RANKING THE PARAMETERS OF DEEP NEURAL NETWORKS USING THE FISHER INFORMATION

Ming Tu[1], *Visar Berisha*[1,2], *Martin Woolf*[3], *Jae-sun Seo*[2], *Yu Cao*[2]

[1] Speech and Hearing Science Department, Arizona State University
[2] School of Electrical, Computer, and Energy Engineering, Arizona State University
[3] Raytheon Co.

ABSTRACT

The large number of parameters in deep neural networks (DNNs) often makes them prohibitive for low-power devices, such as field-programmable gate arrays (FPGA). In this paper, we propose a method to determine the relative importance of all network parameters by measuring the amount of information that the network output carries about each of the parameters - the Fisher Information. Based on the importance ranking, we design a complexity reduction scheme that discards unimportant parameters and assigns more quantization bits to more important parameters. For evaluation, we construct a deep autoencoder and learn a non-linear dimensionality reduction scheme for accelerometer data measuring the gait of individuals with Parkinson's disease. Experimental results confirm that the proposed ranking method can help reduce the complexity of the network with minimal impact on performance.

Index Terms— deep neural networks, Fisher Information, pruning, quantization, complexity reduction, FPGA

1. INTRODUCTION

The expressive power of deep neural networks (DNNs) comes from their application of large numbers of parameters to transform data. Yet the number of these parameters can make DNNs prohibitive for low-power applications. Each weight parameter contributes (approximately) a multiply and an add to forward-pass calculations, and each must be stored in memory. A number of studies have focused on reducing the parametric complexity of DNNs through pruning, regularization, smart quantization, etc. [1, 2]. In this paper we develop a method to determine the relative importance of all network parameters by measuring the amount of information that the network output carries about each of the parameters - the Fisher Information. This ranking can then be used to remove unimportant parameters in the network or to devise quantization strategies that assign more bits to more relevant parameters.

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

A straightforward approach to reducing neural network complexity and memory demands is pruning: the removal of unimportant parameters in the network. A number of studies have approached this topic. In [3], the authors propose to trim neural networks by iteratively turning on and off hidden units and evaluating the change in network performance. Another approach is to remove weights with small magnitude [4]. In [5] and [6], the authors use the second order derivative of the objective function w.r.t. a parameter perturbation to determine irrelevant weights. These two methods involve calculating the diagonal Hessian matrix or its inverse. Perhaps most similar to our work is the natural pruning method in [7, 8]. Both methods use the Fisher Information as a means of characterizing the importance of network parameters; however, [7] and [8], make the limiting assumption that the output network distribution is conditionally Gaussian. Our method makes no such parametric assumptions.

For reducing memory demands, there are a number of studies on DNNs parameter quantization in the literature. The studies in [9, 10, 11] discretize the weights of a neural network according to the weights' ranges. The methods in [12] and [13] use uniform scalar parameter quantization to implement fixed-point versions of the networks. In [14], a new fixed-point implementation of the DNNs is proposed, with the parameters set during DNN training. Other methods use vector quantization to compress the DNNs [2, 15] .

In contrast to previous work, here we consider a more general problem than that of pruning or quantization: our aim is to develop a method to rank the parameters by their relative importance, in a way which doesn't make assumptions about the neural network output's distributional statistics. We can then use this ranking to reduce network complexity for arbitrary problems. We exploit the relationship between the family of f-divergences and the Fisher Information, along with a recently-proposed non-parametric f-divergence, to design the ranking algorithm [16]. We quantify the information in each parameter based on the perturbation strategy originally proposed to estimate the Fisher information [16]; we evaluate the approach on a dimensionality reduction task using a deep autoencoder for pruning and quantization applications.

2. THE FISHER INFORMATION AND DEEP NEURAL NETWORKS

Let us consider the notional DNN in Fig. 1 with input $\mathbf{y} \in \mathbb{R}^L$, parameters $\boldsymbol{\theta} \in \mathbb{R}^d$, and output $\mathbf{x} \in \mathbb{R}^K$. The output distribution of the network depends on the input distribution (assumed to be unknown) and on the parameters $\boldsymbol{\theta}$. We model the output distribution of the network by $p(\mathbf{x}; \boldsymbol{\theta})$ with domain $\mathbb{R}^K$ and parameterized by a multidimensional parameter $\boldsymbol{\theta}$ with domain $\mathbb{R}^d$. In the notional network in Fig. 1, $L = 4$, $K = 2$, and $d = 26$.

We consider a family of probability distribution functions, each corresponding to a different realization of the deep network, $\mathcal{D} = \{p(\mathbf{x}; \boldsymbol{\theta})\}$, where $\mathbf{x}$ is the random variable corresponding to the output of the deep network and $\boldsymbol{\theta}$ represents the weight matrices associated with the network. The $d \times d$ symmetric positive semidefinite Fisher Information matrix, $\mathbf{F} = (F_{ij})$, evaluated at a particular value of $\boldsymbol{\theta}$, is given by

$$F_{ij}(\boldsymbol{\theta}) = \mathrm{E}\left[\frac{\partial \log p(\mathbf{x}; \boldsymbol{\theta})}{\partial \theta_i} \frac{\partial \log p(\mathbf{x}; \boldsymbol{\theta})}{\partial \theta_j}\right]. \quad (1)$$

The Fisher Information Matrix (FIM) provides an estimate of how much 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 the FIM diagonal will not greatly affect the output of the network. Traditionally, estimating the FIM in this context requires complete knowledge of the underlying distribution, $p(\mathbf{x})$, and its exact dependence on the parameter $\boldsymbol{\theta}$. In the ensuing section, we describe a non-parametric method for estimating the FIM for these networks.

3. RANKING THE NETWORK PARAMETERS

We assume that we start with a fully trained DNN. Let us consider the case where the weights of this network are slightly perturbed about the optimal value of $\boldsymbol{\theta}$:

$$q = p(\mathbf{x}; \boldsymbol{\theta} + \mathbf{u}), \quad (2)$$

where $\boldsymbol{\theta} + \mathbf{u}$ is a small perturbation around $\boldsymbol{\theta}$. Amari and Cichocki showed that *any* f-divergence induces a unique information monotonic Riemannian metric, given by the FIM (Thm. 5 in [17]). Using the Taylor expansion they show that any f-divergence, $D_f(p, q)$, is related to the FIM through the asymptotic relation

$$D_f(p, q) = \frac{1}{2}\mathbf{u}^T \mathbf{F}(\boldsymbol{\theta})\mathbf{u} + o(||\mathbf{u}||^2). \quad (3)$$

Divergences in the family of f-divergences, or Ali-Silvey distances, are defined as an average of the ratio of two distributions, weighted by some function $f(t)$: $D_f(p, q) = \int f(\frac{p(\mathbf{x})}{q(\mathbf{x})}) q(\mathbf{x}) d\mathbf{x}$ [18]. Many common divergences used in statistical signal processing fall in this category, including the KL-divergence, the Hellinger distance, the total variation distance, etc. [18].

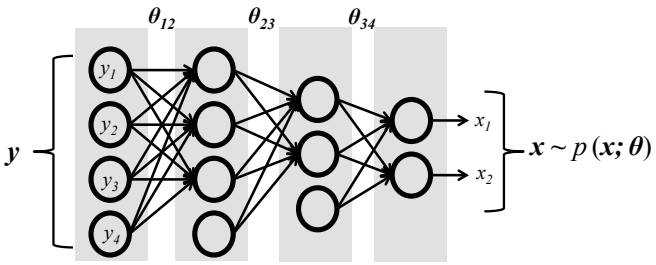

Fig. 1. A notional deep network architecture with input $\mathbf{y}$, output $\mathbf{x}$, and weights $\boldsymbol{\theta}$.

More recently, Berisha et al. have introduced the D_α-divergence between distributions [19, 16], defined as

$$D_\alpha(p, q) = \frac{1}{4\alpha(1-\alpha)}\left[\int \frac{(\alpha p(\mathbf{x}) - (1-\alpha)q(\mathbf{x}))^2}{\alpha p(\mathbf{x}) + (1-\alpha)q(\mathbf{x})} d\mathbf{x} - (2\alpha - 1)^2\right]. \quad (4)$$

This divergence has the remarkable property that it can be estimated directly without estimation or plug-in of the densities p and q based on an extension of the Friedman-Rafsky (FR) multi-variate two sample test statistic [20]. Let us consider N_p samples from p and N_q samples from q, denoted by $\mathbf{X}_p \in \mathbb{R}^{N_p \times K}$ and $\mathbf{X}_q \in \mathbb{R}^{N_q \times K}$. The FR test statistic, $\mathcal{C}(\mathbf{X}_p, \mathbf{X}_q)$, is constructed by first generating a Euclidean minimal spanning tree (MST) on the concatenated data set, $\mathbf{X}_p \cup \mathbf{X}_q$, and then counting the number of edges connecting a data point from p to a data point from q. In [19], the authors define an asymptotically consistent estimator for (4) in terms of the FR test statistic. In other words, the quantity in (4) can be estimated directly from the data sampled from p and q without parametric assumptions on these distributions.

By combining the FIM relationship in (3) with the divergence measure in (4), the authors in [16] outline a method for non-parametrically estimating the FIM. Through multiple perturbations of the parameters of the network, $\boldsymbol{\theta}$, based on (3), we can construct a system of equations where the only unknown parameters are the entries of the FIM. As a result, we can construct a semidefinite program whose solution reconstructs the FIM (see sections 2 and 3 in [16] for exact details).

The size of the FIM is $d \times d$; estimating such a large matrix can be prohibitive for networks with large numbers of parameters. In Algorithm 1, we propose an alternate means of reconstructing just the diagonal entries of the FIM by iteratively perturbing each parameter in the network and generating a new data set from the network output for each perturbation. By calculating the D_α-divergence between the original network output and the network output for the perturbed network, we can iteratively estimate the diagonal components of the FIM matrix one at a time. If we use a random perturba-

Algorithm 1 DNN parameter ranking algorithm based on the Fisher Information

Input: A trained DNN with parameters $\boldsymbol{\theta}$, input data $\mathbf{X}$, and input-output function $\mathbf{Y} = f(\mathbf{X}; \boldsymbol{\theta})$

Output: The indices of parameters of the network, ranked by their contribution to the FIM, $\boldsymbol{\theta}_{\text{IDX}}$

Define: $\mathbf{d} = \emptyset$
$\mathbf{Y} = f(\mathbf{X}; \boldsymbol{\theta})$
$d = \#\text{parameters}$

for $i \in 1 \dots d$ **do**
$\tilde{\boldsymbol{\theta}} = \boldsymbol{\theta}$
$u_i = \text{SmallRandomPertubation}();$
$\tilde{\boldsymbol{\theta}}(i) = \tilde{\boldsymbol{\theta}}(i) + u_i$
$\mathbf{Y}_i = f(\mathbf{X}; \tilde{\boldsymbol{\theta}})$
$\mathbf{d}(i) = D_\alpha(\mathbf{Y}, \mathbf{Y}_i)$
end for
$\boldsymbol{\theta}_{\text{IDX}} = \text{SortIdx}(\mathbf{d})$

tion, we can use Monte Carlo methods to generate multiple estimates of D_α and average to estimate the FIM diagonal components. Since the perturbation is small, the divergence dominates the parameter importance. Thus, we do not divide the divergence by the perturbation in Algorithm 1, keeping parameter importance in a reasonable range.

4. EXPERIMENT AND RESULTS

We consider a non-linear dimensionality reduction task using deep autoencoder for evaluation of the method. The data used in the experiment is from accelerometer data measuring the gait of patients with Parkinson's disease. The bottom of each patient's foot is fitted with 9 pressure sensors that measure impact force (in Newtons) over time with a sampling rate of 100 samples/second. This results in a 18-dimensional time-varying signal. We use a 3-layer autoencoder to reduce the dimension of the data from 18 dimensions to 2, using the nonlinear PCA toolbox for Matlab [21]. The network has one hidden layer with 6 nodes and an output layer has 2 nodes. The total number of parameters including bias terms is 128. Conjugate gradient descent is used to train the autoencoder on 25 seconds of data. An additional 12.5 seconds of data is used to learn the ranking of the parameters using Algorithm 1. Here, the importance of each parameter is the average of 10 perturbations. We use this ranking to remove unimportant parameters by setting their value to 0 and quantize the remaining parameters by assigning the number of bits based on their relative importance. A held-out test set is used to evaluate the performance of the pruning and quantization methods.

We compare the proposed ranking algorithm against methods used in the literature, namely magnitude-based pruning [4] and the Fisher Information-based pruning assuming a parametric data model [7] [8]. The latter method is very similar to Algorithm 1, except for two important differences:

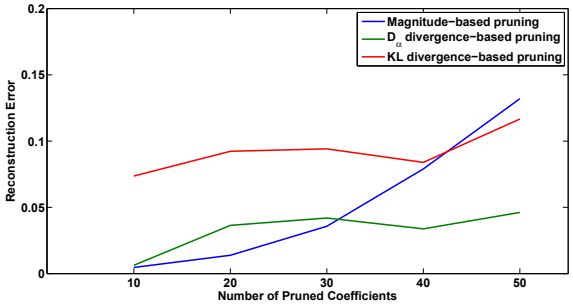

Fig. 2. Reconstruction error change with increasing number of removed parameters. There are a total of 128 parameters in the network.

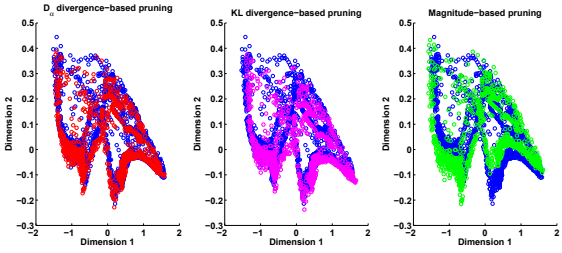

Fig. 3. Original network output compared to output after pruning. In all three plots, blue points represent network outputs without parameter pruning. The other colors represent the outputs of the pruned network using three different pruning methods. The closer the two sets of points, the better the representation.

it uses Kullback-Leibler (KL) divergence between the original and the perturbed model; and it assumes the output data is Gaussian distributed in order to be able to estimate this divergence measure. After we obtain the importance index of each parameter (including bias terms), we conduct two sets of experiments to evaluate our algorithm. First, we prune the neural network by removing a varying number of unimportant parameters. Second, we consider a fixed-point implementation of the neural network and quantize the parameters using a quantization scheme that assigns more bits to more important parameters.

4.1. Complexity reduction through parameter pruning

Given the parameter ranking from all three methods, we prune the neural network by directly removing the least important parameters (as determined by each model). Different numbers of parameters from 10 to 50 with step 10 are removed to test the impact of parameter pruning on neural network performance. To show how parameter pruning will impact the reconstruction error, we then reconstruct the input using the intact decoder trained beforehand. Fig. 2 shows the result of the reconstruction error of different methods by removing different numbers of parameters. We see that by deleting a small

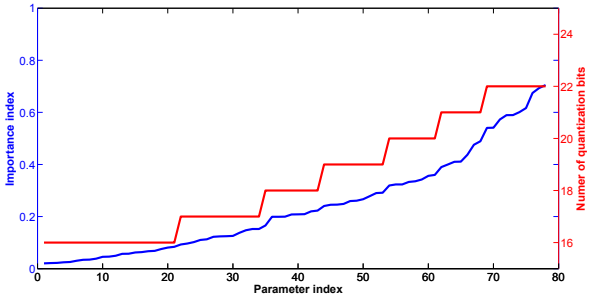

Fig. 4. The blue line is the scaled Fisher information associated with the unpruned network parameters sorted in ascending order. The red step function shows the results of k-means clustering. The parameters falling in each cluster are represented using a different number of bits.

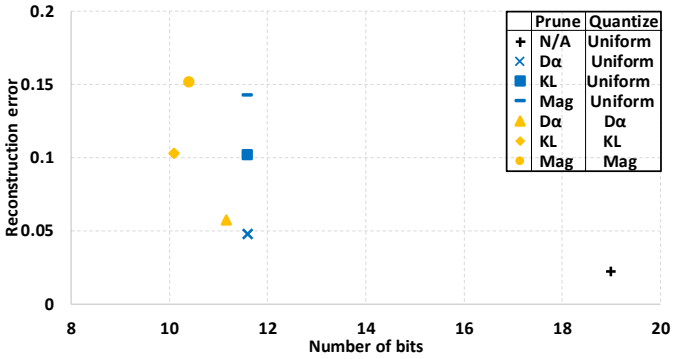

Fig. 5. Average bit rate and reconstruction error for the different methods. The methods that use pruning assume 0 bits/sample for the removed samples.

number of unimportant parameters magnitude-based pruning is the best. However, when more than 30 parameters are removed, divergence-based pruning surpasses magnitude-based pruning and reconstruction error of D_α-divergence is much lower than the method based on the KL divergence.

The FIM estimator depends on the ability to correctly estimate the D_α-divergence. As was discussed in [16], this is more challenging for small values of D_α. When the error in this estimation dominates the ranking of weights, the exact weight selected by D_α is almost random (all weights whose Fisher Information is less than the error threshold are about equally likely). In contrast, weights very near zero are so low that they rarely affect neuron firing. So the magnitude-based pruning makes good decisions early on by removing almost-zero weights, while the D_α method makes near random picks from among the low-Fisher-Information weights because the error in estimation dominates the answer. As the figure shows, this changes dramatically once the magnitude of D_α exceeds the estimation error.

The scatter plots of the 2 dimensional outputs of the original autoencoder and the pruned autoencoder (50 parameters removed) are shown in Fig. 3. It is easy to see that magnitude-based pruning brings largest deviation from the original output, while KL divergence-based pruning is better and D_α-divergence-based pruning is the best.

4.2. Complexity reduction through parameter quantization

In order to assess various methods for reducing the network's stored size, we evaluate both pruning and quantization in combination. When we apply a pruning method, we prune the least important 50 weights from the neural network using one of the methods from section 4. The pruning methods considered are: [none, D_α-ranked, KL-ranked, and Mag-ranked]. We also consider two quantization methods. First is a uniform 19-bit quantization for all parameters. Our second quantization method is a varying bit-rate quantization method based on k-means clustering. We rank-order the weights by an importance metric (D_α, KL, or Mag) and cluster them into 7 clusters. The clusters are then quantized using varying bit depths, from 16 bits (least important) through 22 bits (most important). Fig. 4 shows an overlay of bit-depth clusters versus parameter ranking as determined by the D_α-divergence.

We compare the reduction methods based on two criteria in Fig. 5: the reconstruction error and the number of bits per parameter (BPP) required to achieve this reconstruction error. The results of varying bit-rate quantization are averages of 10 trials. We notice some general trends from this plot: (1) no pruning and uniform quantization produces the lowest error but requires by far the most BPP; (2) relevance-based quantization results in slightly lower BPP than methods with uniform quantization, while pruning dramatically reduces the required BPP; and (3) the networks resulting from pruning and quantization based on the D_α divergence outperform the networks reduced using other methods.

5. CONCLUSION

In this paper we proposed a new Fisher Information criterion based method to rank the set of parameters in a DNN. We used recent results from the authors on non-parametric estimates of the Fisher Information to construct an algorithm that can iteratively estimate the Fisher Information in a DNN through perturbation analysis. We used this method to reduce the complexity of a trained deep autoencoder by removing redundant parameters and by quantizing the remaining parameters based on their relative importance. Moving forward, our aim is to build the pruning and quantization methods into the training procedure by developing new regularization algorithms that make the networks insensitive to parameter perturbation.

6. REFERENCES

[1] Russell Reed, "Pruning algorithms-a survey," *Neural Networks, IEEE Transactions on*, vol. 4, no. 5, pp. 740–747, 1993.

[2] 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.

[3] Michael C Mozer and Paul Smolensky, "Skeletonization: A technique for trimming the fat from a network via relevance assessment," in *Advances in neural information processing systems*, 1989, pp. 107–115.

[4] D Rumelhart, "Learning and generalization," in *IEEE International Conference on Neural Networks.(San Diego 1988), IEEE*, 1988.

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

[6] 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.

[7] Pierre Van De Laar and Tom Heskes, "Pruning using parameter and neuronal metrics," *Neural Computation*, vol. 11, no. 4, pp. 977–993, 1999.

[8] Hyunjin Lee, Hyeyoung Park, and Yillbyung Lee, "Network optimization through learning and pruning in neuromanifold," in *PRICAI 2002: Trends in Artificial Intelligence*, pp. 169–177. Springer, 2002.

[9] Tomas Lundin and Perry Moerland, "Quantization and pruning of multilayer perceptrons: Towards compact neural networks," Tech. Rep., IDIAP, 1997.

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

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

[12] 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.

[13] 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.

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

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

[16] Visar Berisha and Alfred O Hero, "Empirical non-parametric estimation of the fisher information," *Signal Processing Letters, IEEE*, vol. 22, no. 7, pp. 988–992, 2015.

[17] S-I Amari and A Cichocki, "Information geometry of divergence functions," *Bulletin of the Polish Academy of Sciences: Technical Sciences*, vol. 58, no. 1, pp. 183–195, 2010.

[18] Imre Csiszár and Paul C Shields, "Information theory and statistics: a tutorial," *Communications and Information Theory*, vol. 1, no. 4, pp. 417–528, 2004.

[19] V. Berisha, A. Wisler, A.O. Hero, and A. Spanias, "Empirically estimable classification bounds based on a non-parametric divergence measure," *Signal Processing, IEEE Transactions on*, vol. PP, no. 99, pp. 1–1, 2015.

[20] Jerome H Friedman and Lawrence C Rafsky, "Multivariate generalizations of the Wald-Wolfowitz and Smirnov two-sample tests," *The Annals of Statistics*, pp. 697–717, 1979.

[21] Matthias Scholz, "Validation of nonlinear PCA," *Neural processing letters*, vol. 36, no. 1, pp. 21–30, 2012.