. Deep speech 2: End-to-end speech recognition in english and mandarin. *arXiv preprint arXiv:1512.02595*, 2015.

[5] C. Busso, M. Bulut, C.-C. Lee, A. Kazemzadeh, E. Mower, S. Kim, J. N. Chang, S. Lee, and S. S. Narayanan. Iemocap: Interactive emotional dyadic motion capture database. *Language Resources and Evaluation*, 42(4):335–359, 2008.

[6] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio. Attention-based models for speech recognition. In *Advances in neural information processing systems*, pages 577–585, 2015.

[7] G. E. Dahl, D. Yu, L. Deng, and A. Acero. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1):30–42, 2012.

[8] A. Graves. Sequence transduction with recurrent neural networks. *arXiv preprint arXiv:1211.3711*, 2012.

[9] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning (ICML)*, pages 369–376. ACM, 2006.

[10] A. Graves, A.-r. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6645–6649. IEEE, 2013.

[11] A. Graves and J. Schmidhuber. Offline handwriting recognition with multidimensional recurrent neural networks. In *Advances in neural information processing systems*, pages 545–552, 2009.

[12] K. Han, D. Yu, and I. Tashev. Speech emotion recognition using deep neural network and extreme learning machine. In *Interspeech*, pages 223–227, 2014.

[13] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 2006.

[14] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[16] S. Lai, L. Xu, K. Liu, and J. Zhao. Recurrent convolutional neural networks for text classification. In *AAAI*, pages 2267–2273, 2015.

[17] K.-F. Lee and H.-W. Hon. Speaker-independent phone recognition using hidden markov models. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 37(11):1641–1648, 1989.

[18] M. Liang and X. Hu. Recurrent convolutional neural network for object recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3367–3375, 2015.

[19] M. Liang, X. Hu, and B. Zhang. Convolutional neural networks with intra-layer recurrent connections for scene labeling. In *Advances in neural information processing systems (NIPS)*, pages 937–945, 2015.

[20] A.-r. Mohamed, G. E. Dahl, and G. Hinton. Acoustic modeling using deep belief networks. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1):14–22, 2012.

[21] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 23rd international conference on Machine learning (ICML)*, pages 807–814, 2010.

[22] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, et al. The kaldi speech recognition toolkit. In *ASRU*, number EPFL-CONF-192584. IEEE Signal Processing Society, 2011.

[23] A. J. Robinson. An application of recurrent nets to phone probability estimation. *IEEE Transactions on Neural Networks*, 5(2):298–305, 1994.

[24] T. N. Sainath, B. Kingsbury, A.-r. Mohamed, G. E. Dahl, G. Saon, H. Soltau, T. Beran, A. Y. Aravkin, and B. Ramabhadran. Improvements to deep convolutional neural networks for lvcsr. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 315–320. IEEE, 2013.

[25] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak. Convolutional, long short-term memory, fully connected deep neural networks. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4580–4584. IEEE, 2015.

[26] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[27] L. Tóth. Convolutional deep rectifier neural nets for phone recognition. In *Interspeech*, pages 1722–1726, 2013.

[28] L. Tóth. Combining time-and frequency-domain convolution in convolutional neural network-based phone recognition. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 190–194. IEEE, 2014.

[29] K. Veselỳ, M. Karafiát, and F. Grézl. Convolutive bottleneck network features for lvcsr. In *2011 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 42–47. IEEE, 2011.

[30] Y. Zhang, G. Chen, D. Yu, K. Yaco, S. Khudanpur, and J. Glass. Highway long short-term memory rnns for distant speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5755–5759. IEEE, 2016.

[31] W. Zheng, J. Yu, and Y. Zou. An experimental study of speech emotion recognition based on deep convolutional neural networks. In *International Conference on Affective Computing and Intelligent Interaction (ACII)*, pages 827–831. IEEE, 2015.