Reducing the Model Order of Deep Neural Networks Using Information Theory

Ming Tu[1], Visar Berisha[1,2], Yu Cao[2], Jae-sun Seo[2],
[1] Speech and Hearing Science Department, Arizona State University
[2] School of Electrical, Computer, and Energy Engineering, Arizona State University

***Abstract*—Deep neural networks are typically represented by a much larger number of parameters than shallow models, making them prohibitive for small footprint devices. Recent research shows that there is considerable redundancy in the parameter space of deep neural networks. In this paper, we propose a method to compress deep neural networks by using the Fisher Information metric, which we estimate through a stochastic optimization method that keeps track of second-order information in the network. We first remove unimportant parameters and then use non-uniform fixed point quantization to assign more bits to parameters with higher Fisher Information estimates. We evaluate our method on a classification task with a convolutional neural network trained on the MNIST data set. Experimental results show that our method outperforms existing methods for both network pruning and quantization.**

I. Introduction

Deep neural networks (DNNs) have been shown to outperform shallow learning algorithms in applications such as computer vision [1], [2], automatic speech recognition [3], [4] and natural language processing [5], [6]; however DNNs also have large parameter sets, often making them prohibitive for small-footprint devices [7]. For example, while the original LeNet5 network [8] (a classification system based on convolutional neural network) has less than 100K parameters, the winner of the 2012 ImageNet competition [9] has over 60M parameters. The memory access costs alone can make these larger networks unsuitable for low-power settings.

It has been posited that the expressive power of DNNs comes from their large parameter spaces and hierarchical structure; however recent studies have shown that there is often a great deal of parameter redundancy in DNNs [10], [11], making them unnecessarily complex. As a result, reducing the complexity of DNNs has been an area of great interest to the research community in recent years. For example, the authors in [12], [13] used low rank decomposition of the weights to reduce the parameter set and applied this method to a DNN-based acoustic model and to convolutional neural networks (CNN) for image classification. Similarly, the authors in [11] showed that over 95% parameters of DNNs can be predicted without any training and without impacting accuracy.

In addition to low-rank parameter decomposition, network pruning and quantization methods have also been proposed [14]. Neural network pruning has been investigated in early studies, including pruning weights with small magnitudes, optimal brain damage [15] and optimal brain surgeon [16]. The last two methods require estimation of the Hessian matrix of network parameters to decide on their importance; however, the sizes of existing networks make the estimation of this large matrix prohibitive. As a result, for large-scale DNNs, magnitude-based weight pruning is still a popular method [17], [14], [18].

For fixed-point implementations of DNNs, parameter quantization is also required. The studies in [19], [20] discretized the weights of a neural network according to the range of the weights. The methods in [21] and [22] used uniform scalar parameter quantization to implement fixed-point versions of the networks. In [23], a new fixed-point representation for DNN training was proposed, using stochastic rounding for the parameters. Vector quantization based schemes have been applied to CNNs for both computer vision and automatic speech recognition tasks [24], [25], [26].

In this paper, we propose a new method that ranks the parameters of a DNN for both network pruning and parameter quantization. We investigate an information-theoretic approach to reduce the DNN parameter space by using the *Fisher Information* as a proxy for parameter importance. The Fisher criterion is a natural metric for quantifying the relative importance of DNN parameters since it provides an estimate of how much information a random variable carries about a parameter of the distribution. In [27], we introduced a new method to calculate the diagonal of the Fisher Information Matrix (FIM) and showed that it can be used to reduce the size of DNNs. In this paper, we extend this work by using a lower-complexity estimate of the FIM diagonal and evaluating the technique on a much larger network. We validate the method on the MNIST dataset using a CNN and show that our method results in smaller networks with fewer parameters to quantize at a lower bit rate.

The remainder of this paper is organized as follows: In the next section, we introduce our network pruning and quantization scheme. In section III, we validate the algorithm on the MNIST data. We end the paper with a discussion of the results in section IV and concluding remarks in section V.

II. DNNs pruning and quantization

A. Fisher Information and DNNs

We show a notional DNN architecture in Fig. 1. Let us consider the output $\mathbf{y}$ of the DNN as a conditional probability distribution $p(\mathbf{y}|\mathbf{x};\boldsymbol{\theta})$ parameterized by the DNN input, $\mathbf{x}$, and its parameters, $\boldsymbol{\theta}$. The FIM evaluated at a particular value of $\boldsymbol{\theta}$ is defined as:

978-1-4673-9039-2/16 $31.00 © 2016 IEEE
DOI 10.1109/ISVLSI.2016.117

IEEE computer society

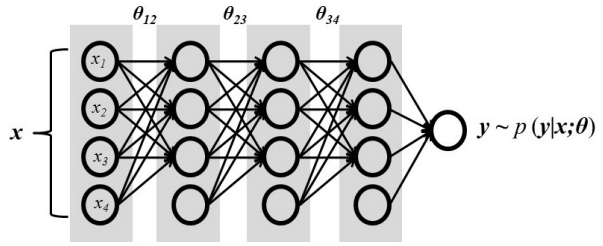

Fig. 1. A notional DNN architecture with input $\mathbf{x}$, output $\mathbf{y}$, and parameter vector $\boldsymbol{\theta}$.

$$\mathbf{F}(\boldsymbol{\theta}) = \mathrm{E}_{\mathbf{y}}\left[\left(\frac{\partial \log p(\mathbf{y}|\mathbf{x};\boldsymbol{\theta})}{\partial\, \boldsymbol{\theta}}\right)\left(\frac{\partial \log p(\mathbf{y}|\mathbf{x};\boldsymbol{\theta})}{\partial\, \boldsymbol{\theta}}\right)^{T}\right]. \tag{1}$$

We can see from eqn. (1) that the FIM is the covariance of the gradient of log likelihood with regards to its parameter $\boldsymbol{\theta}$. Thus, $\mathbf{F}_{\boldsymbol{\theta}}$ is a $p \times p$ symmetric positive semidefinite matrix.

It is easy to see that the diagonal elements of the FIM can be calculated by the expectation of the element-wise multiplication of the gradient:

$$\mathbf{F}_D(\boldsymbol{\theta}) = \mathrm{E}_{\mathbf{y}}\left[\mathbf{g} \odot \mathbf{g}\right], \tag{2}$$

where $\mathbf{g} = \frac{\partial \log p(\mathbf{y}|\mathbf{x};\boldsymbol{\theta})}{\partial\, \boldsymbol{\theta}}$ is the gradient of the log-likelihood and $\odot$ represents element-wise multiplication; $\mathbf{F}_D(\boldsymbol{\theta})$ is a $p \times 1$ vector, each element of which is the Fisher Information of a corresponding parameter.

The Fisher Information provides an estimate of the amount of information a random variable carries about a parameter of the distribution. In the context of a DNN, this provides a natural metric for quantifying the relative importance of any given parameter in the network. The less information an output variable carries about a parameter, the less important that parameter is to the output statistics of the network. As a result, we assume that removing parameters with low entries in $\mathbf{F}_D(\boldsymbol{\theta})$ will not greatly affect the output of the network. That is precisely the approach we take in this paper - we will rank the parameters in a DNN based on their corresponding entries in the FIM diagonal. In the ensuing section, we describe the method we use for approximating $\mathbf{F}_D(\boldsymbol{\theta})$.

B. Estimating the Fisher Information

A number of recent studies on natural gradient descent (NGD) [28] exploit the information geometry of the underlying parameter manifold and apply it to gradient-based optimization of DNNs. Natural gradient descent uses the inverse FIM to constrain the magnitude of the update steps such that the Kullback-Leibler (KL) divergence between the output distribution of the network at iteration t and iteration $t+1$ is constant [29], [30], [31]. This approach avoids large update steps and results in faster convergence.

Algorithm 1 Adam algorithm, excerpted from [32]

Require: step size α**, exponential decay rates** β_1**,** β_2**,** ϵ
Given initial parameter vector $\boldsymbol{\theta}_0$**, initial first and second moment vectors** $\boldsymbol{m}_0 \leftarrow 0$ **and** $\boldsymbol{v}_0 \leftarrow 0$ **and initial timestep** t
While $\boldsymbol{\theta}_t$ **not converged do:**
1. $t \leftarrow t+1$
2. $\boldsymbol{g}_t \leftarrow \nabla_{\boldsymbol{\theta}} f_t(\boldsymbol{\theta}_{t-1})$ **(Get gradients)**
3. $\boldsymbol{m}_t \leftarrow \beta_1 \boldsymbol{m}_{t-1} + (1-\beta_1)\boldsymbol{g}_t$
4. $\boldsymbol{v}_t \leftarrow \beta_2 \boldsymbol{v}_{t-1} + (1-\beta_2)\boldsymbol{g}_t \odot \boldsymbol{g}_t$
5. $\hat{\boldsymbol{m}}_{\boldsymbol{t}} \leftarrow \boldsymbol{m}_t/(1-\beta_1^t)$
6. $\hat{\mathbf{F}}_D(\boldsymbol{\theta}_t) \leftarrow \boldsymbol{v}_t/(1-\beta_2^t)$
7. $\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} - \alpha \hat{\boldsymbol{m}}_{\boldsymbol{t}}/(\hat{\mathbf{F}}_D(\boldsymbol{\theta}_t) + \epsilon)$
end while
return $\boldsymbol{\theta}_t$**,** $\hat{\mathbf{F}}_D(\boldsymbol{\theta}_t)$

For classification problems, DNNs are trained by minimizing the cross-entropy loss function:

$$f(\boldsymbol{\theta}) = -\sum_{i=1}^{N}\sum_{k=1}^{C} \mathbb{1}(y^{(i)} = k) log(p(y^{(i)}|\mathbf{x}^{(i)};\boldsymbol{\theta})), \tag{3}$$

where N is the number of training samples, C is the number of classes, $y^{(i)}$ is the true label of i^{th} sample $\mathbf{x}^{(i)}$ and $\mathbb{1}\{\cdot\}$ is the indicator function. A recent paper proposed a new stochastic optimization method called "Adam" and showed we can efficiently estimate the FIM diagonal at each iteration while minimizing this loss function [32]. Similar to NGD, Adam uses the approximated FIM diagonal to adapt to the geometry of the data. As a result, in this study, we use Adam to train our DNN classification system. The details of the parameter update scheme for Adam are shown in algorithm 1. As the algorithm shows, after the training algorithm converges, it returns both the optimal $\boldsymbol{\theta}_t$ and the approximated Fisher Information $\hat{\mathbf{F}}_D(\boldsymbol{\theta}_t)$. We should note that Adam is not the only choice as the optimizer because standard stochastic gradient descent can also be used; however this would require some other means of estimating $\hat{\mathbf{F}}_D(\boldsymbol{\theta}_t)$.

C. Network Pruning and Quantization

The simplest approach to network pruning is to rank the parameters by comparing their entries in the FIM diagonal and removing the ones with the lowest entries. However, as we will see in the results section, this method does not work well since estimating small values in the FIM diagonal is challenging and unreliable. When the model is over-parameterized, the actual parameter space is much smaller than the number of parameters used in the network. As a result, after training, a number of parameters become close to zero and estimating their influence based on the Fisher Information is challenging [27]. To address this problem we use a combination of magnitude-based and FIM-based pruning. For example, if we want to prune L parameters from the network, we first remove $L(1-r)$ parameters with the smallest magnitude. Then, we rank the remaining parameters based on their FIM diagonal

entries and remove the additional Lr parameters with the smallest entries in the FIM diagonal. The parameter r is between 0 and 1 and can be optimized using cross-validation.

After network pruning, we want to quantize the remaining parameters with the lowest bit representation possible. After removing L parameters, we rank the remaining $p - L$ parameters by comparing their entries in the FIM diagonal and then apply k-means clustering to separate the parameters into several groups. We quantize groups with higher Fisher Information values using more bits and groups with lower Fisher Information using fewer bits.

III. Experiments and Results Analysis

In this section, we present the experiments and results in two parts: network pruning and network quantization. All the experiments were done using the Python neural network library Keras [33] implemented using Theano on an NVIDIA GTX 760 GPU.

We evaluated our algorithms on the MNIST digits data set, which consists of 60K binary images for training and 10K for testing. There are 10 classes (digits from 0 to 9) in the data and the size of each image (and the input dimension of the neural network) is 28×28. We trained a convolutional neural network (CNN) with 2 convolutional layers each with 32 filters. The size of the convolutional kernel was 3×3 and the rectified linear unit (ReLU) activation function was used. There was a 2×2 max-pooling layer following the two convolutional layers with 0.25 dropout probability. Before the output layer, there was a fully connected layer with 128 nodes with ReLU activations and 0.5 dropout probability. The output layer had 10 nodes with softmax activations.

The loss function used to train the network was the categorical cross-entropy shown in eqn. (3). We used Adam as the optimizer with the following settings 1: batch size = 256, number of epochs = 50, step size α = 0.001, β_1 = 0.9, β_2 = 0.999, $\epsilon = 1\text{E}-8$. The accuracy of this model on the MNIST classification task without any pruning and quantization was 99.29%.

Below we describe the performance of the proposed algorithm for both pruning and quantization tasks. While we focus on CNNs in this section, our method is in no way restricted to only CNNs or only classification networks. Indeed, our probabilistic interpretation of the DNN output makes the methodology applicable across all network types, provided that the Fisher Information can be accurately estimated. Since the fully connected layers of CNNs accounts for $\sim$ 90% of the total weights [34], we only focus on the weights (including bias terms) in the fully connected layers in this paper as others have done in [24] [25].

A. Network pruning

The trained network consisted of a total of 591,242 parameters in the fully connected layers. We removed different numbers of parameters using three different methods: (1) magnitude-based pruning where parameters with the smallest magnitude were removed; (2) Fisher Information based pruning where parameters with small entries in the FIM diagonal were removed; and (3) a combination of magnitude and Fisher Information based pruning, where we traded off between the two methods using the parameter r (see Sec. II-C). The number of pruned parameters ranged from 1.0E4 (1.69% of the total parameters) to 5.9E5 (99.79% of the total parameters) with a step size 2.0E4. For the third method, we fixed the r value to 0.05. On this network we found that r values below 0.1 yield good results; however for other networks cross-validation could be used to identify an appropriate value of r.

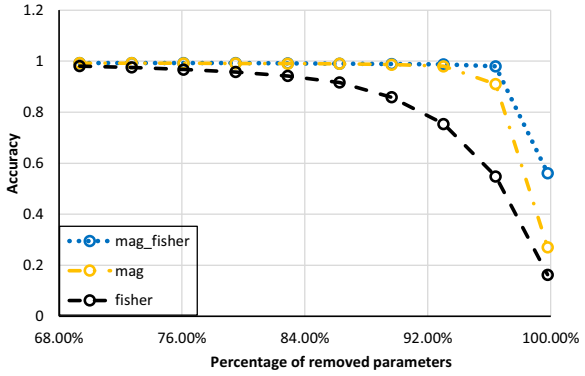

Fig. 2. Accuracy change with increasing percentage of removed parameter.

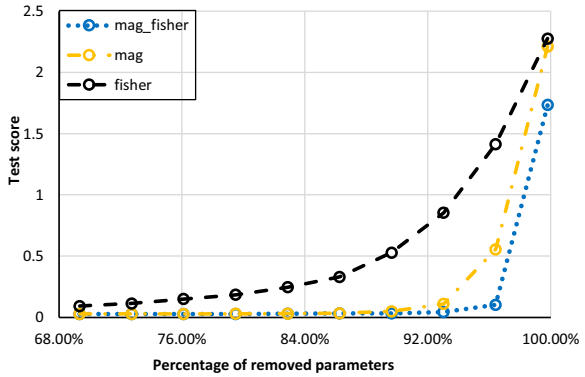

Fig. 3. Test score change with increasing percentage of removed parameter.

After removing unimportant parameters in the network, we evaluated the results on the test data and noted both the accuracy of the model and test scores (loss function evaluated on test data) as different numbers of parameters were removed. Since there was no obvious accuracy drop until 4.1E5 parameters were removed (69.35% of the total parameters), we only show the accuracy and test score starting with 69.35% parameters removed. The results are shown in Fig. 2 and 3. In Fig. 2, we show the accuracy of the model on the test data as we remove an increased number of parameters; In Fig. 3, we show the same plot, but with the test score instead of the

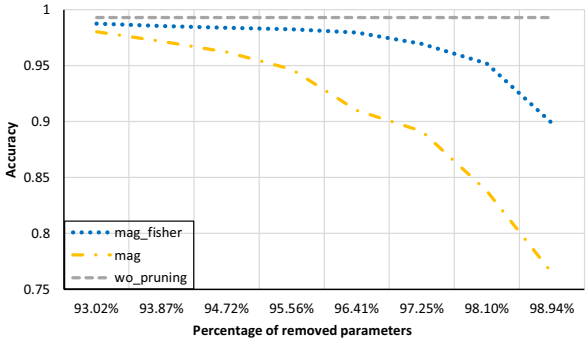

Fig. 4. Comparison between proposed pruning method and magnitude based pruning with finer scale.

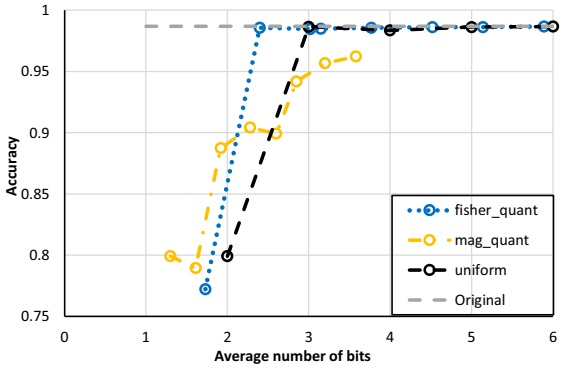

Fig. 5. Scatter plot with line to show pairs of accuracy and average number of bits of different quantization methods.

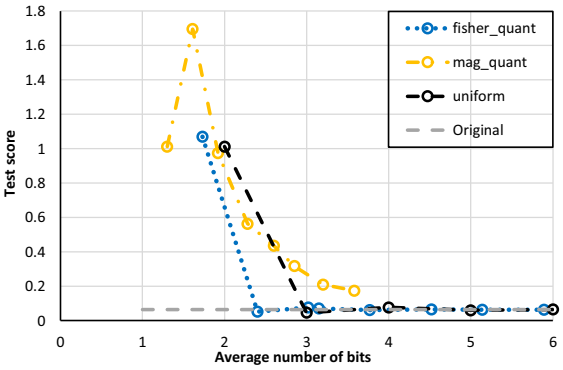

Fig. 6. Scatter plot with line to show pairs of test score and average number of bits of different quantization methods.

accuracy. In these figures, "mag" represents magnitude based pruning, "fisher" represents Fisher Information based pruning and "mag_fisher" represents a combination of magnitude and Fisher Information based pruning.

As Fig. 2 shows, as additional parameters are removed, the accuracy of the model eventually decreases. We find that using only Fisher Information based pruning, the accuracy of the model decreases quickly. This is because estimating the FIM for small parameter values is difficult as explained in Sec. II-C. This is consistent with our finding in [27], where we used the FIM criterion to remove parameters in an autoencoder. Using magnitude-based pruning, there is a clear drop-off in performance after 5.5E5 parameters (93.02% of the total parameters) are removed; however, by using our combination of magnitude and Fisher Information pruning there is no obvious accuracy drop until 5.7E5 parameters (96.41% of the total parameters) are removed. The advantage of the combined method shows that about 2.0E4 more parameters (3.38% of the total parameters) can be removed compared to magnitude based pruning with minimal impact on model performance. The test score in Fig. 3 follows the same trend as the accuracy plot in Fig. 2.

To further highlight the differences in performance between "mag" and "mag_fisher", we zoom in at the point where the accuracy starts to decrease by running the experiment with a smaller step size. These findings are shown in Fig. 4, where we show the accuracy (99.29%) without any pruning. The starting point of "mag_fisher" is 98.75% while for "mag" it is 98.02%. From this result, we can more clearly see the advantage of our combined method compared to magnitude based pruning. Consistent with findings from our previous work [27], the Fisher Information better captures the importance of larger parameters when compared to magnitude-based pruning.

B. Network quantization

After pruning, the remaining parameters in the network must be quantized for fixed-point implementations. Our quantization method is a non-uniform quantization method based on k-means clustering. We rank-order the weights by an importance metric ("fisher" or "mag") and cluster them into k clusters. The clusters are then quantized using varying bit depths, from 1 bit/parameter (least important) to k bits (most important).

To remove the effects of different pruning methods, we first removed 5.4E5 parameters (91.33% of the total parameters) using magnitude-based pruning only. The resulting model had 51,242 (8.67%) parameters remaining and an overall accuracy of 98.67% (less than 1% accuracy loss). We quantized the remaining parameters using three different methods: (1) non-uniform quantization based on Fisher Information ranking; (2) non-uniform quantization based on magnitude-based ranking; and (3) uniform quantization. For methods (1) and (2), we varied the number of clusters (from 3 to 10) and estimated the accuracy of the model for each value of k.

The results are shown in Fig. 5 and Fig. 6: "fisher_quant" represents non-uniform quantization based on Fisher Information, "mag_quant" represents non-uniform quantization based on magnitude ranking, "uniform" represents uniform quantization and "Original" represents the original non-quantized model after removing 5.4E5 parameters (accuracy is 98.67%).

From the result, we see that non-uniform quantization based on the Fisher Information can achieve almost the same accuracy as the original model with only 2.4 bits/parameter. This number is 3 for uniform quantization. Non-uniform quantization based on the magnitude never achieves the accuracy

of the original model. The same pattern is seen for the test score. This shows that on this classification task, non-uniform quantization based on the Fisher Information achieves the highest compression ratio compared to non-uniform quantization based on magnitude ranking and uniform quantization.

IV. Discussion

To evaluate the compression ratio for the example shown here, we analyze the effects of both network reduction and quantization. As we previously saw in Fig. 3, if we limit our acceptable reduction in performance to 1%, then we can remove 92.18% parameters (accuracy is 98.37%) for magnitude based pruning. This results in a compression ratio of 12.8×. For our proposed combination of magnitude and Fisher Information based pruning, we can remove 94.72% parameters (accuracy is 98.38%) and the compression ratio is 18.9×.

For network quantization, we can choose between uniform or non-uniform quantization. If we assume the original parameters are saved in FLOAT32 format, as shown in Fig. 5 using uniform quantization, we can achieve a reduction of $\frac{32}{3} = 10.7\times$; by using non-uniform quantization, we can achieve a reduction of $\frac{32}{2.4} = 13.3\times$. This means that the total compression ratio can be as high as $18.9 \times 13.3 \approx 251.4$ with less than 1% accuracy loss. This is likely an overestimate of the overall compression ratio because we need additional space to store the indices of the parameters that have been removed.

There is a relationship between our method and other previous methods based on estimation of the Hessian diagonal, namely optimal brain damage [15], optimal brain surgeon [16] and our previous work [27]. The first two methods use the entries in the Hessian diagonal of the resulting cost function to identify important and unimportant parameters. These approaches are closely related to our approach when the cost function is the log-likelihood since the second derivative of log-likelihood function (Hessian) evaluated at the maximum likelihood estimate is the observed Fisher Information [35]. Our approach in [27] made use of the relationship between Fisher Information and the family of f-divergences to estimate the FIM diagonal. The principal difference between those approaches and the one we use here is scalability - the stochastic optimization method we use to estimate the FIM diagonal can be scaled to much larger network sizes.

Finally, it is important to note that further gains in performance can be obtained by retraining the network after pruning and before quantization [14]. In this study, in an attempt to isolate the effects of network reduction and quantization, we elected not to retrain the network after the fact.

V. Conclusion

In this paper, we propose a new network reduction and quantization scheme that uses a combination of the parameter magnitude and the Fisher Information as a measure of parameter importance. For network reduction, the proposed algorithm first removes parameters with small magnitude and then further reduces the network by removing additional parameters based on the Fisher Information. Following network reduction, we propose a non-uniform quantization scheme for remaining parameters based on the same Fisher criterion. The results show that the combination of network reduction and quantization results in large compression ratios. In future, our aim is to embed complexity reduction criteria in the training process instead of using it as a post-processing step.

Acknowledgment

This research was supported in part by the Office of Naval Research grant N000141410722 (Berisha), an ASU-Mayo seed grant, and a hardware grant from NVIDIA.

References

[1] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[2] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1–9.

[3] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al., "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, 2012.

[4] Alex Graves and Navdeep Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, 2014, pp. 1764–1772.

[5] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[6] Yoav Goldberg, "A primer on neural network models for natural language processing," *arXiv preprint arXiv:1510.00726*, 2015.

[7] Vinayak Gokhale, Jonghoon Jin, Aysegul Dundar, Berin Martini, and Eugenio Culurciello, "A 240 g-ops/s mobile coprocessor for deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2014, pp. 682–687.

[8] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[9] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[10] Yu Cheng, Felix X Yu, Rogerio S Feris, Sanjiv Kumar, Alok Choudhary, and Shi-Fu Chang, "An exploration of parameter redundancy in deep networks with circulant projections," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2857–2865.

[11] Misha Denil, Babak Shakibi, Laurent Dinh, Nando de Freitas, et al., "Predicting parameters in deep learning," in *Advances in Neural Information Processing Systems*, 2013, pp. 2148–2156.

[12] Jian Xue, Jinyu Li, and Yifan Gong, "Restructuring of deep neural network acoustic models with singular value decomposition.," in *INTERSPEECH*, 2013, pp. 2365–2369.

[13] Emily L Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," in *Advances in Neural Information Processing Systems*, 2014, pp. 1269–1277.

[14] Song Han, Jeff Pool, John Tran, and William Dally, "Learning both weights and connections for efficient neural network," in *Advances in Neural Information Processing Systems*, 2015, pp. 1135–1143.

[15] Yann LeCun, John S Denker, and Sara A Solla, "Optimal brain damage," in *Advances in Neural Information Processing Systems*, 1990, pp. 598–605.

[16] Babak Hassibi and David G Stork, "Second order derivatives for network pruning: Optimal brain surgeon," in *Advances in Neural Information Processing Systems*, 1993, pp. 164–171.

[17] Dong Yu, Frank Seide, Gang Li, and Li 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.

[18] 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.

[19] Fatih Köksal, Ethem Alpaydyn, and Günhan Dündar, "Weight quantization for multi-layer perceptrons using soft weight sharing," in *Artificial Neural Networks (ICANN) 2001*, pp. 211–216. Springer, 2001.

[20] Ryu Takeda, Naoyuki Kanda, and Nobuo Nukaga, "Boundary contraction training for acoustic models based on discrete deep neural networks," in *INTERSPEECH*, 2014.

[21] Xin Lei, Andrew Senior, Alexander Gruenstein, and Jeffrey Sorensen, "Accurate and compact large vocabulary speech recognition on mobile devices.," in *INTERSPEECH*, 2013, pp. 662–665.

[22] Vincent Vanhoucke, Andrew Senior, and Mark Z Mao, "Improving the speed of neural networks on cpus," in *Proc. Deep Learning and Unsupervised Feature Learning NIPS Workshop*, 2011, vol. 1.

[23] Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan, "Deep learning with limited numerical precision," *arXiv preprint arXiv:1502.02551*, 2015.

[24] Yunchao Gong, Liu Liu, Ming Yang, and Lubomir Bourdev, "Compressing deep convolutional networks using vector quantization," *arXiv preprint arXiv:1412.6115*, 2014.

[25] Guillaume Soulié, Vincent Gripon, and Maëlys Robert, "Compression of deep neural networks on the fly," *arXiv preprint arXiv:1509.08745*, 2015.

[26] Yongqiang Wang, Jinyu Li, and Yifan Gong, "Small-footprint high-performance deep neural network-based speech recognition using split-vq," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 4984–4988.

[27] Ming Tu, Visar Berisha, Martin Woolf, Jae-sun Seo, and Yu Cao, "Ranking the parameters of deep neural network using the fisher information," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, under publication, 2016.

[28] Shun-Ichi Amari, "Natural gradient works efficiently in learning," *Neural computation*, vol. 10, no. 2, pp. 251–276, 1998.

[29] Razvan Pascanu and Yoshua Bengio, "Revisiting natural gradient for deep networks," *arXiv preprint arXiv:1301.3584*, 2013.

[30] Daniel Povey, Xiaohui Zhang, and Sanjeev Khudanpur, "Parallel training of dnns with natural gradient and parameter averaging," *arXiv preprint arXiv:1410.7455*, 2014.

[31] Guillaume Desjardins, Karen Simonyan, Razvan Pascanu, et al., "Natural neural networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 2062–2070.

[32] Diederik Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[33] Franois Chollet, "Keras," https://github.com/fchollet/keras, 2015.

[34] Matthew D Zeiler and Rob Fergus, "Visualizing and understanding convolutional networks," in *Computer vision–ECCV 2014*, pp. 818–833. Springer, 2014.

[35] Bradley Efron and David V Hinkley, "Assessing the accuracy of the maximum likelihood estimator: Observed versus expected fisher information," *Biometrika*, vol. 65, no. 3, pp. 457–483, 1978.