Gated Time Delay Neural Network for Speech Recognition

Kaibin Chen¹, Weibin Zhang¹,*, Dongpeng Chen², Xiaorong Huang¹, Boji Liu¹ and Xiangmin Xu¹

¹ South China University of Technology, Guangzhou, China
² VoiceAI Technologies, Shenzhen, China

*E-mail: eeweibin@scut.edu.cn

Abstract. In deep neural networks, the gate mechanism is a very effective tool for controlling the information flow. For example, the gates of Long Short-Term Memory (LSTM) help alleviate the gradient vanishing problem. In addition, these gates preserve useful information. We believe that it will benefit if the system learns to explicitly focus on the relevant dimensions of the input. In this paper, we propose Gated Time Delay Neural Networks (Gated TDNN) for speech recognition. Time-delay layers are utilized to model the long temporal context correlation of speech signal while the gate mechanism enables the model to discover the relevant dimensions of the input. Our experimental results on the Switchboard and the Librispeech data sets demonstrate the effectiveness of the proposed method.

1. Introduction
Speech recognition is one of the key ways to achieve natural man-machine interaction. Speech recognition technologies have been used in many interesting products such as Apple's Siri and Amazon's Alexa etc. Almost all the excellent speech recognition systems are based on neural networks. Due to the short-time stationarity and long-time dynamics of speech signals, it is very crucial to use neural networks that can take advantage of the context information and capture the long-term temporal dependencies of speech. A lot of efforts have been spent on improving the temporal modelling capability to improve the system performance.

As famous powerful models to exploit temporal contexts and capture the long-term dependencies, Recurrent Neural Networks (RNNs [1]) and its variants such as Gated Recurrent Unit (GRU [2]), Long Short-Term Memory (LSTM [3], [4]) have been widely used in speech recognition. However, training recurrent neural networks is relatively slow due to the sequential nature of the learning algorithm.

Recently, it was found that Time Delay Neural Networks (TDNNs [5]) with sub-sampling could model the long temporal context of speech equally effectively. In a TDNN, the upper layers deal with information from a wider temporal context so that it can learn wider temporal relationships. We will use TDNN to capture long-term dependencies in this paper.

The gate mechanism can control the information flow in the network and is widely used in recurrent neural networks such as LSTM [3,4], GRU [2] and its variants. In LSTM, input gates control the information that can be entered into the memory cell [3]. The forget gates determine what the memory cell should forget [3]. And the output gates allow the network to control what information should be propagated to the following layers [3]. This gate mechanism provides the network with the
ability to select information, but more importantly, it helps solve the problem of training vanilla RNN due to vanishing gradient [3].

Apparently, the importance of different parts in current input frame is different for the model. We believe it is beneficial to explicitly select those that are more important. Therefore, we propose to add a gate layer after each neural network layer to give weights to different outputs instead of directly feeding them into the next layer. All the weights (i.e. the gate layers) are trained together with other parameters of the neural network to optimize the objective function.

In next Section, we describe relevant work about TDNN and the gate mechanism in recurrent neural networks. In Section 3, the proposed architecture is described in detail. The experimental setups are presented in Section 4, followed by the experimental results and discussions in Section 5. Finally, the conclusion and future work are given in Section 6.

2. Relevant work

2.1. Time delay neural network

TDNN has been proven to be an effective architecture to model the long-term dependencies of speech signal through the hierarchical structure. In a TDNN, the higher the layer is, the wider context it will process.

To reduce the number of parameters, the transforms in the same layer of a TDNN network are tied across time steps. That means there is only a transform in the same layer of a TDNN that will take care of inputs at different time steps. TDNN is regarded as a precursor to the Convolution Neural Network (CNN) [5]. Through carefully designed architecture, TDNN enables the network to discover relevant context information and common acoustic features that are useful in input sequences, rather than just discovering what is currently happening at current frame [5].

However, the hidden activations of a typical TDNN are computed at all time steps. This leads to large overlap at adjacent time frames and thus will introduce a lot of redundant computation. To avoid the problem, sub-sampling is proposed to allow gaps between the frames [5]. Subsampling substantially reduces the overall computation during the forward pass and back propagation, making the training process much faster.

2.2. Gate mechanism in recurrent neural network

Typically, gate mechanisms in recurrent neural networks such as LSTM apply multiplicative gates [3]. For example, the output gate of LSTM can be given by

\[ h_t = o_t \odot \tanh(c_t) \] (1)

where \( h_t \) is the output of the LSTM at time step \( t \). The vectors \( c_t, o_t \) are the activation of the memory cells and output gates, respectively. \( \odot \) is an element-wise multiplication operation. \( o_t \) is the output of a sigmoid function whose input is a linear transform of the input \( x_t \), the output at time step \( t-1 \) (i.e. \( h_{t-1} \)) and sometimes even the memory cells \( c_t \). The input gate and forget gate are similar to output gate.

3. The proposed method

3.1. Gate mechanism in TDNN

We propose to add a gate layer between two TDNN layers. And the proposed gate layer is shown in Figure 1.

We define the input and output of the proposed gate layer as \( x \) and \( y \), respectively. Note that \( x \) can be the output of the previous time delay layer or the input features and the output \( y \) will be the input of the following TDNN layer. Mathematically, we can get the output \( y \) from \( x \) using

\[ y = x \odot a(x) \] (2)
where $\otimes$ is an element-wise multiplication operation that weights each dimension of the input feature and $\alpha(\cdot)$ is given by an attention function. Due to the weights, the networks are forced to explicitly pay attention to the important part of the input. The attention function is given by

$$a(x) = nonL(W \cdot x)$$

where $W$ is the weight matrix of the affine transformation as shown in Figure 1, $\cdot$ is matrices multiplication and $nonL(\cdot)$ is non-linear operations that will be discussed later.

The proposed gate layer can directly tell the following layer in the network that which part of its current input frame is more important by giving them bigger weights. For those part of the input that are useless or even harmful, we would expect the gate attach very small or even zero weights to them, a mechanism similar to the dropout trick [6] except that it's not stochastic.

As to the non-linear operation $nonL(\cdot)$ given in Equation (3), one of the most obvious options is the $sigmoid$ function, because it is always used as the non-linear function of the gates in LSTM or in other network with gate mechanism. However, it didn’t work well in our experiments. Another obvious option is the $softmax$ function. It has been used in many attention models ([7], [8], [9]), which enable the models to pay attention to different parts in time with different weights. The $softmax$ function can amplify the difference of the input. However, we have found that the $softmax$ function always results in bad results in the proposed gate layer. In addition, most of the weights given by the $softmax$ function is almost zero, meaning that the corresponding dimension will be discarded.

Because of the above reason, we prefer to use the $logsoftmax$ function. We found that the $logsoftmax$ function can attach appropriate weights to different inputs, enable the model to pay attention to relatively more important parts. Figure 2 shows an example before and after the weighting. As can been seen, the weighting operation can substantially amplify the difference of the input signal. In our experiments bellow, we also explore other possible non-linear functions.

Figure 1. The proposed gate layer

Figure 2. The first 45 dimension before and after logsoftmax weighting. The gate mechanism has no effect on the part where the output of previous layer (i.e. output of a Relu function) is zero. Obviously, the relative importance of the feature has been amplified.
3.2. Gated TDNN
Figure 3 shows an example of the proposed Gated TDNN. There is a gate layer between two traditional TDNN layers. Note that the weight matrix of the affine transformation in the gate layer has a size of $n \times n$, where $n$ is the dimensionality of the previous layer’s output. Same as TDNN, the gate layer is tied time steps to reduce the number of parameters.

4. Experiments

4.1. Speech data sets
We evaluate the proposed method on the 309 hours Switchboard conversational telephone speech corpora [10] and the Librispeech [11] that contains about 1000 hours of speech data. For Switchboard, we presented the results on the Hub5’00 evaluation set and the “switchboard” subsets. Language model is built from Fisher transcripts. For Librispeech, we showed all results with 4 different language models on the “clean” test dataset and the more challenging noisy “other” dataset. In addition, for the Librispeech task, different speed and volume perturb factors were used to augment the training data about three times.

4.2. Experimental Setup
Experiments are performed with Kaldi Nnet3, the primary neural network framework in Kaldi [12].

In our experiments, two training criterions were used. One is the frame level cross-entropy (CE), the other is the lattice-free MMI (maximum mutual information) [13]. Word error rate (WER) was used as performance metric. In order to get the frame-level ground truth labels, we used an LDA+MLLT+SAT GMM-HMM system [14] to force-align the training sentences. The input feature we used is 40-dimension Mel-frequency cepstral coefficients (MFCCs) extracted every 10ms and the speaker adaptation was applied by appending the features with the 100-dimention iVectors [15].

Before feeding the features into the TDNN, we used linear discriminant analysis (LDA) to give those important dimensions of input data a larger variance [16]. Adjacent features (3 adjacent features in Switchboard and 5 adjacent features in Librispeech) were spliced. The spliced features were then appended with the iVectors to form the final input of the LDA. The LDA matrices were fixed during the training of neural networks.

4.3. Neural network configuration
The configurations of TDNNs we used are similar to those specified in [5], except with slightly different contexts. For these two tasks, details about the layer-wise context configurations are shown in Table 1. We found these configurations lead to better system performance. In the table, {0} means that only the current frame is used. {-3,0,3} means that we splice the input frames at time t-3, t, and t+3. The activation function was Relu [17]. In addition, to ensure the root mean square of the output of
Relu equates to a given target value, we added a re-norm operation after Relu. The output dimension of each layer was fixed at 625 and 1280 for the Switchboard and Librispeech tasks respectively.

### Table 1. The layer-wise context configurations of the TDNN used in the Switchboard and Librispeech tasks.

| Layer | Switchboard context | Librispeech Context |
|-------|----------------------|---------------------|
| 1     | \{0\}               | \{-1,2\}            |
| 2     | \{-1,0,1\}          | \{-3,3\}            |
| 3     | \{-1,0,1\}          | \{-7,2\}            |
| 4     | \{-3,0,3\}          | \{0\}               |
| 5     | \{-3,0,3\}          | -                   |
| 6     | \{-3,0,3\}          | -                   |
| 7     | \{-3,0,3\}          | -                   |

5. Experiments results

We first evaluate the proposed model on the Switchboard data set using the CE training criteria and the results are shown in Table 2. For Gated TDNN, we evaluated the gate layer with different non-linear operations. As can be seen, adding gate layers help improve the recognition accuracy. Compared with the baseline TDNN model, the best Gated TDNN model improves the recognition relatively by 7.09% and 4.28% on the switchboard subset and the full set respectively. In addition, the non-linear function of the gate layer significantly affects the system performance. The logsoftmax function works the best. In the following experiments, we will only use the logsoftmax as the non-linear function of the gate layer.

### Table 2. WER(%) of the baseline TDNN and Gated TDNN on the Switchboard task using CE as the training criteria. The non-linear function of the gate layer is given inside the brackets.

| Models       | Eval/full | Eval/swbd |
|--------------|-----------|-----------|
| TDNN baseline| 18.7      | 12.7      |
| Log-softmax  | **17.9**  | **11.8**  |
| identity     | 18.1      | 12.3      |
| Sigmoid      | 18.6      | 12.3      |
| Tanh         | 18.4      | 12.7      |
| Renorm       | 18.5      | 12.1      |
| Relu+Renorm  | 18.5      | 12.5      |

We then go on to evaluate the Gated TDNN which uses logsoftmax as non-linear function for the gate layers on a larger data set, the “Librispeech” data set. Table 3 shows the results. Again, the Gated TDNNs outperform the baseline TDNN model. In both the “clean” and “other” test datasets, we can get relative improvements from about 3.0% to about 5.5% with different language models.
Table 3. WER(%) of the baseline TDNN and Gated TDNN on the Librispeech task using CE as the training criteria. The non-linear function of the gate layer is logsoftmax.

| Language model | Test dataset | TDNN baseline | Gated TDNN | Relative Improvement(%) |
|----------------|--------------|---------------|------------|-------------------------|
| fglarge\(^a\) | clean        | 5.17          | 4.96       | 4.06                    |
|                | other        | 12.87         | 12.38      | 3.81                    |
| tglarge\(^b\) | clean        | 5.39          | 5.15       | 4.45                    |
|                | other        | 13.36         | 12.95      | 3.07                    |
| tgmed\(^c\)   | clean        | 6.60          | 6.24       | 5.45                    |
|                | other        | 15.98         | 15.07      | 5.69                    |
| tgsmall\(^d\) | clean        | 7.38          | 7.04       | 3.61                    |
|                | other        | 17.44         | 16.70      | 4.24                    |

\(^a\) Non-pruned 4-gram language model.
\(^b\) The full, non-pruned 3-gram language model.
\(^c\) Slightly less pruned 3-gram language model.
\(^d\) The pruned 3-gram language model, which is used for lattice generation.

Finally, we evaluate the above models using the lattice-free MMI training. Table 4 and Table 5 shows the results on Switchboard and Librispeech respectively. Again, improvement over the baseline TDNN is achieved.

Table 4. WER(%) of the baseline TDNN and Gated TDNN using the lattice-free MMI objective function on the Switchboard task.

| Models            | Eval/full | Eval/swbd |
|-------------------|-----------|-----------|
| TDNN baseline     | 16.5      | 11.0      |
| Log-softmax       | 16.0      | 10.5      |

Table 5. WER(%) of the baseline TDNN and Gated TDNN using the lattice-free MMI training in Librispeech task. The non-linear function of the gate layer is logsoftmax.

| Language model | Test dataset | TDNN baseline | Gated TDNN | Relative Improvement(%) |
|----------------|--------------|---------------|------------|-------------------------|
| fglarge\(^a\) | clean        | 4.22          | 4.09       | 3.08                    |
|                | other        | 10.43         | 10.22      | 2.01                    |
| tglarge\(^b\) | clean        | 4.33          | 4.3        | 0.69                    |
|                | other        | 10.92         | 10.63      | 2.66                    |
| tgmed\(^c\)   | clean        | 5.31          | 5.22       | 1.69                    |
|                | other        | 13.33         | 12.86      | 3.53                    |
| tgsmall\(^d\) | clean        | 5.87          | 5.74       | 2.21                    |
|                | other        | 14.45         | 14.1       | 2.42                    |

\(^a\) Non-pruned 4-gram language model.
The full, non-pruned 3-gram language model.

Slightly less pruned 3-gram language model.

The pruned 3-gram language model, which is used for lattice generation.

6. Conclusions

In this paper, we propose to add gate layers in a TDNN model in order to explicitly teach the TDNN layer that which input dimension is more important. The added gate layers are trained together with other model parameters to optimize the final objective function.

We evaluated the proposed architectures on the Switchboard and Librispeech data sets. Both CE and lattice-free MMI were used to train the models. Our experiments show that gate layers with the logsoftmax non-linear operation performs the best. In addition, the proposed Gated TDNN significantly outperform the baseline TDNN models on both the data sets. In the future, we want to use the same idea in a multi-array task to explicitly attach different weights to different channels, an idea similar to beamforming.

Acknowledgments

This work is supported in part by the National Natural Science Founding of China (61601187, U1636218).

References

[1] Alex G, Abdelrahman M and Geoffrey EH 2013 Speech recognition with deep recurrent neural networks ICASSP pp 6645–49
[2] Junyoung C, Caglar G, Ehre, Kyung H and Yoshua B 2014 Empirical evaluation of gated recurrent neural networks on sequence modelling CoRR pp 1735–80
[3] Sepp H and Jurgen S 1997 Long short-term memory Neural computation vol 9 pp 1735–80
[4] Hasim S, Andrew S and Françoise B 2014 Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition Computer science pp. 338–42
[5] Vijayaditya P, Daniel P and Sanjeev K 2015 A time delay neural network architecture for efficient modeling of long temporal contexts Interspeech pp 3214–18
[6] Srivastava N, Hinton G, Krizhevsky A, Sutskever I and Salakhutdinov R 2014 Dropout: a simple way to prevent neural networks from overfitting Journal of machine learning research vol 15 pp 1929–58
[7] Shan C, Zhang J, Wang Y, et al. 2018 Attention-based end-to-end speech recognition on voice search Eprint Arxiv
[8] Chan W, Jaitly N, Le Q and Vinyals O 2016 Listen, attend and spell: A neural network for large vocabulary conversational speech recognition ICASSP pp. 4960–4
[9] Jan C, Dzmitry B, Kyunghyun C and Yoshua B 2014 End-to-end continuous speech recognition using attention-based recurrent nn: First results Eprint Arxiv
[10] John G, Holliman E and Daniel J Switchboard telephone speech corpus for research and development ICASSP vol 1 pp 517–20
[11] Panayotov V, Chen G, Povey D, et al. 2015 Librispeech: An ASR corpus based on public domain audio books ICASSP pp.5206–10
[12] Ghoshal A et al. 2011 The kaldi speech recognition toolkit IEEE 2011Workshop on Automatic Speech Recognition and Understanding
[13] Daniel P et al. 2016 Purely sequence-trained neural networks for asr based on lattice-free mmi Interspeech pp 2751–5
[14] Shakti PR, Daniel P, Karel V and Jan C 2013 Improved feature processing for deep neural networks Interspeech pp 109–13
[15] Dehak N et al. Frontend factor analysis for speaker verification 2011 IEEE Transactions on audio speech and language processing pp:788-98.
[16] Povey D, Zhang X and Khudanpur S 2014 Parallel training of dnns with natural gradient and parameter averaging Eprint Arxiv
[17] Xavier G, Antoine B and Yoshua B 2012 Deep sparse rectifier neural networks AISTATS pp 315–23