IMPROVED TDNNS USING DEEP KERNELS AND FREQUENCY DEPENDENT GRID-RNNS

F. L. Kreyssig, C. Zhang, P. C. Woodland

Cambridge University Engineering Dept., Trumpington St., Cambridge, CB2 1PZ U.K.
{flk24,cz277,pcw}@eng.cam.ac.uk

ABSTRACT

Time delay neural networks (TDNNs) are an effective acoustic model for large vocabulary speech recognition. The strength of the model can be attributed to its ability to effectively model long temporal contexts. However, current TDNN models are relatively shallow, which limits the modelling capability. This paper proposes a method of increasing the network depth by deepening the kernel used in the TDNN temporal convolutions. The best performing kernel consists of three fully connected layers with a residual (ResNet) connection from the output of the first to the output of the third. The addition of spectro-temporal processing as the input to the TDNN in the form of a convolutional neural network (CNN) and a newly designed Grid-RNN was investigated. The Grid-RNN strongly outperforms a CNN if different sets of parameters for different frequency bands are used and can be further enhanced by using a bi-directional Grid-RNN. Experiments using the multi-genre broadcast (MGB3) English data (275h) show that deep kernel TDNNs reduces the word error rate (WER) by 6% relative and when combined with the frequency dependent Grid-RNN gives a relative WER reduction of 9%.

Index Terms— Time Delay Neural Network, Grid Recurrent Neural Network, Speech Recognition, ResNet

1. INTRODUCTION

Artificial Neural Networks have become the dominant approach to acoustic modelling, achieving dramatic improvements on a range of tasks. One commonly used neural network architecture is the time-delay neural network (TDNN), originally proposed in [1] and often used in its *sub-sampled* form [2]. A TDNN consists of identical fully-connected (FC) layers repeated at different time-steps. It is thus seen as a forerunner to the convolutional neural network (CNN) [3], which applies the FC layer with specified regular shifts. While the TDNN does not have this restriction, the CNN formulation extends easily to multiple axes e.g. time and frequency. Shifting the FC layer incorporates the assumption that the same important features can occur at various time-steps. Incorporating this knowledge reduces the computation and number of parameters required. It is also known that a difference in speaker can express itself in small shifts in frequency. This knowledge can be incorporated into the model by applying CNNs along the frequency dimension [4], which will be one extension to the TDNN that this work evaluates.

Deepening neural networks continues to yield improvements in performance. A striking example in computer vision is the improvement on the ImageNet classification task when transitioning from AlexNet [5] (8 Layers), to VGG [6] (19 Layers) to ResNet [7] (152 Layers). These were all based on CNNs and the increase in depth came from stacking more convolutional layers. This work proposes a method of deepening a TDNN, similar to the one proposed in [8], which is to make the kernel used for the temporal convolution deep. This results in a potentially much more complex kernel.

Thanks to Mark Gales and the MGB3 team for the MGB3 setup used. Florian Kreyssig is supported by the Studienstiftung des Deutschen Volkes.

Two dimensional recurrent neural networks (RNNs) are gaining popularity over CNNs for the modelling of spectro-temporal variations [9, 10, 11, 12]. Here, an efficient Grid-RNN is designed, which can be used as the input to the TDNN architecture and uses a separate set of parameters for different frequency bands. Experiments on the multi-genre broadcast English (MGB3) challenge task are used to evaluate the different changes in architecture and demonstrate the efficacy of the proposed improvements.

The remainder of this paper is organised as follows. Section 2 outlines the proposed extensions to the sub-sampled TDNN architecture and discusses related work. Sections 3 and 4 present the experimental setup and results and Sec. 5 gives conclusions.

2. MODELS INVESTIGATED

2.1. Time-Delay Neural Networks

A TDNN starts with an FC layer that takes a stack of frames as its input and is replicated across different time-steps. The following layer then also takes as input a stack of different time-steps of the preceding layer and is also replicated across different time-steps. The initial layers thus learn to detect features within narrow temporal contexts while the later layers operate on a much larger temporal context.

The original TDNN formulation [1, 13] uses shifts of one frame in time for the FC layers, which is very expensive both in terms of computation and the numbers of parameters when operating on large temporal contexts. It was shown in [2] that neither one frame shifts nor uniform time shifts are necessary. Their proposed sub-sampled TDNN, illustrated in Fig. 1, is constructed by moving the first FC layer across the window $t \in [-13, 9]$ with a temporal context of 5 frames at shifts of 3 frames, thus splitting the total input context into 7 time-bins. This is followed by a binary tree combination of the outputs of the layer. This TDNN will be the baseline acoustic model in this paper. During back-propagation, gradients are accumulated over all instantiations of the FC layers and then normalised.

2.2. Deep Kernels

The sub-sampled TDNN has an effective and well-studied structure. However, the structure of sub-sampled TDNNs generally limits their depth, which in turn limits their modelling strength. Hence, in order to keep to the underlying structure, we try to increase the modelling capability of each of the kernels. For image classification, [8] proposed a CNN structure in which the convolutional filter is replaced by a stack of FC layers. This means that each filter can learn to represent more abstract features.

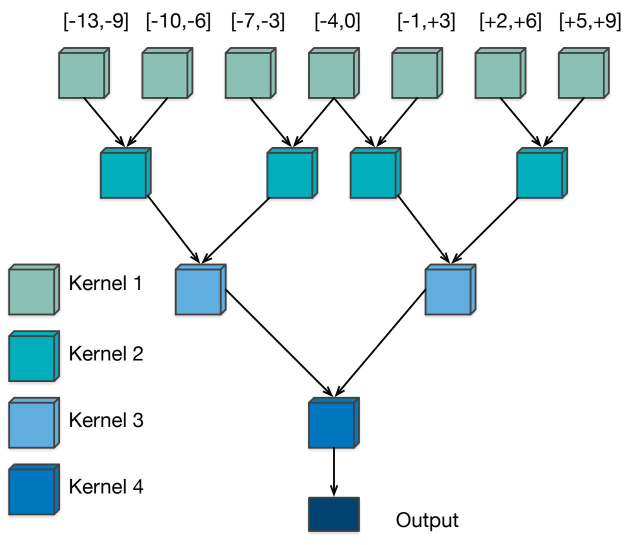

Fig. 1. Baseline TDNN Structure

Three types of temporal kernels, shown in Fig. 2, are evaluated in our experiments. The first kernel is the *Standard*-Kernel, a simple FC layer as used in the baseline structure. The second kernel, the *Double*-Kernel, is built by using two FC layers instead of one. Finally, the *ResNet*-Kernel, is constructed by appending an FC layer with two further FC layers that can be bypassed by a residual connection. Using the ResNet-Kernel increases the depth of the TDNN (including the output layer) from 5 layers to 13.

Even though deeper networks generally improve the performance of neural network architectures, they are usually harder and slower to train. Residual connections [7] are identity mappings from the output of initial layers to the input of later layers and can be described by:

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \theta) + \mathbf{x} \tag{1}$$

where $\mathbf{x}$ and $\mathcal{F}(\mathbf{x}, \theta)$ are the input and the output of the block of layers that is to be "skipped". This direct connection means that back-propagation of the gradients in very deep structures is more effective, since the effective minimum depth, in terms of layers, is reduced. For the ResNet-Kernel, the effective minimum depth is reduced from three layers to a single layer. Furthermore, it has been hypothesised that it is simpler to optimise the residual function $\mathcal{F}$ than the combined mapping. The residual function $\mathcal{F}$ in the *ResNet*-Kernel is an FC layer with Sigmoid activation function, $\sigma(\cdot)$, followed by an FC layer with linear activation function. A σ activation function cannot be used in the last layer of the *ResNet*-Kernel since, due to its non-negative output range, it could only increase the input signal. Hence, here a linear activation function is used.

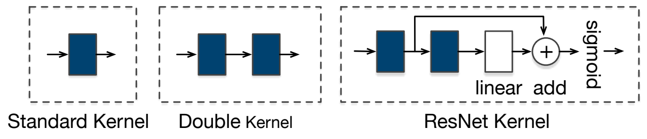

Fig. 2. Different kernels investigated. Darker blocks are FC layers with $\sigma(\cdot)$ activation function. The white block denotes an FC layer with linear activation function.

2.3. Frequency Domain Convolution

CNNs have shown promising results on a range of speech tasks [4, 14, 15, 16, 17, 18]. In a CNN layer a set of filters is convolved with the input which results in multiple output-maps, one per filter. This is followed by the application of an element-wise activation function, such as the $\sigma(\cdot)$ function. The operation that the layer performs on an input map with two axes, such as a spectrogram (time $\times$ frequency), can be written as:

$$h_{i,j,k} = \sigma\left(\sum_{l=0}^{L-T}\sum_{m=0}^{M-F} x_{i+l,j+m} \cdot w_{l,m,k}\right), k = 1\dots K \tag{2}$$

where L is the dimensionality of the time-axis, M is the dimensionality of the frequency-axis, $T \times F$ is the size of the filters, k is the index of the filter and K is the number of filters. $w_{l,m,k}$ are the learned parameters of the CNN. This is generally followed by a *pooling* operation which "summarises" patches in each output map by either computing their average (average-pooling) or their maximum value (max-pooling). This allows for some invariance to shifts in the location of a feature.

In the above description, the same filter would be applied across the entire input space, known as *full weight sharing* (FWS). This assumes that a feature can occur across the entire input space. This is a valid assumption for the temporal axis and hence done in the TDNN architecture. However, it is not generally the case for the frequency axis as the characteristics are different for higher and lower frequencies. To incorporate this into the model design, *limited weight sharing* (LWS) is introduced in [4, 18], where a specific filter is used only for a specific part of the frequency axis, the outputs of which are then max-pooled to a single scalar. Initial experiments did not find this method to give significant improvements, possibly due to the strong loss of information from the large pooling sizes. Hence, we introduce a convolution strategy that strikes a balance between FWS and LWS. The frequency axis is divided into different, but overlapping, frequency bands and convolution followed by max-pooling is performed within these frequency bands and the outputs concatenated. In comparison to LWS the pooling operation does not span the entire frequency band. The convolution can then be described by:

$$h^{f}_{i,j,k} = \sigma\left(\sum_{l=0}^{L-T}\sum_{m=f\cdot S}^{M+f\cdot S-F} x_{i+l,j+m} \cdot w_{l,m,k}\right), k = 1\dots K \tag{3}$$

where f is the index of the frequency band and S is the shift between the frequency bands. The frequency bands are size $M = 10$ and overlap by 5, giving $S = 5$. The input is 40-dimensional along the frequency axis resulting in 7 frequency bands. The filter has size $T \times F = 5 \times 5$ resulting in a size of 6×1 for the output which is reduced to 3×1 by the max-pooling layer that has pooling size 2.

2.4. Frequency Dependent Grid-RNN

RNNs are widely used in speech recognition, often in the form of the Long Short-Term Memory (LSTM) architecture [19, 20, 21]. Recently, standard time-domain LSTMs have been extended to model both the time and the frequency dimensions [22, 9]. This is done by unfolding the two-dimensional (2D) RNNs along both time and frequency. This gives the advantage over CNNs of being able to model correlations between features in time and frequency. Grid-LSTMs [23] have been shown to outperform CNNs as well as the TF-LSTMs [9] as the input to 1-D LSTM layers [10]. Both Grid-LSTMs and TF-LSTMs were unfolded in time one time-step at a time and used the computationally expensive LSTM.

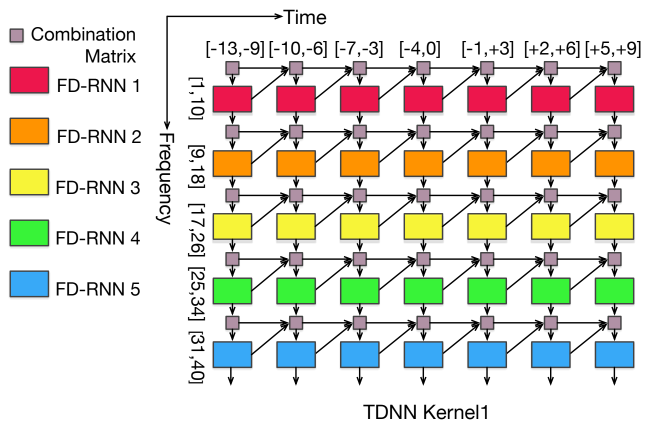

Fig. 3. Frequency Dependent Grid-RNN TDNN

This paper proposes an efficient Grid-RNN architecture, shown in Fig. 3 that uses the vanilla-RNN and groups the input window ($t \in [-13, 9]$) into the same seven time-bins as described in Sec. 2.1. Thus, it can be neatly combined with the previously discussed TDNN architectures. The frequency axis is split into five different bins of size ten, which are shown in Fig. 3. The Grid-RNN structure is composed of two RNNs, one with the $\sigma(\cdot)$ activation function and one with a linear activation. The σ-RNN ($\mathbf{h}^F_{t,k}$) performs feature extraction and the *Linear*-RNN ($\mathbf{h}^I_{t,k}$), called the *Combination Matrix* in Fig. 3, models the information flow between instantiations of the σ-RNN. The linear activation function is used for its improved flow of information over the $\sigma(\cdot)$ activation function. The structure is trained in the unfolded form shown, similar to the work done in [24]. The Grid-RNN is described by the following equations:

$$\mathbf{h}^I_{t,k} = V^I_F \mathbf{h}^F_{t,k-1} + V^I_I \mathbf{h}^I_{t-1,k} + \mathbf{b}^I \tag{4}$$

$$\mathbf{h}^F_{t,k} = \sigma\left(W^F_{(k)} \mathbf{x}_{t,k} + V^F_{(k)} \mathbf{h}^I_{t,k-1} + \mathbf{b}^F_{(k)}\right) \tag{5}$$

where $W^F_{(k)}$ is the frequency-dependent (FD) input weight-matrix, $V^F_{(k)}$ is the FD recurrent weight-matrix and V^I_α are the weight-matrices modulating the information flow. Here, t is the time-band and k is the frequency-band.

For comparison we can formulate vanilla-RNN versions of the TF-LSTM, which we coin TF-RNN,

$$\mathbf{h}_{t,k} = \sigma\left(W\mathbf{x}_{t,k} + V_T \mathbf{h}_{t-1,k} + V_F \mathbf{h}_{t,k-1} + \mathbf{b}\right) \tag{6}$$

and the 2D-Grid-LSTM, which uses two separate LSTMs to model the correlations in frequency and in time:

$$\mathbf{h}^T_{t,k} = \sigma\left(W^T \mathbf{x}_{t,k} + V_T \mathbf{h}^T_{t-1,k} + V_F \mathbf{h}^F_{t,k-1} + \mathbf{b}^T\right) \tag{7}$$

$$\mathbf{h}^F_{t,k} = \sigma\left(W^F \mathbf{x}_{t,k} + V_T \mathbf{h}^T_{t-1,k} + V_F \mathbf{h}^F_{t,k-1} + \mathbf{b}^F\right) \tag{8}$$

From Eqn. 6 it can be seen that the TF-RNN is equivalent to the proposed structure if the *Linear*-RNN of Eqn. 4 is replaced by a concatenation of the two inputs $\mathbf{h}^F_{t,k-1}$ and $\mathbf{h}^I_{t-1,k}$. This structure would have a much longer series of non-linear mappings leading to a potentially very strong loss of information as it moves through the network. By comparison the *Linear*-RNN provides a linear path in time through the network. This is partly analogous to the "skip" connections in time within the TF-LSTM or in frequency and time for the Grid-LSTM that is provided by the memory cell.

Using the same reasoning as in Sec. 2.3, it should be beneficial to untie the parameters of the σ-RNN along the frequency axis, denoted by a frequency dependent Grid-RNN (FD-Grid-RNN). This is shown by the colour-coding in Fig. 3 and by the potentially frequency-band-specific weights in Eqn. 5.

The Grid-RNN has an inherent directionality from the past to the future and from lower frequencies to higher frequencies. The recurrent weight-matrices V^α_β (Eqn. 4-5) of the two RNNs are likely to have a spectral radius below 1, especially due to our use of L2-regularisation. This means that any information provided to the model vanishes exponentially fast [25]. Thus, it is hypothesised that the model can be improved by using a bi-directional FD-Grid-RNN (BD-FD-Grid-RNN), which is constructed by training two FD-Grid-RNNs in parallel, one with the directionality as in Fig. 3 and one with the directions along both the time and frequency axes reversed. The outputs of both are concatenated. This bi-directionality does not increase the inherent latency of the model due its unfolded structure.

3. EXPERIMENTAL SETUP

The proposed architectures were evaluated using multi-genre broadcast (MGB) data [26] from the MGB3 speech recognition challenge task [27]. A 275 hour (275h) training set was selected from 750 episodes where the sub-titles have a phone matched error rate $< 40\%$ compared with the lightly supervised output [28] which was used as training supervision. A 55 hour (55h) subset was sampled at the utterance level from the 275h set. A 63k word vocabulary [29] was used and a trigram word level language model (LM) estimated from both the acoustic transcripts and a separate 640 million word MGB subtitle archive. The test set, **dev17b**, contains 5.55 hours of audio data and 5,201 manually segmented utterances from 14 episodes of 13 shows. System outputs were evaluated with confusion network decoding (CN) [30, 31] as well as 1-best Viterbi decoding.

All experiments were conducted with an extended version of HTK 3.5 [32, 33]. A 40d log-Mel filterbank (FBK) analysis was used without any delta coefficients.[1] These inputs were normalised at the utterance level for mean and at the show-segment level for variance [34]. All models were trained using the cross-entropy criterion and frame-level shuffling used.

About 6k/9k decision tree clustered triphone tied-states along with GMM-HMM/DNN-HMM system training alignments were used for the 55h/275h training sets. The NewBob$^+$ learning rate scheduler [35] was used to train all models with the setup for our previous MGB systems [34]. An initial learning rate of 2×10^{-3} was used for all models and a 800 frame minibatch. *L2*-regularisation was used was and tuned for the 55h systems but not tuned for the 275h systems. To give further context on the MGB3 data, the results are compared to a two layer projected LSTM (LSTMP) followed by an FC layer of same size before the output layer. The LSTMPs were implemented following [20]. For the 55h data set the width of the hidden layers is 500 and the projected vector size is 250. For the 275h dataset these were increased to 1000 and 500 respectively.

4. EXPERIMENTAL RESULTS

4.1. Comparison of Kernels

The different types of kernels given in Fig. 2 were investigated by using them for each kernel location in the TDNN shown in Fig. 1.

[1]LSTMP baselines used 40d log-Mel filterbank analysis $+\Delta$ coeffs.

ID	System	vit	cn
ST_1^{55h}	TDNN	33.6	32.7
DT_1^{55h}	Double-TDNN	32.1	31.5
RT_1^{55h}	ResNet-TDNN	31.1	30.5
ST_2^{55h}	TDNN + 1 FC	32.7	31.9
ST_3^{55h}	TDNN + 2 FC	31.6	30.9
ST_4^{55h}	TDNN+Deep	31.2	30.5
RT_2^{55h}	ResNet-TDNN+Deep	30.5	29.8
SC_1^{55h}	CNN-TDNN	31.6	31.0
RC_1^{55h}	CNN-ResNet-TDNN	30.7	29.9
SG_1^{55h}	Grid-RNN-TDNN	31.2	30.5
RG_1^{55h}	Grid-RNN-ResNet-TDNN	30.8	30.1
RG_2^{55h}	FD-Grid-RNN-ResNet-TDNN	30.2	29.6
RG_3^{55h}	BD-FD-Grid-RNN-ResNet-TDNN	29.7	29.0
L_1^{55h}	2L-LSTMP	31.3	30.6

Table 1. *%WERs for 55h systems on dev17b. Results are with a trigram LM and Viterbi decoding (vit) or CN decoding (cn).*

The WERs for these structures are given in the first section of Table 1. The number of parameters in each of the models is kept roughly constant at 6.6M parameters, by adjusting the layer width. Therefore the layer widths for the TDNNs using the Standard-Kernel (TDNN) and ResNet-Kernel (ResNet-TDNN) are 653 and 500 respectively. The Double Kernel gives some WER reduction, but not as much as the ResNet Kernel. This might due to the lower minimum path through the network of the ResNet-TDNN. Given that the relative WER reduction (WERR) due to confusion network decoding is less for DT_1^{55h} & RT_1^{55h} than for ST_1^{55h} (1.9% and 1.9% compared to 2.7%), it can be inferred that the deep kernels also sharpen the network output distributions.

4.2. Appending Fully-Connected layers

To validate the improvement from adding deep kernels to the TDNN, the model is deepened in a simpler fashion. A number of FC layers are inserted between Kernel 4 and the output layer, which is equivalent to deepening Kernel 4. FC layers were added to Kernel 4 until there is no further improvement in WER, again keeping the overall number of parameters in the network constant at 6.6M.[2]

The second section of Table 1 shows that TDNN architectures improve with increased depth of Kernel 4. The WER of a standard TDNN where Kernel 4 is four layers deep (TDNN+Deep) is similar to that of the ResNet-TDNN. However, both changes are complimentary and replacing Kernel 4 in the ResNet-TDNN by 3 FC layers (ResNet-TDNN+Deep) without any residual connection gives a further improvement. The WER reduction from the ResNet-Kernels increases with the 275h dataset as shown in Sec. 4.5. The optimal number of additional FC layers will depend on the task and the data set. Hence, adding additional FC hidden layers to the TDNN using the CNN or the Grid-RNN was not investigated.

4.3. Addition of Frequency Convolution

The third section of Table 1 shows the improvements from adding frequency domain convolution to the TDNN structure with the Standard-Kernel (CNN-TDNN) and with the ResNet-Kernel (CNN-ResNet-TDNN). The frequency convolution used 100 filters per frequency band. The larger output size of the CNN layer results an increase in the total number of parameters. For the Standard-Kernel and the ResNet-Kernel, rel. WERRs of 5.2% and 2.0% (ST_1^{55h} vs. SC_1^{55h} and RT_1^{55h} vs. RC_1^{55h}) respectively were achieved.

[2]The effect of increasing the hidden layers' size is small as shown by experiments using these structures trained with a hidden layers of 1000 nodes.

4.4. Adding the Grid-RNN

Section four of Table 1 shows the improvements from adding the Grid-RNN to the TDNN with the Standard-Kernel (Grid-RNN-TDNN) and with the ResNet-Kernel (Grid-RNN-ResNet-TDNN). The Grid-RNN had a width of 250 for the σ-RNN and of 500 for the *Linear*-RNN. For the Standard-Kernel, a rel. WERR of 6.7% (ST_1^{55h} vs. SG_1^{55h}) was achieved. For the ResNet-Kernel the relative reduction is 1.3% (RT_1^{55h} vs. RG_1^{55h}) which increases to 3.0% as the parameters across the frequency domain are untied (RG_2^{55h}), and to 4.9% for the bidirectional FD-Grid-RNN-ResNet-TDNN (RG_3^{55h}). This is an overall 11.3% rel. improvement over the baseline TDNN (ST_1^{55h} vs. RG_3^{55h}). These experiments show the strength of the FD-Grid-RNN over the CNN. Besides the ability to model correlations between features in time and in frequency as discussed in previous work, the FD-Grid-RNN has many more parameters designated to the spectro-temporal modelling. In the CNN only 17.5K independent parameters are used for the convolution, a very small fraction of the total number of parameters, as opposed to the BD-FD-Grid-RNN with 1.4M parameters. At the same time the input to the first TDNN-Kernel is larger for the CNN. This shows an issue with the CNN, which is that it only uses a very small number of parameters if the input to the first TDNN-Kernel is kept at a be reasonable size.

4.5. Further Experiments

The performance of key modelling approaches was also tested using the larger 275h training set, and the results shown in Table 2. Each of the models has a hidden layer width of 1000, except the σ-RNN in the Grid-RNNs, which is size 500. The deep kernels continue to give considerable improvement on the larger dataset. Further, the improvement due to the deep kernels scales better to the larger dataset than simply adding hidden layers: ResNet-TDNN+Deep has a 4% relative lower WER than TDNN+Deep in comparison to 2% on the smaller dataset (RT_2^{α} vs. ST_4^{α}).

ID	System	vit	cn
ST_1^{275h}	TDNN	27.3	26.7
RT_1^{275h}	ResNet-TDNN	25.4	25.0
ST_4^{275h}	TDNN+Deep	26.2	25.7
RT_2^{275h}	ResNet-TDNN+Deep	25.1	24.7
RG_3^{275h}	BD-FD-Grid-RNN-ResNet-TDNN	24.6	24.3
L_1^{275h}	2L-LSTMP	26.0	25.6

Table 2. *%WERs for 275h systems on dev17b. Results are with a trigram LM and Viterbi decoding (vit) or CN decoding (cn).*

5. CONCLUSION

In this paper, we presented different extensions to the *sub-sampled* TDNN architecture. Deep Kernels for more abstract feature extraction as well as CNNs and 2D-RNN to reduce the spectro-temporal variation of the input feature. We propose a 2D-RNN architecture that does not rely on LSTMs and is complimentary to the TDNN architecture. We found that using both Deep Kernels and the 2D-RNN offers the results in the best performance. Overall, the combined structure yields a 9% relative reduction in WER over the baseline TDNN architecture.

6. REFERENCES

[1] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J Lang, "Phoneme recognition using time-delay neural networks," *IEEE Trans ASSP*, vol. 37, no. 3, pp. 328–339, 1989.

[2] V. Peddinti, D. Povey, and S. Khudanpur, "A time delay neural network architecture for efficient modeling of long temporal contexts.," *Proc. Interspeech*, Dresden, 2015.

[3] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[4] O. Abdel-Hamid, A. Mohamed, H. Jiang, and G. Penn, "Applying convolutional neural networks concepts to hybrid NN-HMM model for speech recognition," *Proc. ICASSP*, Kyoto, 2012.

[5] A. Krizhevsky, I. Sutskever, and G.E. Hinton, "Imagenet classification with deep convolutional neural networks," *Proc. NIPS*, Lake Tahoe, 2012.

[6] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *Proc. ICLR*, San Diego, 2015.

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proc. CVPR*, Las Vegas, 2016.

[8] M. Lin, Q. Chen, and S. Yan, "Network in network," *Proc. ICLR*, Scottsdale, 2013.

[9] J. Li, A. Mohamed, G. Zweig, and Y. Gong, "Exploring multidimensional LSTMs for large vocabulary ASR," *Proc. ICASSP*, Shanghai, 2016.

[10] T. Sainath and B. Li, "Modeling time-frequency patterns with LSTM vs. convolutional architectures for LVCSR tasks," *Proc. Interspeech*, San Francisco, 2016.

[11] B. Li, T. Sainath, A. Narayanan, J. Caroselli, M. Bacchiani, A. Misra, I. Shafran, H. Sak, G. Pundak, K. Chin, K.C. Sim, R.J. Weiss, K. Wilson, E. Variani, C. Kim, O. Siohan, M. Weintraub, E. McDermott, R. Rose, and M. Shannon, "Acoustic modeling for Google Home," *Proc. Interspeech*, Stockholm, 2017.

[12] B. Li and T. Sainath, "Reducing the computational complexity of two-dimensional LSTMs," *Proc. Interspeech*, Stockholm, 2017.

[13] A. Waibel, "Modular construction of time-delay neural networks for speech recognition," *Neural computation*, vol. 1, no. 1, pp. 39–46, 1989.

[14] T.N. Sainath, A. Mohamed, B. Kingsbury, and B. Ramabhadran, "Deep convolutional neural networks for LVCSR," *Proc. ICASSP*, Vancouver, 2013.

[15] T.N. Sainath, O. Vinyals, A. Senior, and H. Sak, "Convolutional, long short-term memory, fully connected deep neural networks," *Proc. ICASSP*, Brisbane, 2015.

[16] T. Sercu, C. Puhrsch, B. Kingsbury, and Y. LeCun, "Very deep multilingual convolutional neural networks for LVCSR," *Proc. ICASSP*, Shanghai, 2016.

[17] Y. Qian and P.C. Woodland, "Very deep convolutional neural networks for robust speech recognition," *Proc. SLT*, San Diego, 2016.

[18] O. Abdel-Hamid, A. 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*, vol. 22, pp. 1533–1545, 2014.

[19] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[20] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," *Proc. Interspeech*, Singapore, 2014.

[21] V. Peddinti, Y. Wang, D. Povey, and S. Khudanpur, "Low latency acoustic modeling using temporal convolution and LSTMs," *IEEE Signal Processing Letters*, 2017.

[22] J. Li, A. Mohamed, G. Zweig, and Y. Gong, "LSTM time and frequency recurrence for automatic speech recognition," *Proc. ASRU*, Scottsdale, 2015.

[23] Nal Kalchbrenner, Ivo Danihelka, and Alex Graves, "Grid long short-term memory," *Proc. ICLR*, San Juan, 2016.

[24] G. Saon, H. Soltau, A. Emami, and M. Picheny, "Unfolded recurrent neural networks for speech recognition," *Proc. Interspeech*, Singapore, 2014.

[25] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," *Proc. ICML*, Atlanta, 2013.

[26] P. Bell, M.J.F. Gales, T. Hain, J. Kilgour, X. Liu P. Lanchantin, A. McParland, S. Renals, O. Saz, M. Wester, and P.C. Woodland, "The MGB challenge: Evaluating multi-genre broadcast media transcription," *Proc. ASRU*, Scottsdale, 2015.

[27] "http://www.mgb-challenge.org," .

[28] P. Lanchantin, M.J.F. Gales, P. Karanasou, X. Liu, Y. Qian, L. Wang, P.C. Woodland, and C. Zhang, "Selection of Multi-Genre Broadcast data for the training of automatic speech recognition systems," *Proc. Interspeech*, San Francisco, 2016.

[29] K. Richmond, R. Clark, and S. Fitt, "On generating Combilex pronunciations via morphological analysis," *Proc. Interspeech*, 2010.

[30] L. Mangu, E. Brill, and A. Stolcke, "Finding consensus in speech recognition: Word error minimization and other applications of confusion networks," *Computer Speech and Language*, vol. 14, pp. 373–400, 2000.

[31] G. Evermann and P. Woodland, "Large vocabulary decoding and confidence estimation using word posterior probabilities," *Proc. ICASSP*, Istanbul, 2000.

[32] S. Young, G. Evermann, M. Gales, T. Hain, D. Kershaw, X. Liu, G. Moore, J. Odell, D. Ollason, D. Povey, A. Ragni, V. Valtchev, P. Woodland, and C. Zhang, "The HTK book (for HTK version 3.5)," *Cambridge University Engineering Department*, 2015.

[33] C. Zhang and P.C. Woodland, "A general artificial neural network extension for HTK," *Proc. Interspeech*, Dresden, 2015.

[34] P.C. Woodland, X. Liu, Y. Qian, C. Zhang, P. Karanasou M.J.F. Gales, P. Lanchantin, and L. Wang, "Cambridge university transcription systems for the Multi-Genre Broadcast challenge," *Proc. ASRU*, Scottsdale, 2015.

[35] C. Zhang, *Joint Training Methods for Tandem and Hybrid Speech Recognition Systems using Deep Neural Networks*, Ph.D. thesis, University of Cambridge, Cambridge,UK, 2017.