



Compressing Deep Neural Networks using a Rank-Constrained Topology

Preetum Nakkiran¹, Raziël Alvarez², Rohit Prabhavalkar², Carolina Parada²

¹Department of EECS, University of California, Berkeley, USA

²Speech Group, Google Inc., Mountain View, USA

preetum@berkeley.edu, {raziel, prabhavalkar, carolinap}@google.com

Abstract

We present a general approach to reduce the size of feed-forward deep neural networks (DNNs). We propose a rank-constrained topology, which factors the weights in the input layer of the DNN in terms of a low-rank representation: unlike previous work, our technique is applied at the level of the filters learned at individual hidden layer nodes, and exploits the natural two-dimensional time-frequency structure in the input. These techniques are applied on a small-footprint DNN-based keyword spotting task, where we find that we can reduce model size by 75% relative to the baseline, without any loss in performance. Furthermore, we find that the proposed approach is more effective at improving model performance compared to other popular dimensionality reduction techniques, when evaluated with a comparable number of parameters.

Index Terms: deep neural networks, low-rank approximation, keyword spotting, embedded speech recognition

1. Introduction

Deep neural networks (DNNs) have recently emerged as an attractive model for many learning tasks; they offer great representational power, without demanding much feature engineering. Computational advances and the availability of large datasets have made it feasible to train large DNNs, with millions of parameters [1]. DNNs have become a popular foundation for state-of-the-art automatic speech recognition (ASR) systems [2], with numerous successful applications in domains such as large vocabulary continuous speech recognition (LVCSR) [3, 4, 5, 6, 7, 8] and keyword spotting (KWS) tasks [9, 10].

When used as acoustic models in speech recognition, DNNs estimate output posterior probabilities for individual speech units (e.g., context-dependent states, context-independent phones, syllables or words). In recent work, we proposed a DNN-based KWS system [9], which requires a small memory-footprint, thus allowing it to run efficiently on embedded devices. To improve performance, adjacent speech frames are typically stacked together to provide temporal context to the DNNs. This results in a considerable growth in the number of input-to-hidden layer weights, which can be a limitation for models targeted towards embedded devices. For example, the embedded LVCSR system described in [5] contains about 13% of its parameters in the input layer, and this number goes up to 80% for the embedded DNN-based KWS system in [9].

Previous work on reducing the number of independent parameters in a DNN model without significantly affecting performance includes [7], which proposed to *zero-out* a subset of DNN weights that are below a certain threshold and the “optimal brain damage” procedure [11] which proposed to *zero-*

out weights based on the second-derivative of the loss function. Other techniques have been proposed which change the DNN architecture, e.g., through the use of bottle-neck layers [12, 13], or through a low-rank matrix factorization of the weights in the final layer of the DNN [14].

We propose a scheme to compress an existing fully-trained DNN using a low-rank approximation of the weights *associated with individual nodes in the first hidden layer* by means of a rank-constrained DNN layer topology. This allows us to significantly reduce the number of independent parameters in the model, without any loss in performance. Our approach is similar to previous work on learning separable filters [15, 16] which is used in image processing for speeding up computation in convolution neural networks (see, e.g., [17, 18] and the references contained therein). Our work is also similar to the work proposed in [14] – which uses a low-rank factorization of the weight matrix between the hidden and output layers – but differs fundamentally in the way the low-rank approximation is obtained and enforced. In particular, the techniques described in [14] are applied to the hidden-to-output layer weights with the motivation of speeding up DNN training rather than compressing an existing model. Further, as noted in [14], their technique was only found to be effective when applied to the final layer weights in the DNN but not to other intermediate hidden layers.¹ Instead, our technique is applied at the level of individual filters learned at nodes of the first hidden layer in the DNN, which as we show in Section 2 are inherently of low-rank. Our technique is particularly effective in models with a large number of parameters in the input-to-hidden layer weight matrices, e.g., DNN-based KWS models [9] with large input context windows. Finally, since the compression is “built-into” the topology, it removes the need for a separate and potentially expensive decompression step during network evaluation.

In Section 2, we describe the motivation behind the proposed approach, and show how it can be applied to compress a DNN model. We apply the proposed techniques to the DNN-based KWS system presented in [9] that is briefly reviewed in Section 3. We describe the results of our experimental evaluations in Section 4 where we compare against baseline DNNs and the low-rank approach of [14]. Finally, we conclude with a discussion of our findings in Section 5.

2. Rank Constrained Neural Network

We denote an input utterance as, $\mathcal{X} = [\mathbf{X}_1, \dots, \mathbf{X}_T]$, consisting of T input frames of speech, where each frame $\mathbf{X}_t = [x_{t,1}, \dots, x_{t,d}] \in \mathbb{R}^d$. In our experiments, \mathbf{X}_t corresponds to log-mel filter-bank energies, so d corresponds to the number

¹The authors in [14] do not report the effectiveness of the technique when applied to the input-to-hidden-layer weights.

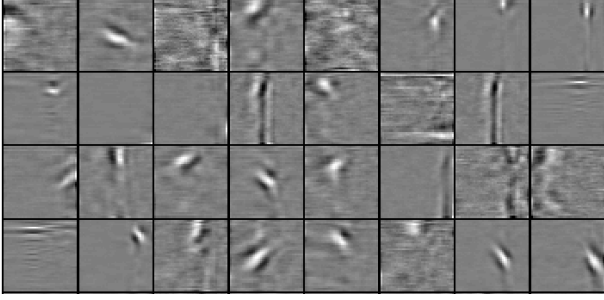


Figure 1: Visualization of a subset of filters learned in our baseline DNN (baseline-240K; see Section 3.1). Each rectangle represents the weights, $W^{(m)}$, associated with a single node in the first hidden layer, wrapped such that time/frequency are horizontal/vertical. Intensities encode weight values, such that larger (positive) weights appear as white and smaller (negative) weights appear black.

of filter-bank channels. We denote the input to the DNN corresponding to frame t by \mathbf{x}_t , which is formed by stacking together T_l and T_r adjacent frames of left- and right-context as is typical for DNN-based AMs, $\mathbf{x}_t = [\mathbf{X}_{t-T_l}, \dots, \mathbf{X}_{t+T_r}]$, where $C = T_l + T_r + 1$ denotes the total number of context frames input to the network. Finally, we assume a fully-connected structure for the input-to-hidden layer weights of the DNN. Thus, each node m in the first hidden layer, computes its output activation $a_t^{(m)}$ as,

$$a_t^{(m)} = f \left(\sum_{i=0}^{C-1} \sum_{j=1}^d w_{i,j}^{(m)} x_{(t-T_l+i),j} \right) \quad (1)$$

where, $f(\cdot)$ denotes a non-linear activation function, and $W^{(m)} = [w_{i,j}^{(m)}]$ denotes the weights connecting the inputs to a particular node m of the first hidden layer, which we interpret as a matrix of size $C \times d$, corresponding to the time-frequency structure in the input \mathbf{x}_t .

2.1. Motivation

The weights, $W^{(m)}$, corresponding to the first hidden layer of the DNN, act as low-level feature detectors, and can thus be considered as “filters” of the input signal. These filters – learned as part of DNN training procedure – tend to have simple structure, which makes them amenable to compression. This can be observed, by examining the filters that are learned from the training data in our baseline DNN (baseline-240K; see Section 3). Each rectangle in Figure 1 visualizes an individual filter as a 2D image, created by wrapping the weights vector into the matrix $W^{(m)} = [w_{i,j}^{(m)}]$, as described in Equation 1. We may consider the operation of each filter as overlaying its weights over the input time-frequency representation of the speech; thus, intense black or white regions correspond to regions of the input that strongly influence the output activation of a particular node. Note that the filters in Figure 1 are indeed highly structured – many of them are simply vertical lines (corresponding to activity across frequencies, at a particular time), or have only small concentrated patches of activity. This leads to our key observation: by considering the hidden layer weights $W^{(m)}$, as a structured weight matrix as described in Equation 1, we can compress it significantly through a low-rank approximation. This is further confirmed by computing the explained vari-

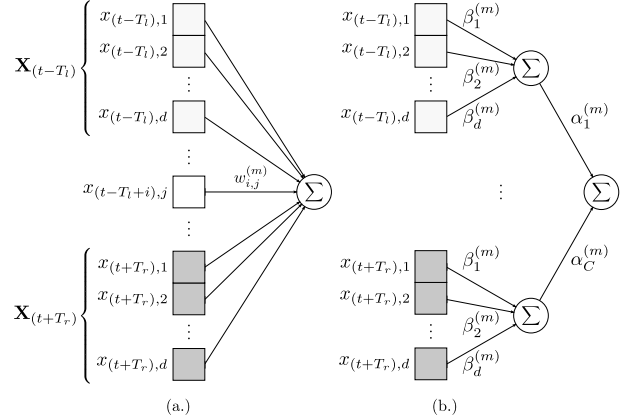


Figure 2: Topology comparison: (a.) Baseline DNN. (b.) Rank-constrained topology of rank-1. The rank-constraint can be encoded into the network by introducing an intermediate linear layer with tied weights, which connects to a unit that computes the “outer” summation in (2) and applies the original non-linear activation function.

ance² of our baseline DNN (baseline-240K – see Section 3.1): more than 97% of the total variance in the original filters is explained by retaining just the top 5 singular values of each filter, which rises to 99% for a rank-10 approximation.

2.2. Rank-Constrained Topology

As described in Section 2.1, the filters learned by the DNN can be well approximated by equivalent filters of low-rank. In order to ensure that the network learns low-rank filters directly (instead of filters of arbitrary rank), we introduce a rank-constrained layer topology for the input-to-first-hidden layer weights, which encodes the low-rank constraint into the network itself. We begin by describing the approach for rank-1 filters, which is then generalized to rank- k filters in Section 2.2.2.

2.2.1. Rank-1 Filters

We approximate the weights $W^{(m)}$ as rank-1 filters by factoring the weights in terms of two vectors $\alpha^{(m)} \in \mathbb{R}^C$ and $\beta^{(m)} \in \mathbb{R}^d$ as,

$$w_{i,j}^{(m)} \approx \alpha_i^{(m)} \beta_j^{(m)} \quad (2)$$

Note that (2) can be interpreted as performing a mix of selectivity in time ($\alpha^{(m)}$) with selectivity in frequency ($\beta^{(m)}$). Thus, we can re-write (1) using (2) as:

$$a_t^{(m)} \approx f \left(\sum_{i=0}^{C-1} \alpha_i^{(m)} \sum_{j=1}^d \beta_j^{(m)} x_{(t-T_l+i),j} \right) \quad (3)$$

This rank-1 approximation can be encoded in the network by introducing an intermediate layer with a linear activation function, and weight tying as illustrated in Figure 2.³

²Ratio of total amount of variance in the low-rank approximation, to that in the original: $\frac{\sum_{r=1}^k \sigma_r^2}{\sum_{r=1}^{\min(C,d)} \sigma_r^2}$, where σ_r is the top r -th singular value in the singular value decomposition (SVD) of $W^{(m)}$.

³In practice, the computation in (3) can be sped-up by caching intermediate terms $\gamma(k) = \sum_{j=1}^d \beta_j^{(m)} x_{k,j}$ which appear in the “inner” summation of (3).

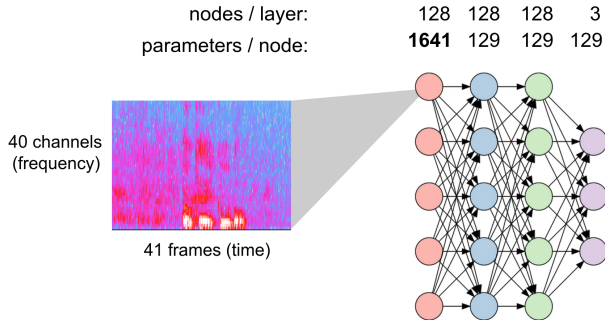


Figure 3: The baseline KWS DNN [9, 10].

2.2.2. Generalization to Rank- k Filters

Since rank- k matrices are the sum of rank-1 matrices, we can encode rank- k filters as sums of k rank-1 filters, parameterized by vectors $\alpha^{(m),r} \in \mathbb{R}^C$ and $\beta^{(m),r} \in \mathbb{R}^d$:

$$w_{i,j}^{(m)} \approx \sum_{r=1}^k \alpha_i^{(m),r} \beta_j^{(m),r} \quad (4)$$

$$a_i^{(m)} \approx f \left(\sum_{r=1}^k \sum_{i=0}^{C-1} \alpha_i^{(m),r} \sum_{j=1}^d \beta_j^{(m),r} x_{(t-T_l+i),j} \right) \quad (5)$$

Thus, a rank- k topology for a single hidden layer node can be encoded in the network by k copies of the rank-1 topology. In practice, this sum is equivalently performed by connecting all the linear intermediate layers ($\beta^{(m),1}, \dots, \beta^{(m),k}$) of the copies into the same final node, which then has kC parameters.

2.3. Initializing parameters of Rank- k Topology

The parameters $\alpha^{(m),r}, \beta^{(m),r}$ of the rank- k topology, are learned from the input data; we initialize them either randomly, or from a fully-trained baseline DNN by performing a singular value decomposition (SVD) on each matrix $W^{(m)}$ and retaining the left and right singular vectors corresponding to the top k singular values. We note that this is different from the methods proposed in [14], since we decompose weights corresponding to each node in the first hidden layer independently. The left singular vectors (scaled by the matrix of singular values) correspond to $\alpha^{(m),r}$, and the right singular vectors correspond to $\beta^{(m),r}$. Furthermore, since the rank-constraint is implicit in the topology itself (as seen in Figure 2), we can leverage traditional DNN training methods to learn the weights in the model.

2.4. Compression Achieved using a Rank- k Topology

When a rank- k topology is applied to the baseline-DNN, this reduces the number of independent parameters *per-filter* from $|W^{(m)}| = Cd$ to $(C+d)k$, which results in a significant reduction for small k . For example, in our experiments, $C = 41$, $d = 40$ and $k = 5$, resulting in a compression ratio of about 4x.

3. Experimental Setup

In order to determine the effectiveness of the proposed techniques, we evaluate the effectiveness of the rank-constrained topologies on the DNN-based keyword spotting system proposed in [9], which we review briefly in this section.

3.1. Baseline System

Our baseline system (baseline-240K) is based on the system in [9], which consists of a DNN trained to predict individ-

System	Initialization	Total Parameters
baseline-240K	random	243,072
baseline-84K	random	83,619
rc-init	SVD of baseline-240K	85,379
rc-noinit	random	85,379
low-rank [14]	random	102,019

Table 1: Description of the systems used in our experiments.

ual word targets within a given keyword phrase. The input to the DNN consists of 40-dimensional log-mel filter-bank energies ($d = 40$), with $T_l = 30$ frames of left and $T_r = 10$ frames of right-context ($C = 41$). The baseline DNN consists of 3 fully-connected hidden layers with 128 nodes each, with a rectified linear unit (ReLU) activation function on each of the hidden layer nodes [19, 20]. The final softmax output layer predicts individual word targets corresponding to the keyphrase. The (word) labels for each frame are determined by a forced-alignment using a large LVCSR system [4]. The system is trained to optimize a cross-entropy criterion using asynchronous stochastic gradient descent (ASGD) implemented in the large-scale DNN training infrastructure Dist-Belief [21].

During evaluation, a score for each utterance is computed over sliding windows in the utterance using the modified keyword scoring function proposed in [10].

3.1.1. Experimental Systems

All of our DNN systems use the same input features – formed by stacking 41 frames of 40-dimensional features – as described in Section 3.1.

We consider an additional DNN baseline, wherein we reduce the total number of system parameters by reducing the number of hidden layer nodes: we use 3 fully-connected hidden layers, with 48 nodes in each, so that the total number of system parameters is approximately 84K (baseline-84K).

We also consider a system (low-rank) based on the low-rank matrix factorization approach proposed in [14]. This system factorizes the input-to-hidden-layer weights in terms of two DNN layers: the DNN inputs are fully connected to a low-rank layer of size 48 with a linear activation function, which is then fully connected to a hidden layer with 128 nodes. These are followed by two additional fully connected hidden layers with 128 nodes each.

Performance is compared against a system that uses the proposed rank-constrained topology of rank-5 (determined based on the analysis presented in Section 2.1). We consider two versions of this system: either initialized by computing an SVD of the input-to-hidden-layer weights of baseline-240K as described in Section 2.3 (rc-init) or by initializing the weights randomly (rc-noinit); in either case, the system weights are learned on the data. A description of all experimental systems used in this paper appears in Table 1.

3.2. Datasets

We evaluate the proposed approach on the same training, development, and evaluation sets as in our previous work [10]. We train KWS systems for fourteen phrases comprising various voice actions.⁴ The entire dataset consists of about 10K–15K utterances containing each of the keyword phrases, and a

⁴Specifically: “answer call”, “decline call”, “email guests”, “fast forward”, “next playlist”, “next song”, “next track”, “pause music”, “pause this”, “play music”, “set clock”, “set time”, “start timer”, “take note”.

much larger set of 396K utterances that do not contain any of the phrases, which are randomly partitioned into training, development and evaluation sets, so that each evaluation set contains about 2K positive and 35K negative examples per-phrase.

Our systems are trained on multi-style training data created by adding car and cafeteria noise to the clean training data at an SNR in the range [-5dB, +10dB] as detailed in [10]. We report system performance by plotting receiver operating characteristic (ROC) curves on three evaluation sets representing clean, and noisy conditions (*lower curves are better*). In addition to the ‘clean’ evaluation set (clean), we create two noisy sets by adding car noise at -5dB (car_-5dB) and cafeteria noise at +5dB (cafe_5dB).⁵ Details of how these datasets were created can be found in [10].

4. Results

Our goal is to determine whether we can reduce the number of independent parameters in the DNN without any loss in system performance. As can be seen from the results presented in Figure 4, baseline-85k which reduces model size by reducing the number of hidden layer nodes in each of the three hidden layers performs significantly worse than the larger 240k parameter baseline in all cases, by about 25% relative across the range of false alarms (FAs). The proposed rank-constrained topology (rc_init), with 75% fewer independent parameters than baseline-240k (from which it is initialized) performs as well or better than baseline-240k across the entire range of FA rates for all of the evaluation sets and outperforms baseline-85k by a large margin. We hypothesize that the improvements in performance seen in rc_init over baseline_240k are due to the fact that the filters learned in the rank-constrained topology are significantly smoother and simpler than their baseline counterparts, and may therefore be less sensitive to the effect of noisy training data.

Furthermore, the proposed rank-constrained system with random initialization, which enforces the rank-constraint during training (rc_noinit) outperforms the system which uses a low-rank matrix factorization [14] although it has fewer independent parameters (85k in rc_noinit vs. 102k in low-rank). Additionally, performance can be improved further through initialization based on an SVD as seen by the improvement in performance between rc_init over rc_noinit.

Overall, the rank-constrained topology proposed in this work allows us to reduce the number of parameters in the model by 75% relative to the baseline, *without any loss in performance*.

5. Conclusions

We presented a technique for reducing the number of independent parameters in a DNN system, that we term a ‘rank-constrained’ topology. Our approach is motivated by the observation that the filters learned at individual nodes in the first hidden layer have significant structure and are thus amenable to compression. In experimental evaluations, we find that the proposed rank-constrained topology allows us to reduce the number of independent parameters by 75% relative to our baseline system, without any loss in performance, and is more effective than previously proposed techniques.

⁵We also evaluated performance on the far-field sets (clean_100cm) and (car_-5db_100cm) defined in [10], where we observed similar trends as those reported in Section 4. Therefore, we do not report results on the far-field sets here in the interest of space.

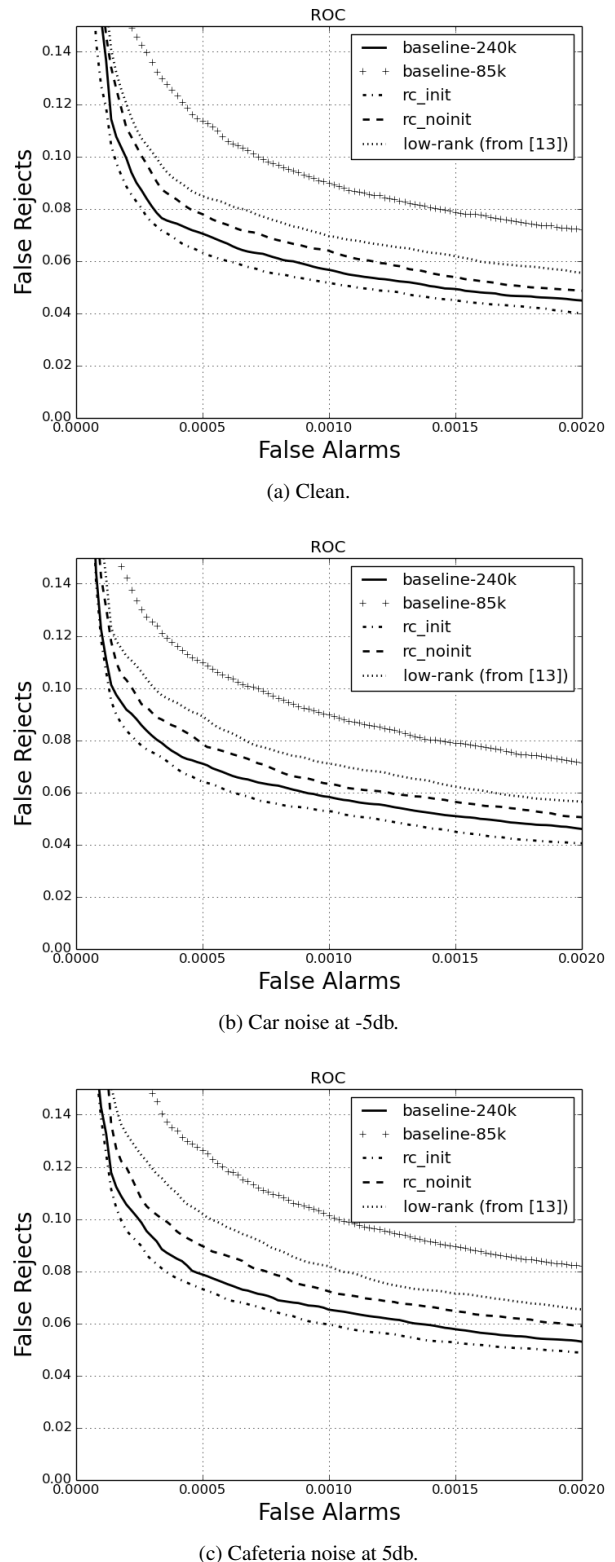


Figure 4: ROC curves comparing performance of our experimental systems described in Section 3.1.1 on the three evaluation sets: clean, car_-5db, and cafe_5db, averaged over all of the fourteen keywords. Values on axes indicate the fraction of utterances that are incorrectly classified. Lower curves indicate better performance.

6. References

- [1] L. Deng and D. Yu, "Deep learning: Methods and applications," *Foundations and Trends in Signal Processing*, vol. 7, no. 3–4, pp. 197–387, 2014.
- [2] L. Deng, G. Hinton, and B. Kingsbury, "New types of deep neural network learning for speech recognition and related applications: An overview," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, May 2013, pp. 8599–8602.
- [3] G. Dahl, T. N. Sainath, and G. Hinton, "Improving deep neural networks for lvcsr using rectified linear units and dropout," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, May 2013, pp. 8609–8613.
- [4] N. Jaitly, P. Nguyen, A. Senior, and V. Vanhoucke, "Application of pretrained deep neural networks to large vocabulary speech recognition," in *Proceedings of Annual Conference of the International Speech Communication Association (Interspeech)*, 2012, pp. 2578–2581.
- [5] X. Lei, A. Senior, A. Gruenstein, and J. Sorensen, "Accurate and compact large vocabulary speech recognition on mobile devices," in *Proceedings of Annual Conference of the International Speech Communication Association (Interspeech)*, 2013, pp. 662–665.
- [6] G. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pretrained deep neural networks for large-vocabulary speech recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 1, pp. 30–42, Jan 2012.
- [7] F. Seide, G. Li, and D. Yu, "Conversational speech transcription using context-dependent deep neural networks," in *Proceedings of Annual Conference of the International Speech Communication Association (Interspeech)*, 2011, pp. 437–440.
- [8] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *Signal Processing Magazine, IEEE*, vol. 29, no. 6, pp. 82–97, Nov 2012.
- [9] G. Chen, Parada, C., and G. Heigold, "Small-footprint keyword spotting using deep neural networks," in *Proceedings of IEEE International conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2014, pp. 4087–4091.
- [10] R. Prabhavalkar, R. Alvarez, C. Parada, P. Nakkiran, and T. N. Sainath, "Automatic gain control and multi-style training for robust small-footprint keyword spotting with deep neural networks," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, (to appear), 2015.
- [11] Y. LeCun, J. S. Denker, and S. A. Solla, "Optimal brain damage," *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, vol. 2, pp. 598–605, 1990.
- [12] T. N. Sainath, B. Kingsbury, and B. Ramabhadran, "Auto-encoder bottleneck features using deep belief networks," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, march 2012, pp. 4153–4156.
- [13] F. Grézl and P. Fousek, "Optimizing bottle-neck features for LVCSR," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2008, pp. 4729–4732.
- [14] T. 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 *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2013, pp. 6655–6659.
- [15] S. Treitel and J. L. Shanks, "The design of multistage separable planar filters," *IEEE Transactions on Geoscience Electronics*, vol. 9, no. 1, pp. 10–27, Jan 1971.
- [16] R. Rigamonti, A. Sironi, V. Lepetit, and P. Fua, "Learning separable filters," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013, pp. 2754–2761.
- [17] F. Mamalet and C. Garcia, "Simplifying convnets for fast learning," in *Proceedings of International Conference on Artificial Neural Networks (ICANN)*, 2012, pp. 58–65.
- [18] M. Jaderberg, A. Vedaldi, and A. Zisserman, "Speeding up convolutional neural networks with low rank expansions," *arXiv preprint arXiv:1405.3866*, 2014.
- [19] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of International Conference on Machine Learning (ICML)*, 2010, pp. 807–814.
- [20] M. D. Zeiler, M. A. Ranzato, R. Monga, M. Mao, K. Yang, Q. V. Le, P. Nguyen, A. Senior, V. Vanhoucke, J. Dean, and G. E. Hinton, "On rectified linear units for speech processing," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013, pp. 3517–3521.
- [21] J. Dean, G. S. Corrado, R. Monga, K. Chen, M. Devin, Q. V. Le, M. Z. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, and A. Y. Ng, "Large scale distributed deep networks," in *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, 2012, pp. 1223–1231.