INTERSPEECH 2017
August 20–24, 2017, Stockholm, Sweden

Compressed time delay neural network for small-footprint keyword spotting

Ming Sun[1†], *David Snyder*[2†*], *Yixin Gao*[1†], *Varun Nagaraja*[1], *Mike Rodehorst*[1], *Sankaran Panchapagesan*[1], *Nikko Strom*[1], *Spyros Matsoukas*[1], *Shiv Vitaladevuni*[1]

[1]Alexa Machine Learning, Amazon.com
[2]Center for Language and Speech Processing, Johns Hopkins University

mingsun@amazon.com, david.ryan.snyder@gmail.com,
{yixigao,nagavaru,mikerode,panchi,nikko,matsouka,shivnaga}@amazon.com

Abstract

In this paper we investigate a time delay neural network (TDNN) for a keyword spotting task that requires low CPU, memory and latency. The TDNN is trained with transfer learning and multi-task learning. Temporal subsampling enabled by the time delay architecture reduces computational complexity. We propose to apply singular value decomposition (SVD) to further reduce TDNN complexity. This allows us to first train a larger full-rank TDNN model which is not limited by CPU/memory constraints. The larger TDNN usually achieves better performance. Afterwards, its size can be compressed by SVD to meet the budget requirements. Hidden Markov models (HMM) are used in conjunction with the networks to perform keyword detection and performance is measured in terms of area under the curve (AUC) for detection error tradeoff (DET) curves. Our experimental results on a large in-house far-field corpus show that the full-rank TDNN achieves a 19.7% DET AUC reduction compared to a similar-size deep neural network (DNN) baseline. If we train a larger size full-rank TDNN first and then reduce it via SVD to the comparable size of the DNN, we obtain a 37.6% reduction in DET AUC compared to the DNN baseline.

Index Terms: keyword spotting, time delay neural network, singular value decomposition, small-footprint.

1. Introduction

Keyword spotting refers to the task of detecting spoken words of interest in audio signals. It has been an active research area in speech recognition for decades, and widely used in numerous applications. As one approach, general large vocabulary continuous speech recognition (LVCSR) systems are applied to decode the audio signal, with keyword searching conducted in the resulting lattices or confusion networks [1, 2, 3]. These methods require relatively high computational resources for the LVCSR decoding, and also introduce latency.

Recently, with increasing popularity of voice assistant systems, small-footprint keyword spotting systems have been attracting more attention [4, 5, 6]. For example, Alexa on Amazon Tap requires a keyword spotting system running continuously on the device under tight CPU, memory, latency and power usage constraints. The Tap device only starts to stream audio to the cloud for further processing after the local system gets triggered by the spoken keyword. Such embedded keyword spotting systems are designed to have high recall to make devices easy to use, as well as low false accepts to mitigate privacy concerns.

† These authors contributed equally to this work. *Work conducted while the author was at Amazon.com

Small-footprint keyword spotting systems often use hidden Markov models (HMM) to represent both the keyword and the background audio [7, 8, 9]. The background model is also called the filler model in some literature, and can be used to model non-keyword speech, or nonspeech noise etc. Typical background models consist of loops over simple speech/nonspeech phones, or for more complicated cases, a normal phone set or confusable word set. During decoding, Viterbi search is applied to find the best path in the decoding graph. The keyword spotting system is triggered when the likelihood ratio of keyword model vs. background model exceeds a pre-defined threshold. Traditionally, Gaussian mixture models (GMM) were commonly used to model the observed acoustic features. With deep neural networks (DNN) becoming mainstream for acoustic modeling, which outperform GMM consistently, this approach can be extended to include discriminative information by incorporating a hybrid DNN-HMM decoding framework [10].

As alternative approaches for small-footprint keyword spotting, in recent years, it has been proposed to build systems based on a single DNN or convolutional neural network (CNN), with no HMM involved [4, 11, 12, 13]. During decoding, framewise posteriors for keyword are smoothed within a sliding window. The system fires when smoothed keyword posteriors exceed a pre-defined threshold. The trade off between balancing false rejects and false accepts can be performed by tuning the threshold. Context information is incorporated by stacking frames as input. Some keyword spotting systems are built on a recurrent neural network (RNN) which can model sequential data. In the RNN category, long short-term memory (LSTM) is a popular model with its ability to deal with the vanishing gradient problem [14, 15, 16, 17].

Time delay neural networks (TDNN) are designed so that the initial layers focus on modeling narrow context information, while the higher layers learn from wider temporal context information [18, 19]. TDNN training computation can be reduced by sub-sampling its temporal connections [20]. Recently, TDNN has been widely applied to a variety of tasks [21, 22, 23]. In this work, we use a TDNN-HMM based approach for keyword spotting. To train the TDNN acoustic model, we employ transfer learning by initializing the TDNN model with an existing model of the same architecture trained using LVCSR phone targets, as well as multi-task training which uses a secondary layer with LVCSR phone targets to regularize the primary training task for keyword phone targets [10]. Similar to [20], subsampling is used to reduce TDNN complexity. On top of these techniques, we propose applying singular value decomposition (SVD) to further reduce the ranks of the affine transform matrices [24, 25, 13], which imposes further regularization on TDNN training. With SVD compression, we have the flexibility to train

http://dx.doi.org/10.21437/Interspeech.2017-480

a larger TDNN model first, then compress it to meet the model complexity requirement afterwards. We show that this approach generates a better TDNN model than directly training a TDNN with required model complexity, for a keyword spotting task.

The remainder of this paper is organized as follows: Section 2 introduces our keyword spotting system. Section 3 describes TDNN related details, including how SVD is applied for compression. Experimental setup and results are presented in Section 4. Section 5 is for the conclusion.

2. Keyword Spotting System

Our HMM based keyword spotting system is shown in Figure 1, which follows the conventional keyword vs. background/filler HMM structure. Here the keyword 'Alexa' is shown as an example. The keyword model consists a cascade of HMM states for 'Alexa' phones. Our system uses three-state HMMs to model each keyword phone. For simplicity purpose, single-stage HMMs are illustrated in Figure 1. The background/filler model loops over single-state HMMs for speech (SP) and non-speech (NSP).

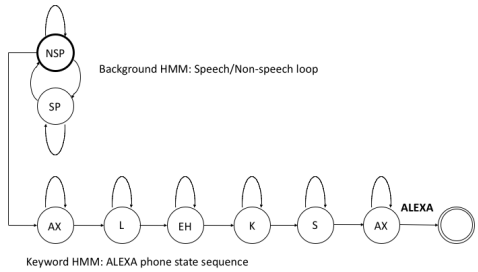

Figure 1: *HMM-based keyword spotting system*

DNNs or TDNNs are used as acoustic models for our HMM-based keyword spotter. The log mel filter-bank energies (LFBEs) are computed as input acoustic features. We extract 20-dimensional LFBEs over 25ms frames with a 10ms frame shift. Given the input acoustic features of each frame, the output layer of DNN/TDNN models its posterior distribution over the HMM states for both keyword and background/filler models. Viterbi search is run separately in the competing keyword and background/filler decoding graph, and the keyword hypothesized segment is extracted when the final state of the keyword model is reached. The likelihood ratio of the keyword vs background/filler model over the hypothesized keyword segment is computed, and the system triggers when this likelihood ratio exceeds a pre-defined threshold.

3. TDNN Architecture and Training

In a conventional DNN, temporal context information is captured by stacking l left frames and r right frames to form the whole input. Let d denote the feature dimension for each frame. Then the DNN has an input layer of dimension $d \times (l + 1 + r)$. By contrast, a TDNN processes the information from the context window in a hierarchical way. Figure 2 shows the architecture of our TDNN based acoustic model for keyword spotting. The input layer of TDNN focuses on modeling a narrow context, while the deeper layers of TDNN work on modeling wider temporal context information. For each hidden layer of TDNN, its parameters are tied across different time stamps, with its lower layers trained to learn translation invariant feature forms [18, 19]. For our TDNN architecture, its input layer processes a narrow context window of 5 consecutive frames ($l = 2, r = 2$). This is labeled as $[-2, 2]$. Our training recipe starts with the approach described in [10] with transfer learning and multi-task learning, but we replace the DNN with a temporal connection sub-sampled TDNN. The SVD compression proposed in this paper follows the ideas in [13]. Details will be presented in the remaining part of this section.

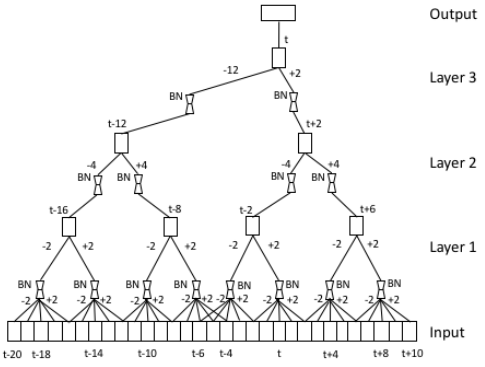

Figure 2: *TDNN architecture with SVD compression. 'BN' labels linear bottleneck layers.*

3.1. Sub-sampling

A sub-sampling based approach has been proposed to reduce the complexity of TDNN [20]. The idea is that a fully connected TDNN computes hidden layer activations at all time stamps, which causes redundancies due to the input context overlapping for the correlated neighboring activations. As a result, for each hidden layer, activations are computed only for a sampled subset of time stamps, which are fed to the next hidden layer. For example, as shown in Figure 2, at each time stamp, layer 1 splices sampled input at current frame minus 2 and current frame plus 2. This is labeled by time offset $\{-2, 2\}$.

3.2. Transfer Learning

Transfer learning is a widely used approach in machine learning, which transfers the knowledge learned from a related task to improve the main training task [26, 27, 28, 29]. As an application example of transfer learning on neural networks, the hidden layers of a trained network can be initialized from another network of the same size trained for a related task [4]. For our case, an LVCSR TDNN with the same architecture is trained to initialize the keyword spotting TDNN. This gives a better initial point and helps the keyword spotting TDNN converge to a better local optimum.

3.3. Multi-task Learning

Multi-task learning refers to the cases where two or more auxiliary tasks are jointly learned with the main task during training. It has been shown to be an effective method to regularize the learning of the main task, and hence improves the performance of the main task [30]. In the context of neural network training, one application of multi-task learning adds separate output layers for auxiliary tasks [31, 32, 33, 34, 10]. The lower hid-

den layers are shared among the main task and the auxiliary tasks, while the auxiliary tasks could have their own branches in higher hidden layers. As a result, additional parameters are added in training time for layers trained with auxiliary tasks. However, after training these branches of auxiliary tasks can be removed, and only the required branch of the main task is kept. Hence the decoding complexity is not increased.

Figure 3 shows the architecture for our multi-task training scheme, with an auxiliary training task with the LVCSR targets. As a result, we prepare two sets of targets for TDNN training data. One set is used for keyword spotting, which represents HMM states shown in Figure 1. The other set consists the LVCSR targets. Our TDNN architecture has two shared hidden layers for both keyword and LVCSR targets, while a separate hidden layer is added for each task before the output layer. It has been shown in our previous experiments that having the last hidden layer separated for each task could improve the performance of multi-task training [10].

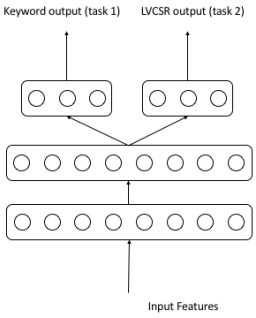

Figure 3: *Multi-task training*

Cross-entropy loss is used for TDNN training in our experiments. For multi-task learning, the total loss is calculated as

$$\mathcal{L}_t^* = \lambda \mathcal{L}_t^1 + (1-\lambda)\mathcal{L}_t^2 \quad (1)$$

Here $\mathcal{L}_t^1$ is the cross-entropy loss for the main task keyword spotting output layer, and $\mathcal{L}_t^2$ is the cross-entropy loss for the auxiliary task LVCSR output layer. The weighting factor λ can be tuned to balance the two tasks during training. For our experiments, we choose $\lambda = 0.9$ from development set.

3.4. SVD Approximation

In this paper we propose to further regularize and compress the TDNN using SVD. Linear bottlenecks based low-rank weight matrices have been extensively studied for speech recognition [24, 25, 13]. The total number of parameters can be reduced by inserting bottleneck layers into networks. To elaborate, let M and N denote the number of hidden nodes in two neighboring hidden layers. When a bottleneck layer of R nodes is added between, with R properly chosen, the number of multiplications could be reduced from $M \times N$ to $(M + N) \times R$. In general, LVCSR neural networks have the majority of their parameters in the input and output layer, and these layers are where SVD is mostly applied to. Instead, we apply SVD to approximate all hidden layers of our TDNN network, as well as the input layer. Figure 2 shows bottleneck layers (labeled by 'BN') are added to our TDNN model. It has been shown that DNN with SVD compression effectively improves keyword spotting performance [13].

To train TDNN with SVD approximation, we start with training a larger size full-rank TDNN at first. After that, we add linear bottleneck layers initialized by SVD of the full-rank affine matrices to TDNN, one layer at time, starting from the input layer. The dimensions of the linear bottleneck layers are selected to meet the parameter budget, and they are within a reasonable range to maintain the performance of full-rank TDNN measured by frame accuracy on the cross-validation set. One epoch of pre-training is applied at each time when a bottleneck layer is added. Finally the SVD-compressed TDNN is trained with additional epochs for the purpose of fine-turning.

3.5. Summary of TDNN Training Algorithm

The details of our TDNN training, including transfer learning, multi-task learning and SVD compression are summarized as follows:

i. Train a full-rank LVCSR TDNN with the same architecture as the full-rank keyword TDNN. The hidden layers of the LVCSR TDNN are used for initialization.

ii. Add a separate hidden layer and an output layer for the main keyword spotting task. Other hidden layers are initialized by the LVCSR TDNN model.

iii. Train a TDNN jointly with the keyword spotting task and LVCSR task using multi-task learning setup.

iv. Add linear bottleneck layers to the full-rank TDNN and pre-train. These linear bottleneck layers are initialized by SVD.

v. Run additional epochs of multi-task fine-tuning for the SVD compressed TDNN.

vi. Remove the last separate hidden layer and output layer for the LVCSR task. The remaining TDNN is used for keyword spotting.

4. Experimental Results

The word 'Alexa' is chosen as the keyword for our experiments. We use an in-house far-field corpus which contains far-field data collected under different conditions. This dataset contains an order of magnitude more instances of keyword and background speech utterances than the largest previous studies [4, 12] for model training, tuning and testing. Considering the large size of our corpus, the development set partition is sufficient to tune parameters, and the test set partition is large enough to show strong statistical difference.

4.1. Model Building

The GPU-based distributed trainer described in [35] is used for our experiments. Sigmoid function is used as the activation function. We use exponential decaying learning rate scheduling for TDNN training, including all stages of LVCSR TDNN training, full-rank TDNN multi-task training, and SVD-compressed TDNN multi-task fine-tuning. The initial learning rate is set to be 0.008 for both LVCSR TDNN training and full-rank TDNN multi-task training, and 0.000125 for SVD-compressed TDNN multi-task fine-tuning. The decaying factor is 2 for the first few epochs, and it is reduced to 1.2 for annealing in the remaining epochs.

For comparison purposes, we also train a baseline DNN model with SVD compression, as well as a full-rank TDNN without SVD compression. This SVD compressed DNN model

is trained following the same recipe as in Section 3.5. Our previous experiments have shown that SVD improves DNN performance for keyword spotting [13]. Hence this DNN model is a strong baseline. For the full-rank TDNN without SVD compression, we allow additional multi-task training epochs, so that the total number of training epochs are the same for the purpose of fair comparison. As a result, the DNN and TDNN with SVD have 20 epochs in the full-rank multi-task learning stage and 20 additional epochs after SVD compression, while the TDNN without SVD is trained with 40 epochs in the multi-task training directly. To enable a fair comparison, all three models have the same total temporal context and roughly the same number of parameters ($\leq 100k$). All three models are initialized by LVCSR models. They all have two shared hidden layers and one separate hidden layer for multi-task learning with the keyword and LVCSR tasks. The architecture details of all three models are summarized in Table 1.

Table 1: *Summary of model architecture details. Both DNN and TDNN-B are compressed with SVD, while TDNN-A has no SVD. All three models have similar number of parameters ($\leq 100k$)*

Model		**DNN**	**TDNN-A**	**TDNN-B**
Network Context		$[-20, 10]$	$[-20, 10]$	$[-20, 10]$
Input	Context	$[-20, 10]$	$[-2, 2]$	$[-2, 2]$
	Dim	620	100	100
Layer1	BN Dim	55	–	55
	Context	$\{0\}$	$\{-2, 2\}$	$\{-2, 2\}$
	Dim	200	135×2	193×2
Layer2	BN Dim	55	–	55
	Context	$\{0\}$	$\{-4, 4\}$	$\{-4, 4\}$
	Dim	200	135×2	193×2
Layer3	BN Dim	55	–	55
	Context	$\{0\}$	$\{-12, 2\}$	$\{-12, 2\}$
	Dim	200	135×2	193×2

TDNN-A represents the TDNN model with no SVD, while DNN and TDNN-B are compressed by SVD. For example, TDNN-B has input layer dimension 100 by splicing 5 frames (2 in past and 2 in future) of 20-dimensional LFBE features. The 100 dimensional input vector is connected to a bottleneck (BN) layer with dimension 55, which is later mapped to dimension 193 at layer 1. Since two time stamp contexts ($t-2$ & $t+2$) are spliced at layer 1, the dimension of layer 1 is labeled as 193×2, which is further connected to the bottleneck of next layer with dimension 55.

4.2. Evaluation

Our keyword spotting system is evaluated using two metrics: miss rate, which is one minus recall, and false accept rate, which is a normalized number of false accepts. Detection error tradeoff (DET) curves can be obtained by tuning the transition parameters and exit penalties for the HMM structure shown in Figure 1. The DET curves for all three models described in Section 4.1 are plotted in Figure 4.

The x-axis labels false accept rate, and the y-axis labels miss-rate. Absolute numbers of false accept rates have been obscured in this paper due to confidentiality. Instead, we plot false accept rates up to a multiplicative constant. The false accept range considered in our experiments is aligned with a low value range which can be considered for production deployment purpose. To compare overall performance, we compute area under the curve (AUC) numbers for HMM DET curves. Smaller

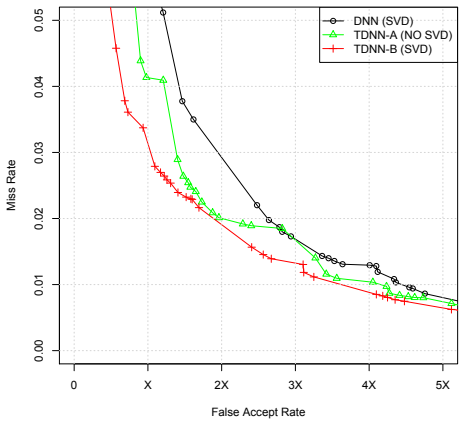

Figure 4: *DNN/TDNN-HMM DET curves for 'Alexa' detection.*

AUC numbers indicate better keyword spotting performance. The black circle curve represents the DNN, while the green triangle and red plus curves represent the TDNN without SVD and the TDNN with SVD, respectively. Note that all three models have a similar number of parameters. The DNN and TDNN-B are first trained using larger, full-rank affine matrices. This is followed by SVD compression which brings their total parameter count down to similar as the parameter count of TDNN-A. We observe that the TDNN improves over the DNN for keyword spotting performance, while the TDNN with SVD compression shows the best performance among all three models. Table 2 condenses the contents of Figure 4 into AUC relative change over the baseline. We see that the TDNN without SVD (TDNN-A) improves on the DNN baseline with 19.7% AUC reduction, while the TDNN with SVD (TDNN-B) further improves on the DNN baseline with 37.6% AUC reduction.

Table 2: *Relative change of HMM DET AUC for TDNN models without SVD compression (TDNN-A) and with SVD compression (TDNN-B), compared to the baseline SVD compressed DNN. All three models have comparable number of parameters ($\leq 100k$). Lower AUC indicates better performance*

Model	**DNN**	**TDNN-A**	**TDNN-B**
AUC Relative Change	0%	-19.7%	-37.6%

5. Conclusions

In this paper we present our work of building a TDNN-HMM based keyword spotting system. We propose to apply SVD compression on TDNN to regularize training and reduce complexity, together with LVCSR initialization, multi-task learning and temporal connection sub-sampling. Our experimental results show that for similar size models, the TDNN with no SVD compression already outperforms the SVD compressed DNN baseline by 19.7% HMM DET AUC reduction. Finally, the TDNN with SVD compression further improves keyword spotting performance, which shows 37.6% AUC reduction over the SVD compressed DNN baseline.

6. References

[1] D. R. Miller, M. Kleber, C.-L. Kao, O. Kimball, T. Colthurst, S. A. Lowe, R. M. Schwartz, and H. Gish, "Rapid and accurate spoken term detection," in *Eighth Annual Conference of the International Speech Communication Association*, 2007.

[2] G. Chen, O. Yilmaz, J. Trmal, D. Povey, and S. Khudanpur, "Using proxies for OOV keywords in the keyword search task," in *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*. IEEE, 2013, pp. 416–421.

[3] S. Tsakalidis, R. Hsiao, D. Karakos, T. Ng, S. Ranjan, G. Saikumar, L. Zhang, L. Nguyen, R. Schwartz, and J. Makhoul, "The 2013 BBN Vietnamese telephone speech keyword spotting system," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 7829–7833.

[4] G. Chen, C. Parada, and G. Heigold, "Small-footprint keyword spotting using deep neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 4087–4091.

[5] M. Sun, V. Nagaraja, B. Hoffmeister, and S. Vitaladevuni, "Model shrinking for embedded keyword spotting." in *Machine Learning and Applications (ICMLA), 2015 IEEE 14th International Conference on*. IEEE, 2015, pp. 369–374.

[6] Q. He, G. W. Wornell, and W. Ma, "An adaptive multi-band system for low power voice command recognition," *Interspeech 2016*, pp. 1888–1892, 2016.

[7] R. C. Rose and D. B. Paul, "A hidden Markov model based keyword recognition system," in *Acoustics, Speech and Signal Processing (ICASSP), 1990 IEEE International Conference on*. IEEE, 1990, pp. 129–132.

[8] J. G. Wilpon, L. R. Rabiner, C.-H. Lee, and E. Goldman, "Automatic recognition of keywords in unconstrained speech using hidden Markov models," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 38, no. 11, pp. 1870–1878, 1990.

[9] J. Wilpon, L. Miller, and P. Modi, "Improvements and applications for key word recognition using hidden Markov modeling techniques," in *Acoustics, Speech, and Signal Processing, 1991. ICASSP-91., 1991 International Conference on*. IEEE, 1991, pp. 309–312.

[10] S. Panchapagesan, M. Sun, A. Khare, S. Matsoukas, A. Mandal, B. Hoffmeister, and S. Vitaladevuni, "Multi-task learning and weighted cross-entropy for DNN-based keyword spotting," *Interspeech 2016*, pp. 760–764, 2016.

[11] P. Nakkiran, R. Alvarez, R. Prabhavalkar, and C. Parada, "Compressing deep neural networks using a rank-constrained topology." in *INTERSPEECH*, 2015, pp. 1473–1477.

[12] T. N. Sainath and C. Parada, "Convolutional neural networks for small-footprint keyword spotting." in *INTERSPEECH*, 2015, pp. 1478–1482.

[13] G. Tucker, M. Wu, M. Sun, S. Panchapagesan, G. Fu, and S. Vitaladevuni, "Model compression applied to small-footprint keyword spotting," in *Proc. Interspeech*, 2016.

[14] S. Fernández, A. Graves, and J. Schmidhuber, "An application of recurrent neural networks to discriminative keyword spotting," in *International Conference on Artificial Neural Networks*. Springer, 2007, pp. 220–229.

[15] M. Woellmer, B. Schuller, and G. Rigoll, "Keyword spotting exploiting long short-term memory," *Speech Communication*, vol. 55, no. 2, pp. 252–265, 2013.

[16] P. Baljekar, J. F. Lehman, and R. Singh, "Online word-spotting in continuous speech with recurrent neural networks," in *Spoken Language Technology Workshop (SLT), 2014 IEEE*. IEEE, 2014, pp. 536–541.

[17] M. Sun, A. Raju, G. Tucker, S. Panchapagesan, G. Fu, A. Mandal, S. Matsoukas, N. Strom, and S. Vitaladevuni, "Max-pooling loss training of long short-term memory networks for small-footprint keyword spotting." in *Spoken Language Technology Workshop (SLT), 2016 IEEE*. IEEE, 2016.

[18] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang, "Phoneme recognition using time-delay neural networks," *IEEE transactions on acoustics, speech, and signal processing*, vol. 37, no. 3, pp. 328–339, 1989.

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

[20] V. Peddinti, D. Povey, and S. Khudanpur, "A time delay neural network architecture for efficient modeling of long temporal contexts." in *INTERSPEECH*, 2015, pp. 3214–3218.

[21] V. Peddinti, G. Chen, D. Povey, and S. Khudanpur, "Reverberation robust acoustic modeling using i-vectors with time delay neural networks." in *INTERSPEECH*, 2015, pp. 2440–2444.

[22] D. Snyder, D. Garcia-Romero, and D. Povey, "Time delay deep neural network-based universal background models for speaker recognition," in *Automatic Speech Recognition and Understanding (ASRU), 2015 IEEE Workshop on*. IEEE, 2015, pp. 92–97.

[23] D. Povey, V. Peddinti, D. Galvez, P. Ghahrmani, V. Manohar, X. Na, Y. Wang, and S. Khudanpur, "Purely sequence-trained neural networks for ASR based on lattice-free MMI," in *INTERSPEECH*, 2016.

[24] T. N. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, and B. Ramabhadran, "Low-rank matrix factorization for deep neural network training with high-dimensional output targets," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 6655–6659.

[25] A. Senior and X. Lei, "Fine context, low-rank, softplus deep neural networks for mobile speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 7644–7648.

[26] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[27] M. Sun, C. E. Priebe, and M. Tang, "Generalized canonical correlation analysis for disparate data fusion," *Pattern Recognition Letters*, vol. 34, no. 2, pp. 194–200, 2013.

[28] M. Sun and C. E. Priebe, "Efficiency investigation of manifold matching for text document classification," *Pattern Recognition Letters*, vol. 34, no. 11, pp. 1263–1269, 2013.

[29] C. Shen, M. Sun, M. Tang, and C. E. Priebe, "Generalized canonical correlation analysis for classification," *Journal of Multivariate Analysis*, vol. 130, pp. 310–322, 2014.

[30] R. Caruana, "Multitask learning," in *Learning to learn*. Springer, 1998, pp. 95–133.

[31] G. Heigold, V. Vanhoucke, A. Senior, P. Nguyen, M. Ranzato, M. Devin, and J. Dean, "Multilingual acoustic models using distributed deep neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 8619–8623.

[32] M. L. Seltzer and J. Droppo, "Multi-task learning in deep neural networks for improved phoneme recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 6965–6969.

[33] P. Bell and S. Renals, "Regularization of context-dependent deep neural networks with context-independent multi-task training," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 4290–4294.

[34] R. Giri, M. L. Seltzer, J. Droppo, and D. Yu, "Improving speech recognition in reverberation using a room-aware deep neural network and multi-task learning," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 5014–5018.

[35] N. Strom, "Scalable distributed DNN training using commodity GPU cloud computing." in *INTERSPEECH*, vol. 7, 2015, p. 10.