

Compression of Acoustic Model via Knowledge Distillation and Pruning

Chenxing Li^{*†}, Lei Zhu[‡], Shuang Xu^{*}, Peng Gao[‡] and Bo Xu^{*}

^{*}Institute of Automation, Chinese Academy of Sciences, Beijing, P.R.China

[†]University of Chinese Academy of Sciences, Beijing, P.R.China

[‡]AI Lab, Rokid Inc. Beijing, P.R.China

{lichenxing2015, shuang.xu, xubo}@ia.ac.cn, {zhulei, gaopeng}@rokid.com

Abstract—Recently, the performance of speech recognition system based on neural network has been greatly improved. Arguably, this huge improvement can be mainly attributed to deeper and wider layers. These systems are more difficult to be deployed on the embedded devices due to their large size and high computational complexity. To address these issues, we propose a method to compress deep feed-forward neural network (DNN) based acoustic model. In detail, a state-of-the-art acoustic model is trained as the baseline model. In this step, layer normalization is applied to accelerating the model convergence and improving the generalization performance. Knowledge distillation and pruning are then conducted to compress the model. Our final model can achieve $14.59\times$ parameters reduction, $5\times$ storage size reduction and comparable performance compared with the baseline model.

Keywords—speech recognition, layer normalization, knowledge distillation, pruning, model compression

I. INTRODUCTION

In recent years, speech recognition system has been greatly improved with the application of deep learning. Speech recognition plays more and more important roles in human-computer interaction and gradually changes people's life. Long short-term memory recurrent neural network [1], [2], [3], gated recurrent neural network [4], deep convolutional neural network [5] and end-to-end methods [6], [7], [8], [9] have been applied to speech recognition system to further improve performance. What's more, residual net [10] and highway network [11] are widely used to increase the depth of the model.

Though deeper and bigger models have brought better performance, they all contain a large number of parameters, resulting in long training and testing time, over-fitting and storage size issues. At the same time, in some cases, the cumbersome model is not suitable. For example, some scenarios require that speech recognition system should have low latency, low computation cost and high accuracy. Therefore, it is necessary to build small-footprint speech recognition system to meet actual requirements. Model compression provides a practical solution. Model compression can be used to compress acoustic model in speech recognition system.

There is extensive literature in model compression. At present, the research of model compression mainly focuses on matrix factorization, knowledge distillation, pruning and quantization. In paper [12], [13], [14], [15], the methods of

matrix factorization, such as singular value decomposition, low-rank matrix factorization and structured linear layers, are introduced in detail. Another method is knowledge distillation [16], [17], [18], [19], [20], [21]. Knowledge distillation is also called teacher-student training. It uses teacher model to generate soft targets which are used to train the student model. Kullback-Leibler divergence is used to measure the distance between the two models. By using this method, student model can achieve comparable performance with teacher model. Hyper-parameter, named temperature, and the ground truth labels can be used to further improve performance. Recently, pruning has been applied in speech recognition [22], computer vision [23] and neural machine translation [24]. In paper [23], weights below the threshold are directly pruned in weight matrices. This method converts a fully connected network into a sparse network. By retraining the sparse network, performance can be recovered from weight reduction. This method can effectively reduce the number of parameters and maintain the model performance. Quantization [25] is also a very effective method. It reduces the number of bits required to represent the weight. However when quantization bits are less than 8, actual application requires customized hardware.

In this paper, we propose a method to integrate knowledge distillation and pruning to compress the acoustic model of speech recognition system, and layer normalization [26] is applied to accelerating the convergence and improve the generalization performance. Our method can achieve high compression rate and maintain accuracy. Besides, it is easy to be deployed. Firstly, a state-of-the-art deep feed-forward neural network (DNN) model is trained as teacher model. Secondly, knowledge distillation is used to transfer the knowledge from the teacher model to student model, and the temperature and the ground truth labels are used to further enhance performance. Thirdly the small model is compressed by removing weights which are less than the predefined threshold. Then the sparse model is retrained to recover from performance loss. Pruning and retraining can be repeated to further reduce model complexity.

In this experiment, performance is measured quantitatively by character error rate (CER). DNN-6-1024-LN represents DNN with 6 hidden layers and 1024 nodes in each layer, and layer normalization is adopted. DNN-3-512-KD and DNN-3-512-Pr are compressed models both with 3 hidden layers

and 512 nodes in each hidden layer. By using knowledge distillation, DNN-6-1024-LN can be compressed to DNN-3-512-KD. The number of parameters is reduced by a factor of 3.55. DNN-3-512-Pr can be obtained by pruning DNN-3-512-KD. This method can reduce the number of parameters by a factor of 4.10. In summary, DNN-6-1024-LN can be effectively compressed to DNN-3-512-Pr with 1.8% performance loss. The total number of parameters is reduced by a factor of 14.59, from 9.62 million to 0.66 million. The storage size is reduced by a factor of 5, and the feed-forward time is reduced by 1.77.

The rest of this paper is organized as bellow. In Section II we introduce the algorithms in detail. The data preparation and experiment setup are described in Section III, while the results of our experiment are presented in Section IV. Finally, Section V gives conclusions and discussions.

II. THE ALGORITHMS

A. Layer normalization

Training a state-of-the-art neural network with many layers and a large number of parameters always spends a lot of time. One way to reduce training time is to use batch normalization [27]. Batch normalization, over a mini-batch of training cases, uses the distribution of the summed input to a neuron to calculate a mean and variance. Then batch normalization uses the mean and variance to normalize the summed input to a neuron on each training case. However, the performance of batch normalization is highly affected by the size of the mini-batch. Besides, batch normalization can not be perfectly applied to recurrent neural network (RNN).

A recently proposed method, layer normalization, can address issues above successfully. Layer normalization uses the distribution of all of the summed input to neurons in a layer on a single case to calculate a mean and variance which are then used to normalize the summed input. Layer normalization can greatly reduce training time and improve generalization performance. Therefore, layer normalization is adopted in this paper. Specifically, layer normalization will be applied before the non-linearity. The normalization is done with the following formula:

$$x_{iL}^{LayerNorm} = \alpha_{iL} \times \frac{x_{iL} - \mu_L}{\sqrt{\sigma_L^2 + \epsilon}} + \beta_{iL}, \quad (1)$$

where μ_L, σ_L^2 are the mean and variance in layer L . In order to prevent the occurrence of zero, a small number ϵ is added to denominator. α_{iL} and β_{iL} are the scale and shift parameters for neuron i in layer L , which are trainable during training process. x_{iL} is the summed input for neuron i in layer L . Finally, x_{iL} can be normalized to $x_{iL}^{LayerNorm}$.

B. Knowledge distillation

Knowledge Distillation is firstly inspired by biological evolution. Small creatures will live in an optimal way for traveling and reproduction and continue acquiring knowledge from adults. The same in machine learning, Hinton [16] points that teacher model can be trained if it can extract information

from data easily. Then knowledge can be transferred from the teacher model to a student model which is suitable for deployment. In other words, the student model is trained to imitate the behaviors of the teacher model.

The training process of knowledge distillation is different from the traditional process. Labels are generated by the teacher model. In teacher model, because the incorrect answers may also carry a lot of knowledge about how teacher model tends to generalize, this information can be adopted to further improve the performance of student model. Therefore, class probabilities produced by the teacher model are used to train the small model. Cross entropy loss function is modified as:

$$\mathcal{L}^{KD}(\theta) = - \sum_j \tilde{y}_j \log y_j, \quad (2)$$

where y_j represents the output of the student model, and \tilde{y}_j is the output of the teacher model. In Eq. 2, \tilde{y}_j is no longer a one-hot vector. Optimizing the loss function is equal to minimize the Kullback-Leibler divergence between the teacher and the student model. Besides, some improvements can be applied to this method to further enhance performance. Firstly, to prevent the probabilities of incorrect answer play a very small role due to their small value, temperature is used in Softmax non-linearity to flatten posterior distribution. Softmax non-linearity is modified as follows:

$$y_j = \frac{\exp(z_j/T)}{\sum_j \exp(z_j/T)}, \quad (3)$$

where z_j is summed input of class j in output layer. T is the temperature. y_j is the output of class j . Temperature is used both in teacher and student model.

Secondly, it is beneficial to use the ground truth labels. By interpolating the ground truth labels into training process, better performance can be maintained. Thus we can get the following loss function:

$$\mathcal{L}(\theta) = \lambda \mathcal{L}^{KD}(\theta) + \frac{1-\lambda}{T^2} \mathcal{L}^{CE}(\theta), \quad (4)$$

λ is a tuned parameter. In this experiment, we use Eq. 4 as the loss function. Therefore, the small model is trained as a mixture of standard cross-entropy and knowledge distillation.

C. Pruning

Knowledge distillation reduces model size by changing model structure. Pruning reduces model size by discarding unnecessary weights in model. Pruning gets inspiration from human brain. For newborns, synapses are created and then gradually pruned, finally grow into adults form.

Pruning consists of three steps. The first step is to train a conventional neural network. The second step removes connections which weights are below the threshold. This step converts a dense network into a sparse network. Paper [24] proposes three ways to prune weights in deep neural network. The first one sorts all parameters and then prunes the weights below global threshold. This method fails to take the difference

of parameters in different layers into account. The second method sorts the parameters of each layer and removes the weights that are smaller than the threshold of each layer. The third method also prunes the parameters of each layer, but the threshold is determined by standard deviation of the weights of that layer. This paper also suggests that the use of the first method, though the simplest, can obtain better performance. In this work, the first method is applied to pruning the deep neural network.

The third step is to retrain the sparse network. In this step, for convenience, the pruned weights are set to zero and gradients are also set to zero at the corresponding position. So we can continue training the network without changing the training structure. Multiple iterations can be obtained in the second and third step to get satisfactory performance. When performance begins to decrease, pruning is stopped. The network can be stored as a sparse matrix, which can considerably reduce the model size.

III. EXPERIMENTAL SETUP

A. Data preparation

In order to demonstrate the effectiveness of proposed method of model compression, extensive experiments have been performed. In this work, HKUST [28] Mandarin Chinese conversation telephone corpus (LDC2005S15, LDC2005T32) is used. It contains 150-hour speech, 873 calls in training set, 15-hour speech, 81 calls in development set and 5-hour speech, 24 calls in test set. In our study, the development set is used to control the learning rate and choose the model. The experimental results are all evaluated on the test set.

B. Training preparation

Speech pre-processing and feature extraction are the same as those of general LVCSR [10]. Firstly, speech signal is divided into frames. For each frame, acoustic features are generated based on 40-dimensional log-mel filterbank features and 3-dimensional pitch features [29] along with their first and second order derivatives. In this experiment, 5 past frames and 5 future frames are appended to current frame, constituting a total of 1419-dimensional feature vector. The alignments are generated by a well-trained GMM-HMM system with 2848 senones. A trigram language model is trained on all transcription texts of the training set. The models are all trained on Tensorflow [30] and decoded by Kaldi [31].

IV. RESULTS

A. Layer normalization

At the beginning of this experiment, a state-of-the-art DNN is firstly trained, and layer normalization is applied before non-linear function in each layer. In order to decide the appropriate architecture, four kinds of DNN are conducted. The results are shown in Table I.

DNN-6-1024 and DNN-6-2048 are DNN with 6 hidden layers and 1024, 2048 nodes in each hidden layer respectively. Similarly, DNN-6-2048-LN represents DNN with 6 hidden

TABLE I
Comparison of systems with different architecture.

Model	CER (%)
DNN-6-1024	40.76
DNN-6-1024-LN	40.16
DNN-6-2048	39.83
DNN-6-2048-LN	39.32

layers and 2048 nodes in each hidden layer, and layer normalization is applied before the non-linearity of each layer.

As can be seen from the Table I, layer normalization in DNN-6-1024-LN and DNN-6-2048-LN can effectively improve the performance. About 0.6% decrease of CER is obtained. The number of parameters of DNN-6-2048-LN is almost 4 times as DNN-6-1024-LN. For computation efficiency, DNN-6-1024-LN is chosen as the teacher model for subsequent model compression.

B. Knowledge distillation

The training process of knowledge distillation is done in two stages.

In the first stage, the architecture of student model needs to be decided. The student model must be carefully chosen as a trade-off between model size and model performance. Thus the student model can acquire knowledge as much as possible, while the number of parameters should also be within reasonable size. Four kinds of DNN are applied to comparing the learning ability. Besides, temperature T and tuned parameter λ are not considered in this stage. The results of experiment are shown in Table II. In the following table, the compression rate represents the ratio of the model parameters before compression to the model parameters after compression.

TABLE II
Comparison of learning ability of small models.

Model	CER (%)	Compression Rate
DNN-6-1024-LN	40.16	—
DNN-3-800	43.08	2.04×
DNN-3-512	43.75	3.55×
DNN-3-400	44.14	4.74×
DNN-3-256	47.41	7.84×

DNN-6-1024-LN is the teacher model, which only provides the soft targets for student model. DNN-3-800, DNN-3-512, DNN-3-400 and DNN-3-256 are candidate small models for acquiring knowledge, which denote DNN with 3 hidden layers and 800, 512, 400, 256 nodes in each hidden layer respectively. Learned from the results, it is obvious that DNN-3-800 has the lowest CER, but the compression rate is only 1.82. For DNN-3-400 and DNN-3-256, though the number of parameters is small, the performance decrease significantly. Therefore, considering the trade-off between parameter and performance, DNN-3-512 is chosen as the student model for the next stage.

In the second stage, temperature T and tuned parameter λ are taken into account. Temperature can introduce more information of error classes to small model, and the information of the ground truth labels can be obtained by using λ . These methods can help to explore the maximums of knowledge that small model can acquire. The results are shown in Table III.

TABLE III
CER (%) of the student models with varied hyper-parameters.

$\lambda \backslash T$	1	2	5	10
0	44.27	43.72	44.76	45.05
0.2	42.75	42.10	43.60	44.52
0.4	43.22	43.13	43.20	43.53
0.6	42.95	42.49	43.76	43.48
0.8	42.57	43.08	43.87	43.03
1	43.75	43.17	44.08	44.78

Compared with DNN-3-512 in Table II, it is obvious that the use of T and λ can effectively improve performance. When $T = 2$ and $\lambda = 0.2$, DNN-3-512 can reach the best performance with the CER of 42.10%. The best model is denoted as DNN-3-512-KD. DNN-3-512 with $T = 1$ and $\lambda = 0$ is denoted as DNN-3-512-Ref, which is trained only by using ground truth labels. The experimental results of knowledge distillation are shown in Table IV.

TABLE IV
Results of knowledge distillation.

Model	CER (%)	Parameters (million)	Compression Rate
DNN-6-1024-LN	40.16	9.62	—
DNN-3-512-KD	42.10	2.71	3.55×
DNN-3-512-Ref	44.27	2.71	3.55×

In Table IV, knowledge distillation can successfully transfer knowledge from DNN-6-1024-LN to DNN-3-512-KD. The number of parameters is reduced by a factor of 3.55, from 9.62 million to 2.71 million. DNN-3-512-KD suffers 2% performance loss. At the same time, compared with DNN-3-512-Ref, DNN-3-512-KD can achieve 2% improvement with the same model size.

C. Combined knowledge distillation and pruning

Furthermore, pruning is applied to compressing DNN-3-512-KD. In this experiment, pruning is divided into two steps. The first step is to select threshold. If threshold is too large, the model can not recover from pruning. If threshold is too small, the number of iterations will increase. Therefore, it is important to select an appropriate threshold. Different thresholds are adopted to directly prune the model. After pruning, retraining is conducted to recover from performance loss.

In Table V, when threshold is bigger than 0.1, DNN-3-512-KD can not recover from weight reduction. Performance de-

TABLE V
Results of pruning with different threshold.

Model	Threshold	CER (%)	Compression Rate
DNN-3-512-KD	—	42.10	—
DNN-3-512	0.1	42.24	2.29×
DNN-3-512	0.2	44.07	3.42×
DNN-3-512	0.4	48.88	6.57×
DNN-3-512	0.6	57.88	12.57×

TABLE VI
Results of compression method

Model	CER (%)	Parameters (million)	Storage Size	$T_{forward}$ (second)
DNN-6-1024-LN	40.16	9.62	37M	165.8369
DNN-3-512-KD	42.10	2.71	11M	93.7702
DNN-3-512-Ref	44.27	2.71	11M	93.7702
DNN-3-512-Pr	41.96	0.66	7.4M	94.8652

grades significantly when the threshold gets bigger. Therefore, in the following pruning, threshold is initially selected as 0.1.

The second step is to prune DNN-3-512-KD. Weights are directly removed if their values are smaller than the threshold, which converts a dense network to a sparse network. Then retraining is applied to recovering from performance loss. By re-selecting the threshold and iterating the steps above, the model can be effectively compressed. Pruning is stopped until the model can not recover from weight reduction. In our work, the threshold is gradually increased by 0.05 for every three rounds. After 10 rounds of iterative training, the performance of the model begins to decrease. Therefore, we stop pruning and store the model as DNN-3-512-Pr. Pruned models are saved as compressed row format (CSR) with low bit index, and sparse matrix multiplication is unused for pruned models.

The results are summarized in Table VI. Because we do not compress language model, we only evaluate the feed-forward time of acoustic model on test set. In Table VI, $T_{forward}$ represents feed-forward time. The models are tested on CPU. Compared with DNN-3-512-KD, the number of parameters of DNN-3-512-Pr is compressed by 4.11. The storage size of DNN-3-512-Pr is reduced by 1.48. The performance of DNN-3-512-Pr improves slightly. Compared with DNN-6-1024-LN, the CER of DNN-3-512-Pr increases 1.8%. The number of parameters is compressed by a factor of 14.59, from 9.62 million to 0.66 million. The storage size is reduced by 5, from 37M to 7.4M. The feed-forward time is reduced by 1.77. What's more, DNN-3-512-Pr, compared with DNN-3-512-Ref, achieves a 2% performance improvement, meanwhile the number of parameters is reduced by a factor of 4.11.

D. Weight distribution

In pruning, we directly remove the weights which are below the global threshold. Therefore, we are interested in exploring the weight distribution of DNN-3-512-Pr.

TABLE VII
Distribution of weights in DNN-3-512-Pr

Layer	Parameters	Compression Rate
DNN-Aff-1	21411	$33.93\times$
DNN-Aff-2	40212	$6.52\times$
DNN-Aff-3	80545	$3.25\times$
DNN-Output	517451	$2.82\times$
Total	659619	$4.11\times$

In Table VII, DNN-Aff-1, DNN-Aff-2, DNN-Aff-3 and DNN-Output are affine transform matrices of the first hidden layer, the second hidden layer, the third hidden layer and output layer respectively. It is obvious that DNN-3-512-Pr has more weights in DNN-Output than other layers. That is because the weights in DNN-Output have greater value than the others. We can also say that the weights closer to the output layer play more important role in the network.

V. CONCLUSIONS AND DISCUSSIONS

This paper presents a new scheme of model compression, which can effectively compress the acoustic model in speech recognition system. In detail, we integrate knowledge distillation and pruning to compress acoustic model, and layer normalization is applied to improving the performance. The results show that layer normalization can effectively improve CER by 0.6%. By using knowledge distillation, DNN-6-1024-LN can be compressed to DNN-3-512-KD, with 2% performance loss. The number of parameters is reduced by a factor of 3.55, from 9.62 million to 2.71 million. After pruning, DNN-3-512-KD can be compressed to DNN-3-512-Pr, with a slight performance improvement. The number of parameters is reduced by a factor of 4.11, from 2.71 million to 0.66 million. In summary, the acoustic model, DNN-6-1024-LN, can be effectively compressed to DNN-3-512-Pr, which achieves $14.59\times$ parameters reduction, $5\times$ storage size reduction and $1.77\times$ feed-forward speed up, with 1.8% loss of accuracy.

This paper presents a method to compress the acoustic model in speech recognition system, which is easy to be deployed. It can achieve high compression rate with little performance loss. Since our compression method can be easily integrated and combined with other advanced techniques, it has great potential for further improvement. Firstly, although we only explore DNN structure in this work, our method can be easily and effectively extended to more powerful models such as CNN and LSTM. Secondly, our method can be combined with weight sharing and quantization to achieve higher compression rate. Thirdly, after pruning, the weights are mainly distributed in output layer. We can further compress output layer and speed up Softmax computations. At the same time, the feed-forward time of DNN-3-512-Pr have not been improved because sparse matrix multiplication is not applied in this experiment. We can utilize sparse matrix multiplication to further reduce the feed-forward time.

VI. ACKNOWLEDGMENTS

This work is supported by National Key Research and Development Program of China under Grant No.2016YFB1001404 and Beijing Engineering Research Center Program under Grant No.Z171100002217015.

REFERENCES

- [1] A. Graves, *Supervised Sequence Labelling with Recurrent Neural Networks*. Springer Berlin Heidelberg, 2012.
- [2] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," *Computer Science*, pp. 338–342, 2014.
- [3] H. Sak, A. Senior, K. Rao, and F. Beaufays, "Fast and accurate recurrent neural network acoustic models for speech recognition," *Computer Science*, 2015.
- [4] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.
- [5] T. Sercu and V. Goel, "Advances in very deep convolutional neural networks for lvsr," *arXiv preprint arXiv:1604.01792*, 2016.
- [6] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *ICML*, vol. 14, 2014, pp. 1764–1772.
- [7] Y. Miao, M. Gowayyed, and F. Metze, "Eesen: End-to-end speech recognition using deep rnn models and wfst-based decoding," in *Automatic Speech Recognition and Understanding (ASRU), 2015 IEEE Workshop on*. IEEE, 2015, pp. 167–174.
- [8] J. Chorowski, D. Bahdanau, K. Cho, and Y. Bengio, "End-to-end continuous speech recognition using attention-based recurrent nn: First results," *arXiv preprint arXiv:1412.1602*, 2014.
- [9] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," in *Advances in Neural Information Processing Systems*, 2015, pp. 577–585.
- [10] Y. Zhao, S. Xu, and B. Xu, "Multidimensional residual learning based on recurrent neural networks for acoustic modeling," *Interspeech 2016*, pp. 3419–3423, 2016.
- [11] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Highway networks," *arXiv preprint arXiv:1505.00387*, 2015.
- [12] J. Xue, J. Li, and Y. Gong, "Restructuring of deep neural network acoustic models with singular value decomposition," in *Interspeech*, 2013, pp. 2365–2369.
- [13] 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.
- [14] Q. Le, T. Sarlós, and A. Smola, "Fastfood-approximating kernel expansions in loglinear time," in *Proceedings of the international conference on machine learning*, 2013.
- [15] V. Sindhwani, T. Sainath, and S. Kumar, "Structured transforms for small-footprint deep learning," in *Advances in Neural Information Processing Systems*, 2015, pp. 3088–3096.
- [16] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.
- [17] J. Li, R. Zhao, J.-T. Huang, and Y. Gong, "Learning small-size dnn with output-distribution-based criteria," in *INTERSPEECH*, 2014, pp. 1910–1914.
- [18] 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.
- [19] L. Lu, M. Guo, and S. Renals, "Knowledge distillation for small-footprint highway networks," *arXiv preprint arXiv:1608.00892*, 2016.
- [20] J. H. Wong and M. J. Gales, "Sequence student-teacher training of deep neural networks," 2016.
- [21] Y. Kim and A. M. Rush, "Sequence-level knowledge distillation," *arXiv preprint arXiv:1606.07947*, 2016.
- [22] D. Yu, F. Seide, G. Li, and L. Deng, "Exploiting sparseness in deep neural networks for large vocabulary speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*. IEEE, 2012, pp. 4409–4412.
- [23] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Advances in Neural Information Processing Systems*, 2015, pp. 1135–1143.

- [24] A. See, M.-T. Luong, and C. D. Manning, "Compression of neural machine translation models via pruning," *arXiv preprint arXiv:1606.09274*, 2016.
- [25] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.
- [26] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.
- [27] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.
- [28] Y. Liu, P. Fung, Y. Yang, C. Cieri, S. Huang, and D. Graff, "Hkust/mts: A very large scale mandarin telephone speech corpus," *Lecture Notes in Computer Science*, vol. 4274, pp. 724–735, 2006.
- [29] P. Ghahremani, B. BabaAli, D. Povey, K. Riedhammer, J. Trmal, and S. Khudanpur, "A pitch extraction algorithm tuned for automatic speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 2494–2498.
- [30] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.
- [31] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz *et al.*, "The kaldı speech recognition toolkit," in *IEEE 2011 workshop on automatic speech recognition and understanding*, no. EPFL-CONF-192584. IEEE Signal Processing Society, 2011.