STRUCTURAL SPARSIFICATION FOR FAR-FIELD SPEAKER RECOGNITION WITH INTEL® GNA

Jingchi Zhang[⋆] *Jonathan Huang*[†] *Michael Deisher*[◇] *Hai Li*[⋆] *Yiran Chen*[⋆]

[⋆] Duke University, Durham, North Carolina, USA
[†]Intel Corporation, Santa Clara, California USA
[◇]Intel Corporation, Hillsboro, Oregon, USA

ABSTRACT

Recently, deep neural networks (DNN) have been widely used in speaker recognition area. In order to achieve fast response time and high accuracy, the requirements for hardware resources increase rapidly. However, as the speaker recognition application is often implemented on mobile devices, it is necessary to maintain a low computational cost while keeping high accuracy in far-field condition. In this paper, we apply structural sparsification on time-delay neural networks (TDNN) to remove redundant structures and accelerate the execution. On our targeted hardware, our model can remove 60% of parameters and only slightly increasing equal error rate (EER) by 0.18% while our structural sparse model can achieve more than $1.5\times$ speedup.

***Index Terms*—** structural sparsification, far-field speaker recognition, time-delay neural network

1. INTRODUCTION

Far-field speaker recognition has gained much interest in the research community, with its prevalent applications in consumer devices such smart speakers and smartphones. The far-field condition presents additional challenges in speaker recognition, due to the severity of reverberation and background noise. Similar to automatic speech recognition, deep learning acoustic features have shown great improvements in these conditions compared to prior techniques. A number of speaker recognition systems based on deep neural network (DNN) embeddings have been reported in the literature [1][2][3]. More recently, SRI developed the VOiCES dataset [4] specifically for far-field speaker recognition, and showed their DNN embeddings significantly outperformed the i-vector systems [5].

The objective of our work is to develop a speaker recognition system robust to the far-field channel conditions, using advanced model training methodology. Furthermore, we designed the system to be simple to perform inference using ultra-low power accelerators such as the Intel® GNA [6]. In contrast to the popular approach of using probabilistic linear discriminant analysis (PLDA) [7] in the back-end, our system only relies on the simple cosine distance for scoring. This allows for the computations to be performed end-to-end on the accelerator. Finally, to get the best model size efficiency, the crux of the paper will focus on the application of structural sparsification to our DNN model. Applying suitable sparse granularity on the model could reduce the latency of the model inference and improve the performance of the real-time speech recognition.

There have been extensive studies on accelerating DNN models. Pruning [8] and sparsity methods [9] can effectively reduce the size of CNN models while keeping the performance similar to the original models. However, randomly distributed zeros in models do not have benefit for execution on hardware. [10] elaborates the benefit of structural sparsity over non-structural sparsity on locality and parallelism during hardware execution. To force zero parameters to form a regular arrangement, structural sparsity [11] is proposed for CNNs to learn sparse structures like channel and filter.

In order to reduce the model size and the inference time, we apply a structural sparsity learning method to speaker recognition models. The sparse structure we achieve is computationally friendly to specific hardware. Specifically, we add a group Lasso [12] penalty to the loss function, where the group is the structure desired to be sparse. The sparse model performance is the same or even better compared to the baseline with fewer non-zero parameters. Also, we test our method on three different sparse granularity levels and found that under the same number of non-zero parameters, models with smaller granularity achieve lower equal error rate (EER) than models with larger granularity. Sparse model performance exceeds that of dense models regardless of the granularity with the same number of non-zero parameters.

2. RELATED WORK

Computational acceleration methods have been heavily explored for the past years. Pruning and sparsification have proven effective at removing redundant parameters and structures. In [8][13], pruning connections of fully connected layers was proved effective at reducing the size of *Alexnet* and

VGG-16. However, most of the computation and parameters are from convolution layers. From this perspective, Wei *et al.* [11] propose a framework that can reduce model size by eliminating redundant structures in CNNs such as filters or channels. They claimed to achieve 3.1× speedup on *Alexnet* on GPU while keep the accuracy the same.

For speech recognition tasks, recurrent neural network (RNN) and long short-term memory (LSTM) models are widely used. It is more difficult to learn sparse structures for these models because the structures usually contain information on time sequences. Eliminating those structures would have more impact on performance. Narang *et al.* [14] conducted Connection Pruning for RNNs and reduced 90% of connections. Wei *et al.* [15] further applied group Lasso regularization on LSTMs and achieved 10.59× speedup without perplexity loss. Zhang *et al.* [10] also extended the structural sparsity learning method to LSTM models for speech recognition and removed 72.5% parameters with negligible accuracy loss.

3. METHODOLOGY

3.1. Model topology

This work is based the x-vector model structure [3], with some simplifications. Compared to the original x-vector model, our architecture, shown on Table 1, has increased the input feature dimension from 24 to 40, reduced the pooling dimension from 1500 to 512, removed a fully-connected layer between the embedding and speaker output layers, and reduced the embedding dimension from 512 to 256. In our testing, these modifications did not degrade recognition performance and had much lower complexity. We use this topology as the baseline for structural sparsity learning. Also, in this particular case, TDNN can be written as a one-dimension convolution, so we implemented the model as a 5-layer CNN.

The softmax output is only used for model training purposes; for speaker enrollment and verification, the DNN embedding is taken at the output of Segment6 on Table 1. One speaker embedding is computed for an entire utterance, regardless of length. We use cosine distance of this 256-dimension embedding vectors between enrollment and test utterances to produce the speaker recognition score.

3.2. Loss function

While the conventional softmax loss works reasonably well for training speaker embeddings, it is specifically designed for classification, not verification tasks. Speaker recognition systems trained with softmax loss typically use PLDA in the backend to improve separation between speakers. The triplet loss function, which is designed to reduce intra-speaker and increase inter-speaker distance, has shown to be more effective for speaker recognition [2]. Likewise, the end-to-end loss [1] has better performance than softmax. The downside to these kinds of losses is that the training infrastructure is significantly more complicated than one used for supervised learning with softmax. In a prior study [16], we explored the use of several recently proposed loss functions that were first introduced in face recognition research. These loss functions are drop-in replacements for softmax, thus modification to training code is simple with little overhead in training speed. We found Additive Margin Softmax (AM-softmax) [17] to perform best in the far-field test set, and incorporating PLDA did not improve performance against the simpler cosine distance. The elimination of the PLDA in the inference pipeline makes the entire model easy to deploy to target hardware, with the help of tools such as the Intel® Distribution of OpenVINO$^{\text{TM}}$ toolkit [18].

Table 1: Model configuration

	layer context	Affine	Convolution
Layer1	[t-2,t+2]	200×512	512 40×5
Layer2	{t-2,t,t+2}	1536×512	512 512×3
Layer3	{t-2,t,t+2}	1536×512	512 512×3
Layer4	{t}	512×512	512 512×1
Layer5	{t}	512×512	512 512×1
Stats pooling	[0,T)	512T×1024	N/A
Segment6	{0}	1024×256	N/A
Softmax	{0}	256×N	N/A

N denotes the number of training speakers.

3.3. Training details

We describe our training pipeline as a three step process:

1. **Baseline model training:** We find that we get significantly better results when we start the sparsification process with a well-trained dense model. We train the model with AM-softmax loss, SGD optimizer learning rate decaying from 0.01 to 0.0001 in 30 epochs with cosine annealing. The weight decay and batch size are set to 1e-6 and 256, respectively. For each batch, we select random segments of training utterances between 2.5 to 3.0 seconds. These settings, except for the number of epochs, are used in subsequent steps. The output of this step is the best dense model we can produce, and it also serves as a baseline to measure EER against.

2. **Learning sparse structure:** We use the model from step 1 to initialize the dense model, and trained 20 epochs with the group Lasso regularization together with the AM-softmax loss:

$$E(w) = E_D(w) + \lambda \cdot \sum_{k=1}^{K} ||w_k||_2, \qquad (1)$$

where the first term $E_D(w)$ is the original AM-softmax loss function, and the second term is the contribution from the group Lasso loss function. The group Lasso loss is essentially the summation of K (the total number of groups) L2 norm of group weights w_k in predefined groups (e.g. chunks of 8 or 16, or entire convolution filter). It rewards to total loss function for forcing low values to group weights. The coefficient λ controls the balance between AM-softmax loss and group Lasso loss. This step produces sparse structures by training on the new loss function $E(w)$. Groups with L2 values below a threshold are set to 0, and discarded in the learning process for the next step.

3. **Fine-tuning:** Lastly, we fine-tune the training for 20 epochs on the sparse model produced by step 2 using only AM-softmax loss.

More detail on step 2 and step 3 can be found in [10].

4. EXPERIMENTAL SETUP

4.1. Datasets and augmentation

We use VoxCeleb 1 and 2 [19] [20] to train the system. These datasets have 7323 identities combined. We perform 9x data augmentation plus original clean speech to produce 12.7 million training utterances. For each data augmentation, we randomly choose from 2000 room impulse responses generated from Pyroomacoustics [21], and add randomly selected background noise from MUSAN [22] and AudioSet [23]. For the test set, we used the VOiCES far-field dataset [4], which we believe captures the essence of challenging channel conditions. For all speech utterances, we use 40-dimension log-mel filterbanks, with 3-second sliding window mean subtraction.

4.2. Hardware implementation

This work is targeting TDNN inference on the Intel® Gaussian & Neural Accelerator (GNA) [24]. Intel® GNA is designed for continuous inference with neural networks on edge devices with high performance and very low power consumption. Since Intel® GNA fetches weight matrices in 16-byte chunks of `int8` or `int16` weights, we investigated structural sparsity on chunks of 8 `int16` elements or 16 `int8` elements. Inference measurements were made on an Intel® Celeron® Processor J4005 with Intel® GNA inside.

5. RESULTS

In the experiments, the sparsity of filters is defined as the number of zero filters over all filters, while the sparsity of chunks is defined as the the number of zero chunks over all chunks. We applied the sparsity learning only to layers 1-4. Our experiments showed that layers 5 and above were reluctant to achieve sparsity. We suspect that this is because near the output of the network, the hidden representations contain high density of information for speaker recognition. This seems to happen at the input of the stats pooling layer.

5.1. Result analysis

The experimental results are shown in Table 2. We applied the structural sparsity on filters and chunks. Filter sparsity can be deployed on all hardware without any special modification. While applying sparsity on chunk-8 and chunk-16 are targeted at Intel® GNA. Also, we run experiments on dense models to compare the performance of sparse models and dense models.

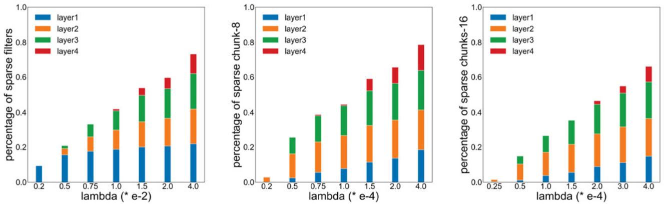
Figure 1 is the visualization of the relationship between the coefficient λ and sparsity in each layer. Y-axis denotes the overall percentage of sparsity in four layers. It is shown clearly when λ increases, the sparsity increases. However, the sparsity growth in each layer is different. Filter sparsity shows a different growth trend from chunk sparsity. In Figure 1(a), sparse filters in the first layer (blue bar) account for much of the overall sparsity. However, in Figure 1(b) and (c), the first layer is not very sparse while layers 2 and 3 have a majority of chunks learned to be zero. We suspect that the low sparsity in layer 1 is due to the denser spectral input dimension compared to other layers; and that in layer 4 the output representation is becoming more relevant for the speaker recognition task, thus having making the network sparse here would result in higher penalty on the AM-softmax loss.

Figure 2 is the visualization of the relationship of non-zero parameters and EER or min detection cost function (minDCF) at $P_{target} = 0.01$ (consistent with the VOiCES evaluation protocol). The X-axis represents the number of non-zero parameters and Y-axis is the EER and minDCF. We compared the filter sparsity, chunk-8 sparsity, and chunk-16 sparsity with dense models of different sizes. It is shown in Figure 2(a) that when the number of parameters is large, sparse models achieve lower EER than dense models of the same size. However when the number of non-zero parameters is small, dense models have better performance. In our experimental setting, the turning point is around 0.7 million parameters. For example, with EER around 2.0%, it is clear that models with smaller granularity have lower size. Chunk-8 models can reach 1.99% EER with 0.99 million parameters and chunk-16 has 1.96% EER under 1.07 million parameters. Comparing with baseline, smaller dense models reach 2.03% EER with 1.73 million parameters, chunk-8 and chunk-16 both reach lower EER with less than 60% of the parameters. Also, when non-zero parameter count is larger than 1.5 million, there is a tendency that chunk-8 has the best performance while filter sparsity has higher EER under the same non-zero parameter count. As for the relationship of minDCF, as is shown in Figure 2(b), we observe similar patterns as seen in EER.

A somewhat surprising finding in these results is that, filter_1, chunk8_1, and chunk16_1 with less parameters have slightly lower EER, 1.76%, 1.61%, and 1.68%, respectively, compared to the baseline of 1.81%. We believe this is because the group Lasso loss is an effective regularizer, and when used

Table 2: Model performance with different sparsity levels

Method	λ	size	EER	min	Method	λ	size	EER	min
dense		(M)	(%)	DCF	*filter*	(e-2)	(M)	(%)	DCF
baseline	-	2.47	1.81	0.23	baseline	-	2.47	1.81	0.23
dense_1	-	1.73	2.03	0.25	filter_1	0.2	2.14	1.76	0.22
dense_2	-	1.42	2.12	0.25	filter_2	0.5	1.70	1.88	0.24
dense_3	-	1.15	2.20	0.27	filter_3	0.75	1.27	2.07	0.26
dense_4	-	0.91	2.44	0.30	filter_4	1	1.04	2.24	0.27
dense_5	-	0.70	2.51	0.30	filter_5	1.5	0.79	2.49	0.31
dense_6	-	0.54	2.79	0.35	filter_6	2	0.69	2.55	0.31
dense_7	-	0.41	3.62	0.41	filter_7	4	0.50	3.49	0.38
Method	λ	size	EER	min	Method	λ	size	EER	min
chunk16	(e-4)	(M)	(%)	DCF	*chunk8*	(e-4)	(M)	(%)	DCF
baseline	-	2.47	1.81	0.23	baseline	-	2.47	1.81	0.23
chunk16_1	0.25	2.28	1.68	0.22	chunk8_1	0.2	2.29	1.61	0.21
chunk16_2	0.5	1.73	1.86	0.24	chunk8_2	0.5	1.33	1.93	0.25
chunk16_3	1	1.33	1.90	0.25	chunk8_3	0.75	0.99	1.99	0.27
chunk16_4	1.5	1.07	1.96	0.26	chunk8_4	1	0.85	2.29	0.28
chunk16_5	2	0.84	2.28	0.29	chunk8_5	1.5	0.65	2.57	0.33
chunk16_6	3	0.70	2.49	0.32	chunk8_6	2	0.57	3.10	0.36
chunk16_7	4	0.56	3.13	0.37	chunk8_7	4	0.43	3.62	0.42



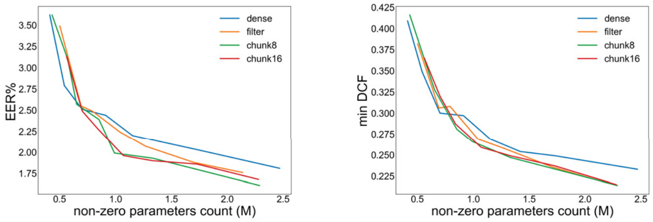
(a) *filter* (b) *group-8* (b) *group-16*

Fig. 1: Relationship between λ and structural sparsity

in a small dose, helps produce more generalized models.

5.2. Measurements on Intel® GNA

We also measured the actual inference time to find out how much speedup sparse models could achieve on Intel® GNA. We measured all the models we get in table 2 and found the relationship between the hardware speedup and EER. As is shown in Figure 3, four lines represent three different sparse granularity and one dense model and they all start at the baseline point. Generally, dense models always have higher EER when speedups are the same, which confirms our expectation. It means that under the same EER, structural sparse models are always faster than the dense model. It is also important to point out that when speedup is small, around 1.2× speedup, sparse models have speedup even with lower EER, which is a free-meal. However, there may exist some oscillation when measuring the inference time in hardware so the results may not be precise. This may explain why sparse models show no benefit when speedup is around 1.9× and the trend is consistent.



(a) EER (b) minDCF

Fig. 2: Visualization of EER and minDCF versus parameter count in millions.

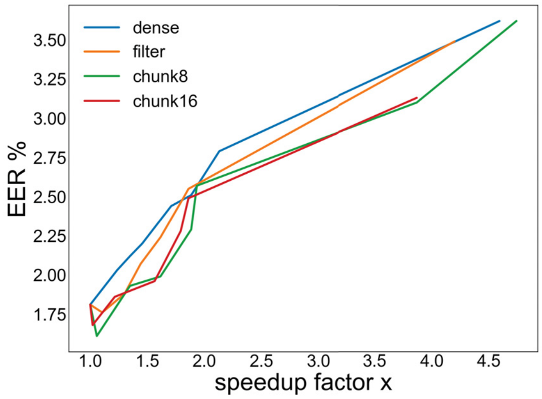


Fig. 3: Speedup on Intel® Pentium® Silver J4005

6. CONCLUSION

In this paper, we applied structural sparsification for speaker recognition models. By using pretrained models and group Lasso regularization, we kept the good performance of the original model while reducing the number of parameters and accelerating the actual execution. For structural sparse models that are only slight smaller than the full size dense model, we achieved better performance on both EER and minDCF metrics.

Acknowledgements. This work is supported by the National Science Foundation CCF-1910299.

7. REFERENCES

[1] Georg Heigold, Ignacio Moreno, Samy Bengio, and Noam Shazeer, "End-to-end text-dependent speaker verification," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 5115–5119.

[2] Chao Li, Xiaokong Ma, Bing Jiang, Xiangang Li, Xuewei Zhang, Xiao Liu, Ying Cao, Ajay Kannan, and Zhenyao Zhu, "Deep speaker: an end-to-end

neural speaker embedding system," *arXiv preprint arXiv:1705.02304*, 2017.

[3] David Snyder, Daniel Garcia-Romero, Gregory Sell, Daniel Povey, and Sanjeev Khudanpur, "X-vectors: Robust dnn embeddings for speaker recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5329–5333.

[4] Colleen Richey, Maria A Barrios, Zeb Armstrong, Chris Bartels, Horacio Franco, Martin Graciarena, Aaron Lawson, Mahesh Kumar Nandwana, Allen Stauffer, Julien van Hout, et al., "Voices obscured in complex environmental settings (voices) corpus," *arXiv preprint arXiv:1804.05053*, 2018.

[5] Mahesh Kumar Nandwana, Julien van Hout, Mitchell McLaren, Allen R Stauffer, Colleen Richey, Aaron Lawson, and Martin Graciarena, "Robust speaker recognition from distant speech under real reverberant environments using speaker embeddings.," in *Interspeech*, 2018, pp. 1106–1110.

[6] G. Stemmer, M. Georges, J. Hofer, P. Rozen, J. Bauer, J. Nowicki, T. Bocklet, H. R. Colett, O. Falik, M. Deisher, and S. J. Downing, "Speech recognition and understanding on hardware-accelerated DSP," in *Proc. Interspeech*, 2017, pp. 2036–2037.

[7] Sergey Ioffe, "Probabilistic linear discriminant analysis," in *European Conference on Computer Vision*. Springer, 2006, pp. 531–542.

[8] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural networks," in *Advances in Neural Information Processing Systems*, 2015.

[9] B. Liu, M. Wang, H. Foroosh, M. Tappen, and M. Pensky, "Sparse convolutional neural networks," in *The IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[10] Jingchi Zhang, Wei Wen, Michael Deisher, Hsin-Pai Cheng, Hai Li, and Yiran Chen, "Learning efficient sparse structures in speech recognition," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 2717–2721.

[11] W. Wen, C. Wu, Y. W, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," in *Advances in Neural Information Processing Systems*, 2016.

[12] Ming Yuan and Yi Lin, "Model selection and estimation in regression with grouped variables," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 68, no. 1, pp. 49–67, 2006.

[13] Song Han, Huizi Mao, and William J Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[14] S. Narang, G. Diamos, S. Sengupta, and E. Elsen, "Exploring sparsity in recurrent neural networks," *arXiv:1704.05119*, 2017.

[15] W. Wen, Y. He, S. Rajbhandari, W. Wang, F. Liu, B. Hu, Y. Chen, and H. Li, "Learning intrinsic sparse structures within long short-term memory," *arXiv:1709.05027*, 2017.

[16] Jonathan Huang and Tobias Bocklet, "Intel Far-Field Speaker Recognition System for VOiCES Challenge 2019," in *Proc. Interspeech 2019*, 2019, pp. 2473–2477.

[17] Feng Wang, Jian Cheng, Weiyang Liu, and Haijun Liu, "Additive margin softmax for face verification," *IEEE Signal Processing Letters*, vol. 25, no. 7, pp. 926–930, 2018.

[18] "Openvino toolkit," `https://docs.openvinotoolkit.org/`, Accessed: 2019-10-14.

[19] Arsha Nagrani, Joon Son Chung, and Andrew Zisserman, "Voxceleb: a large-scale speaker identification dataset," *arXiv preprint arXiv:1706.08612*, 2017.

[20] Joon Son Chung, Arsha Nagrani, and Andrew Zisserman, "Voxceleb2: Deep speaker recognition," *arXiv preprint arXiv:1806.05622*, 2018.

[21] Robin Scheibler, Eric Bezzam, and Ivan Dokmanić, "Pyroomacoustics: A python package for audio room simulation and array processing algorithms," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 351–355.

[22] David Snyder, Guoguo Chen, and Daniel Povey, "MUSAN: A Music, Speech, and Noise Corpus," 2015, arXiv:1510.08484v1.

[23] Jort F Gemmeke, Daniel PW Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R Channing Moore, Manoj Plakal, and Marvin Ritter, "Audio set: An ontology and human-labeled dataset for audio events," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 776–780.

[24] M. Deisher and A. Polonski, "Implementation of efficient, low power deep neural networks on next-generation intel client platforms," *http://sigport.org/1777*, 2017.