Deeper Time Delay Neural Networks for Effective Acoustic Modelling

Xiaorong Huang\textsuperscript{1,b}, Weibin Zhang\textsuperscript{1,a}, Xiangmin Xu\textsuperscript{1}, Ruixiang Yin\textsuperscript{1} and Dongpeng Chen\textsuperscript{2}

\textsuperscript{1}South China University of Technology, Guangzhou, China
\textsuperscript{2}Voiceai Technologies, Shenzhen, China

Email: \textsuperscript{a}eeeweibin@scut.edu.cn, \textsuperscript{b}eexiaorong@mail.scut.edu.cn

Abstract. Time delay neural networks (TDNNs) have been shown to be an efficient network architecture for modelling long temporal contexts in speech recognition. Meanwhile, the training times of TDNNs are much less, compared with other long temporal contexts models based on recurrent neural networks. In this paper, we propose deeper architectures to improve the modelling power of TDNNs. At each TDNN layer that needs spliced input, we increase the number of transforms so that the lower layers can provide more salient features for upper layers. Dropout is found to be an effective way to prevent the model from overfitting once the depth of the model is substantially increased. The proposed architectures significantly improvements the recognition accuracy in Switchboard and AMI.

1. Introductions

Speech recognition is one of the key components for many interesting products such as Apple's Siri, Amazon's Alexa etc. Currently, neural networks have been adopted in almost all state-of-the-art speech recognition systems. Due to the temporal dynamics in speech, it is very important to incorporate neural networks that can capture the long-term temporal dependencies of speech. On the other hand, the modelling power of the neural networks themselves can substantially affect the recognition accuracy of the system. A lot of effort has been spent on these two aspects to improve the system performance.

To exploit temporal contexts and capture the long-term dependencies, Recurrent Neural Networks (RNN) [1] and its variants such as Gated Recurrent Unit (GRU) [2] and Long Short Term Memory (LSTM) [3] have been widely used in speech recognition. However, recurrent neural networks are slower to be trained due to the sequential nature of the learning algorithm [4]. Recently, time delay neural networks with sub-sampling have been proposed for effective modelling of long temporal contexts of speech [4]. In a TDNN, the upper layers deal with information from a wider temporal context and thus can learn wider temporal relationships. We will use TDNN to capture long term dependencies in this paper.

It is well-known that the depth of neural networks is a major determinant of modelling power. Deeper and deeper networks are being developed to achieve state-of-the-art results [5, 6, 7]. Models with hundreds or even thousands of layers such as GoogleNet [8] and ResNet [9] have significantly improve the performance in image recognition tasks.

We are driven by the same motivation that lead to the success of very deep models in computer vision [8, 9].
In this paper, we explore deeper time delay neural networks (TDNNs) for acoustic modelling. It has been shown that TDNNs can effectively capture the long-term dependencies of speech [4, 10]. In addition, they need much less training time during training, compared with recurrent neural networks such as LSTM. Previously, each layer of TDNNs typically contains one linear operation, followed by another non-linear operation [4, 10, 11, 12]. We propose to add more operations to each TDNN layer while keeping the number of parameters manageable. An idea that is similar to stochastic depth neural networks [13] is applied to each TDNN layer to make the model deeper without suffering from poor generalization.

The next section describes relevant work, including LSTM, TDNN, dropout and stochastic depth networks. Then, we described our proposed architecture in detail in Section 3. The experimental setups and results are presented in Section 4, followed by conclusions and future work in Section 5.

2. Relevant work

Recurrent neural networks have been shown to be able to learn the temporal dynamics of speech signal [12]. For example, dynamic contexts information is wisely stored in each LSTM cell state through gate control.

On the other hand, TDNN models the long-term dependencies through a carefully designed hierarchical structure [10]. The initial transforms of a TDNN process information from relatively narrow contexts and the deeper layers process the hidden activations from a wider context. In a typical TDNN, hidden activations are computed at all time steps. Large overlap between the activations computed at adjacent time frames will cause redundant computation. Rather than simply splicing frames contiguously, sub-sampling is proposed to allow gaps between the frames and thus substantially reduces the overall computation [4].

Cheng et. al. [12] recently proposed an architecture by combining TDNN and LSTM together. The new structure (TDNN-LSTM) enables the model to learn longer contexts information. The incorporated frame-level dropout [12] is similar to the dropout scheme used in stochastic depth networks [13].

In this paper, we focus on the sub-sampling TDNN architecture and aim to make it much deeper to improve the modelling power.

3. The proposed neural network architectures

3.1. Time delay neural network

With carefully designed hierarchical architecture, each layer of a TDNN operates at a different temporal resolution and the higher layers can process speech input from a larger time span. Therefore, TDNN is a suitable model to learn the temporal structure of acoustic events and the temporal relationship between such events.

To reduce the number of parameters in a TDNN model, the transforms in the same layer of a TDNN network are tied across time steps. These shifted transforms in the same layer of a TDNN are designed and trained to look for a certain acoustic event. Therefore, TDNN can discover useful acoustic features in the input, regardless of when in time they actually occurred. This translation invariant property, i.e., the network are not sensitive to shift in time, has been verified experimentally in [10]. This parameter-sharing idea is similar to convolutional neural networks (CNNs) and TDNNs are seen as a precursor to CNNs [4].

Another way to reduce the number of parameters and speed up computation is sub-sampling [4]. Instead of splicing together continuous temporal window of frames at each layer, sub-sampling allows gaps between frames, for example, the splicing configuration {-2, 2} means that we splice the input at current time step minus 2 and the current time step plus 2. The authors in [4] found that it is best to splice wider contexts as we go to higher layers of the network. Minimum overlaps between input contexts can be achieved through the carefully designed hierarchical structure. And the sub-sampling process helps reduce the model size and speed up the training significantly.
There are different ways to make the TDNN deeper. The higher part of a TDNN network, where the input contexts are $\{0\}$ (i.e. no spliced input), is actually a simple feed-forward neural network. We can make this part deeper by incorporating techniques such as ResNet [9] and stochastic depth networks [13]. However, we found that it is useless to simply make this part deeper. Therefore, we focus on making the lower part of a TDNN network (i.e. layers that need splicing at the input) deeper in this paper. Two ways will be investigated.

In this paper, all the transformations in the same layer of the lower part (i.e. spliced input) of a TDNN is called a TDNN block. In the architecture of a traditional TDNN network [4, 10, 12], each TDNN block contains only one layer (e.g. an affine transformation followed by another non-linear operation). The ability of these blocks to effectively extract the temporal contexts information is limited.

Instead of using a single layer for each TDNN block, we propose to use a deep neural network. In Figure 1, the first layer of a TDNN block is simply the traditional TDNN layer. In our experiments below, it contains an affined transformation, followed by the ReLu non-linear operation. Usually the dimension of the input of this layer is large due to splicing (e.g. 2048 or even larger). However, the dimension of the output is relatively small, e.g. 1024 in our experiments. Therefore, one of the functions of the first layer is dimension reduction. On top of the first TDNN layer within each TDNN block, we add more layers to make the network deeper.

The first method is to simply add multiple network layers on top the first TDNN layer. Figure 1 shows a deeper TDNN network where each TDNN block has three layers (a TDNN layer plus two DNN layers). We call this model TDNN-DNN below. Compared with the original TDNN networks, the modelling capability in each TDNN block is improved.

3.2. Deeper TDNN

There are different ways to make the TDNN deeper. The higher part of a TDNN network, where the input contexts are $\{0\}$ (i.e. no spliced input), is actually a simple feed-forward neural network. We can make this part deeper by incorporating techniques such as ResNet [9] and stochastic depth networks [13]. However, we found that it is useless to simply make this part deeper. Therefore, we focus on making the lower part of a TDNN network (i.e. layers that need splicing at the input) deeper in this paper. Two ways will be investigated.

In this paper, all the transformations in the same layer of the lower part (i.e. spliced input) of a TDNN is called a TDNN block. In the architecture of a traditional TDNN network [4, 10, 12], each TDNN block contains only one layer (e.g. an affine transformation followed by another non-linear operation). The ability of these blocks to effectively extract the temporal contexts information is limited.

Instead of using a single layer for each TDNN block, we propose to use a deep neural network. In Figure 1, the first layer of a TDNN block is simply the traditional TDNN layer. In our experiments below, it contains an affined transformation, followed by the ReLu non-linear operation. Usually the dimension of the input of this layer is large due to splicing (e.g. 2048 or even larger). However, the dimension of the output is relatively small, e.g. 1024 in our experiments. Therefore, one of the functions of the first layer is dimension reduction. On top of the first TDNN layer within each TDNN block, we add more layers to make the network deeper.

The first method is to simply add multiple network layers on top the first TDNN layer. Figure 1 shows a deeper TDNN network where each TDNN block has three layers (a TDNN layer plus two DNN layers). We call this model TDNN-DNN below. Compared with the original TDNN networks, the modelling capability in each TDNN block is improved.
In the second method, we allow each adding layer to be dropped entirely during training. The corresponding transformation is then bypassed through the skip connection. If all adding layers are skipped, the proposed architecture corresponds to a traditional TDNN. During testing, we keep all the adding layers in order to utilize the full-length network with all its model capacity. This idea comes from stochastic depth neural networks [13]. Therefore, we call this method TDNN-SD below. An TDNN-SD network is shown in Figure 2. The dropout scheme in stochastic depth neural networks [13] is based on mini-batch. However, we found that it is better to use a frame-level dropout scheme in our experiments.

3.3. Dropout

We found that dropout is a key factor to the success of the proposed models since they are easier to overfit. There are mainly two kinds of dropout used in our model.

The traditional dropout, also known as per-element dropout [12], in which the dropout mask for each element is chosen independently, is found to be helpful to all TDNN layers and the DNN layers in the TDNN-DNN architecture.

The dropout schedule proposed in [12] is used. Specifically, the dropout schedule can be expressed as a piecewise linear function on the interval [0,1], where \( f(0) \) and \( f(1) \) give the dropout rate at the start and the end of training respectively. As an example of the notations used below, the schedule \( 0@0; 0.3@0.5; 0@1 \) denotes a function \( f(x) \) that is linearly interpolated between the points of \( f(0)=0 \), \( f(0.5)=0.3 \), and \( f(1)=1 \). Since the first point is always for \( x=0 \) and the last is always for \( x=1 \), this schedule can also be written as \( 0; 0.3@0.5; 1 \). The schedules introduced above always end at zero.

Unlike the per-element dropout, the entire vector in per-frame dropout is set to either zero or one. The per-frame dropout is applied to all SD layers in TDNN-SD models.

4. Experiments

4.1. Experimental setup

For our experiments we used the 309 hours Switchboard conversational telephone speech task [14] and the 100 hours AMI corpus [15] which contains recordings of spontaneous conversations in meeting scenarios. The nnet3 toolkit in Kaldi speech recognition [16] was used to build our experimental systems. For the results on SWBD, we report the WER on the eval2000 test set, which is also known as the complete Hub5 '00 evaluation set and contains 3.72 hours of speech. For AMI, we present results on the IHM and SDM datasets.

4.2. GMM-HMM acoustic models

Alignments for neural network training and the numerator lattices for the lattice-free maximum mutual information (LF-MMI) training are generated from the GMM-HMM systems. Our language model is similar to that described in [17] and the features used in the GMM-HMM systems are LDA+MLLT+SAT features described in [18].

4.3. DNN-HMM acoustic models

The input feature for TDNN network is 40-dimensional Mel-frequency cepstral coefficients (MFCCs) without cepstral truncation. Speaker adaptation was applied by appending the input with the 100-dimensional i-vectors. The i-vectors contained information about the cepstral mean-offset of the speakers’ data and cepstral mean normalization (CMN) for the MFCCs was therefore not necessary. Data augmentation was used to create more training data by applying speech perturbations. Experiments setup in AMI are similar to those described in [19] except that alignments obtained using the IHM SAT GMM-HMM system were used to train the SDM system. The LF-MMI training criterion proposed in [11] was used to train all the models and he chunk size in LF-MMI was fixed to 1.5 seconds to perform sequence training.
Table 1. The layer-wise contexts configuration of the TDNN used in the SWBD and AMI experiments.

| Layer | SWBD Context | AMI Context |
|-------|--------------|-------------|
| 1     | {-1, 0, 1}   | {-1, 0, 1}  |
| 2     | {-1, 0, 1}   | {-1, 0, 1}  |
| 3     | {-1, 0, 1}   | {-1, 0, 1}  |
| 4     | {-3, 0, 3}   | {-3, 0, 3}  |
| 5     | {-3, 0, 3}   | {-3, 0, 3}  |
| 6     | {-3, 0, 3}   | {-3, 0, 3}  |
| 7     | {-3, 0, 3}   | {-3, 0, 3}  |
| 8     | -            | {-3, 0, 3}  |
| 9     | -            | {-3, 0, 3}  |

Table 2. Results of adding DNN layers at the higher part of a TDNN network on SWBD. The baseline system means no DNN layer added.

| Num of adding DNN layers | SWBD      |          |
|--------------------------|-----------|----------|
|                          | TrainDev  | Hub 5'00 |
| Baseline                 | 13.02     | 15.0     |
| 2                        | 12.66     | 14.9     |
| 4                        | 12.77     | 14.8     |
| 6                        | 12.72     | 14.7     |
| 8                        | 12.83     | 14.8     |
| 10                       | 12.55     | 15.0     |

For the baseline system of TDNN, we used a deeper architecture proposed in [12]. Our preliminary experiments showed that the deeper architecture was better than the one used in [4]. Details about the layer-wise contexts configuration are shown in Table 1. We used ReLu as the activation function and set the output dimension of each layer to 1024.

4.4. Different ways to make the TDNN deeper

Table 2 shows the results of adding more DNN layers at the higher part of a TDNN network (i.e. those layers without spliced input). We find it useless to simply make the higher part of a TDNN network deeper. Therefore, in the following experiments, we will focus on deepening the lower part of the TDNN network.

4.5. Results of TDNN-DNN models

The results of TDNN-DNN models is described in Table 3. When the network architecture gets deeper, we need to use a larger dropout rate. As can be seen, adding DNN layers in the TDNN blocks significantly improve the recognition accuracy, indicating that a single TDNN layer in the baseline
Table 3. Results of TDNN-DNN models with various dropout schedules SWBD and AMI. The dropout schedule is '0, 0@0.20, p@0.50, 0'. TDNN-DNN-n means a TDNN-DNN model with n DNN layers.

| Model          | Dropout prob. p | SWBD Train | SWBD Dev | SWBD Hub5'00 | IHM Dev | IHM Eval | IHM Dev | IHM Eval | SDM Dev | SDM Eval |
|----------------|-----------------|------------|----------|--------------|---------|----------|---------|----------|---------|----------|
| Baseline       | None            | 13.02      | 15.0     | 21.4         | 21.5    | 39.1     | 42.9    |          |         |          |
| TDNN-DNN-1     | 0.1             | 12.51      | 14.4     | 21.0         | 21.1    | **38.4** | **41.9**|          |         |          |
| TDNN-DNN-2     | 0.1             | 12.26      | 14.4     | 20.9         | 21.0    | 38.5     | 42.2    |          |         |          |
| TDNN-DNN-3     | 0.1             | 12.43      | 14.6     | **20.8**     | **20.7**| 38.7     | 42.3    |          |         |          |
| TDNN-DNN-1     | 0.3             | 12.26      | 14.3     | 20.8         | 21.1    | 38.7     | 42.9    |          |         |          |
| TDNN-DNN-2     | 0.3             | 12.28      | 14.3     | 21.0         | 21.0    | 39.1     | 43.1    |          |         |          |
| TDNN-DNN-3     | 0.3             | 12.41      | 14.4     | 21.1         | 21.2    | 39.4     | 43.0    |          |         |          |

Table 4. Results of TDNN-SD models on SWBD and AMI. TDNN-SD-n means a TDNN-SD model with n SD layers.

| Model        | Params(M) | SWBD Train | SWBD Dev | SWBD Hub5'00 | IHM Dev | IHM Eval | IHM Dev | IHM Eval | SDM Dev | SDM Eval |
|--------------|-----------|------------|----------|--------------|---------|----------|---------|----------|---------|----------|
| Baseline     | 31.9      | 13.02      | 15.0     | 21.4         | 21.5    | 39.1     | 42.9    |          |         |          |
| TDNN-SD-1    | 39.3      | 12.13      | 14.3     | 20.5         | 20.3    | **37.3** | **41.3**|          |         |          |
| TDNN-SD-2    | 46.6      | 12.05      | 14.2     | **20.2**     | **20.1**| 37.6     | 41.4    |          |         |          |
| TDNN-SD-3    | 53.9      | **12.00**  | **14.1** | 20.2         | 20.4    | 37.9     | 41.5    |          |         |          |

TDNN architecture may not be sufficient to process the temporal contexts and the expansion to TDNN blocks by adding more DNN layers after each TDNN layer is reasonable.

The best TDNN-DNN model outperforms the baseline TDNN model by 4.6% relatively on the SWBD data set. Almost the same conclusions can be drawn on the AMI corpus. The relative WER reduction by using TDNN-DNN models is about 3.7% on IHM and 2.3% on SDM.

Meanwhile, the comparison of TDNN-DNN-1, TDNN-DNN-2 and TDNN-DNN-3 in SWBD shows that adding to much DNN layers may lead to a decrease in performance.

4.6. Results of TDNN-SD models
There are two different dropout schemes for the TDNN-SD models (i.e. the per-element dropout scheme for all layers and the per-frame dropout scheme for the adding SD layers). The per-element dropout scheme is based on the settings that achieve the best results in Table 3. The dropout probability p for the per-frame dropout scheme that is applied to all adding SD layers is set to 0.3 in all of the following experiments.

The results of TDNN-SD models are shown in Table 4. By using the stochastic depth technique, we can train deeper models that achieve improved recognition accuracy. The best TDNN-SD model outperform the baseline model by 6% relatively on SWBD. And the relative WER reduction by using TDNN-SD models is about 6.5% on IHM and 3.7% on SDM.

Compared with the TDNN-DNN models, the TDNN-SD models enable deeper models to be trained, especially on the SWBD corpus where more training data are available.
5. Conclusion
This paper explores ways to improve the modeling capabilities of TDNN models. Different ways were explored to make the TDNN network deeper. We found that it is useless to add more layers to the upper part of the TDNN where no spliced input is required. Thus, we focused on making the lower part (layers that need spliced input) of the TDNN deeper. Two methods (i.e. adding more feed forward layers with/without per-frame dropout) to make the TDNN blocks deeper are explored in this paper. Experiments show that adding more layers in the TDNN blocks significantly improves the modeling capability of the system and the recognition accuracy. In addition, applying the per-frame dropout scheme, an idea that is similar to stochastic depth networks, to the adding layers can effectively alleviate the overfitting problem, allowing deeper and more robust models to be trained. Our best models significantly outperform the baseline models on both the SWBD and AMI datasets. In the future we would like to apply the same idea to other neural network model and use singular value decomposition (SVD) to reduce network parameters.

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] 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
[5] Geoffrey H et al. 2012 Deep neural networks for acoustic modelling in speech recognition: The shared views off our research groups IEEE Signal Processing Magazine vol 29 pp 82–97
[6] Duisheng C, Weibin Z, Xiangmin X and Xiaofeng X 2017 Deep networks with stochastic dept APSIPA pp 1–4
[7] Pegah G, Jasha D and Michael L 2016 Linearly augmented deep neural network ICASSP pp 5085–9
[8] Christian S et al. 2015 Going deeper with convolutions CVPR pp 1–9
[9] Kaiming H, Xiangyu Z, Shaoqing R and Jian S 2016 Deep residual learning for image recognition CVPR pp 770–8
[10] Alex W et al. 1989 Phoneme recognition using time delay neural networks (IEEE) Trans. Acoust. Speech, Signal Process. vol 37 pp 328–9
[11] Daniel P et al. 2016 Purely sequence-trained neural networks for asr based on lattice-free mmi Interspeech pp 2751–5
[12] Gaofeng C et al. 2017 An exploration of dropout with lstms Interspeech pp 1586–90
[13] Gao H, Yu S, Zhuang L, Daniel S and Kilian Q 2016 Deep networks with stochastic depth ECCV pp 646–1
[14] John G, Holliman E and Daniel J Switchboard telephone speech corpus for research and development ICASSP vol 1 pp 517–20
[15] Mccowan I et al. 2005 The ami meeting corpus Noldus Information Technology pp.28–39
[16] Ghoshal A et al. 2011 The kaldi speech recognition toolki IEEE 2011Workshop on Automatic Speech Recognition and Understanding
[17] Karel V, Arnab G, Luk’ as B and Daniel P 2013 Sequence-discriminative training of deep neural networks Interspeech pp 2345–9
[18] Shakti PR, Daniel P, Karel V and Jan C 2013 Improved feature processing for deep neural networks Interspeech pp 109–13
[19] Vijayaditya P et al. 2016 Far-field asr with out parallel data Interspeech pp 1996–2000