OPTIMIZING THE BIT ALLOCATION FOR COMPRESSION OF WEIGHTS AND ACTIVATIONS OF DEEP NEURAL NETWORKS

Wang Zhe[1,2]*, Jie Lin*[2]*, Vijay Chandrasekhar*[2]*, Bernd Girod*[1]

[1]Department of Electrical Engineering, Stanford University
[2]Institute for Infocomm Research, A*STAR, Singapore

ABSTRACT

For most real-time implementations of deep artificial neural networks, both weights and intermediate layer activations have to be stored and loaded for processing. Compression of both is often advisable to mitigate the memory bottleneck. In this paper, we propose a bit allocation framework for compressing the weights and activations of deep neural networks. The differentiability of input-output relationships for all network layers allows us to relate the neural network output accuracy to the bit-rate of quantized weights and layer activations. We formulate a Lagrangian optimization framework that finds the optimum joint bit allocation among all intermediate activation layers and weights. Our method obtains excellent results on two deep neural networks, VGG-16 and ResNet-50. Without requiring re-training, it outperforms other state-of-the-art neural network compression methods.

Index Terms— Deep Learning, Coding, Compression.

1. INTRODUCTION

Deep Neural Networks [1, 2, 3, 4, 5] (DNNs) have become the state-of-the-art techniques for many computer vision tasks. However, DNNs often have millions of parameters and are computationally demanding. The high memory and computation cost makes it challenging to deploy DNNs on resource-limited mobile devices. One of the ways to mitigate this issue is compression of DNN weights. With millions of parameters defining the deep cascade of convolutional and fully connected layers, these weights cannot be kept in cache but have to be loaded from off-chip memory when needed. Similarly, the intermediate layer activations have to be stored and reloaded. The volume of both should be reduced as much as possible to mitigate the memory bottleneck that might otherwise slow down processing. It is not surprising that neural network compression has received considerable attention in the last few years [6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27].

In this paper, we study the impact of quantization of both weights and activations on the performance of neural networks. We show how to minimize the performance loss due to quantization while achieving the lowest possible bit-rate without re-training the quantized neural networks, which would be computationally expensive. Most prior works only quantize (or prune) the weights of a neural network while keeping the floating-point representation of the activations. However, for many state-of-the-art neural networks, the data volume associated with the activations is not negligible. Thus, our work considers the problem of compressing both weights and activations in a joint framework.

The main novelty of this paper is summarized as follows. We first analyze how quantization affects the performance of neural networks. We find the relationship between the quantization of weights and activations and the output error of a neural network and then formulate our compression framework as an optimization problem to minimize this output error. While quantizing and encoding each layer's weights and activations independently, we assign unequal bit-rate to different layers. Compared with equal bit allocation in prior proposals, our unequal bit allocation strategy reduces the output error of the quantized neural networks while maintaining the same average bit-rate. We propose an efficient method to solve the bit allocation problem using a Lagrangian formulation. The proposed method finds the Pareto-optimal bit allocation in polynomial time. We evaluate the proposed compression framework on two deep neural networks, VGG-16 [2] and ResNET-50 [4]. Our method does not require computationally expensive re-training. Without re-training, our scheme outperforms other state-of-the-art DNN compression methods over the ImageNet [28] dataset.

Few prior proposals have discussed the compression of intermediate activation layers [12, 16, 14, 26, 27] and none analyze the quantization of weights and activations in a common framework. To the best of our knowledge, our work is the first paper to jointly analyze the quantization of weights and activations and solve the bit allocation problem using a Lagrangian formulation.

The remainder of this paper is structured as follows. Section 2 analyzes the impact of quantization. Section 3 develops our bit allocation framework for weights and activations. Section 4 reports experimental results on the performance of our scheme and compares it with methods previously proposed in the literature.

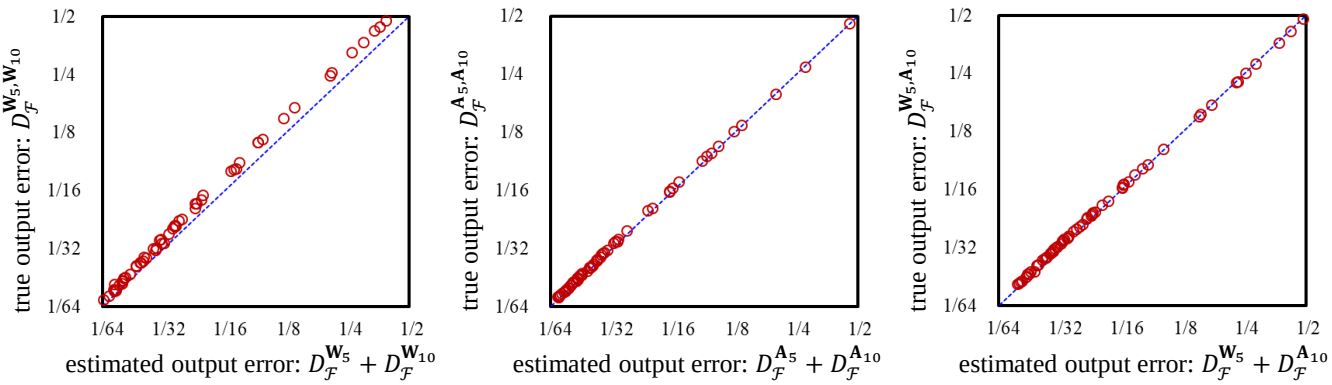

Fig. 1: Demonstration of the output error additivity when quantizing two layers of weights or activations in VGG-16 [2]. The figures from left to right are: (a) quantizing weights in layer 5 and weights in layer 10; (b) quantizing activations in layer 5 and activations in layer 10; (c) quantizing weights in layer 5 and activations in layer 10.

2. ANALYSIS OF QUANTIZATION IMPACT

In this paper, we adopt uniform scalar quantization of each layer's weights and activations, followed by entropy coding of the quantization index. The quantizer stepsize is individually chosen for each weight layer and each activation layer. Let W_i denote the weights of layer i and A_i denote the activations of layer i for all $1 \leq i \leq L$ where L is the total number of layers. Given neural network $\mathcal{F}$ and input image I, we have output vector $V = \mathcal{F}(I)$. If we quantize the weights W_i, we obtain a modified output vector $\widehat{V}$. The output error caused by quantizing W_i is defined as the expectation of squared Euclidean distance between original output V and modified output $\hat{V}$ divided by the dimension of V

$$D_F^{W_i} = \frac{E(d(V, \widehat{V}))}{dim(V)} \tag{1}$$

where $E(.)$ denotes the expectation operator and $d(X, Y)$ is the squared Euclidean distance between vectors X and Y. Correspondingly, we have $D_{\mathcal{F}}^{A_i}$ denote the output error caused by quantization of activations A_i for all $1 \leq i \leq L$.

2.1. Additivity of Output Error

Let $D_{\mathcal{F}}$ denote the output error caused by quantizing all layer's weights and activations. We expect that $D_{\mathcal{F}}$ equals the sum of all output error due to the quantization of individual layer's weights and activations

$$D_{\mathcal{F}} = D_{\mathcal{F}}^{W_1} + ... + D_{\mathcal{F}}^{W_L} + D_{\mathcal{F}}^{A_1} + ... + D_{\mathcal{F}}^{A_L}. \tag{2}$$

Figure 1 illustrates the additivity of the output error caused by quantizing two layers of weights or activations. The vertical axis in Figure 1 represents the output error when quantizing two layers simultaneously. The horizontal axis represents the sum of output error caused by quantizing each layer individually. The data points in Figure 1 are all very close to the diagonal, meaning that the additivity property holds. This property can be shown mathematically by linearizing the quantization error of weights and activations using Taylor series expansion as the neural network is continuously differentiable and the quantization error can be considered as small deviations.

3. BIT ALLOCATION OPTIMIZATION

Let R^{W_i} and R^{A_i} denote the total bit-rates of weights and activations in the i-th layer, respectively. According to (2), the output error of quantizing all layers' weights and activations is the sum of the output error caused by quantizing each layer's weights and activations individually. The bit allocation framework is thus formulated as

$$\begin{aligned} \underset{R^{W_i}, R^{A_i}}{\arg\min} \quad & D_{\mathcal{F}}^{W_1} + D_{\mathcal{F}}^{A_1} + ... + D_{\mathcal{F}}^{W_L} + D_{\mathcal{F}}^{A_L} \\ s.t. \quad & R^{W_1} + R^{A_1} + ... + R^{W_L} + R^{A_L} = R^{Total} \end{aligned} \tag{3}$$

where R^{Total} is the total number of bits needed to represent the weights and activations of all layers. Our goal is to find the optimum bit allocation $\{R^{W_i}\}_{i=1}^{L}$ and $\{R^{A_i}\}_{i=1}^{L}$ to minimize output error within a constrained total bit budget for weights and activations.

3.1. Pareto-optimal Bit Allocation

We use a classical Lagrangian rate-distortion formulation to find the optimum bit allocation. Our Lagrangian cost function is

$$\mathcal{J} = \sum_i (D_{\mathcal{F}}^{W_i} + D_{\mathcal{F}}^{A_i}) - \lambda \cdot \sum_i (R^{W_i} + R^{A_i}). \tag{4}$$

By setting the partial derivatives of $\mathcal{J}$ with respect to R^{W_i} and R^{A_j} to zero, we obtain

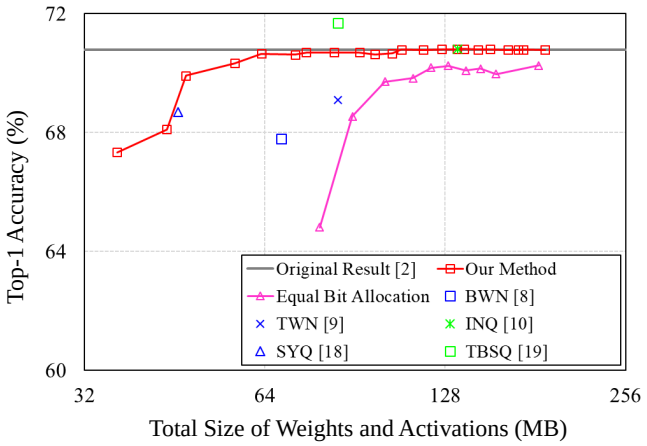

Fig. 2: The results of state-of-the-art methods on VGG-16 [2] over ImageNet [28] dataset. The total size of weights and activations of original VGG-16 is 579 MB.

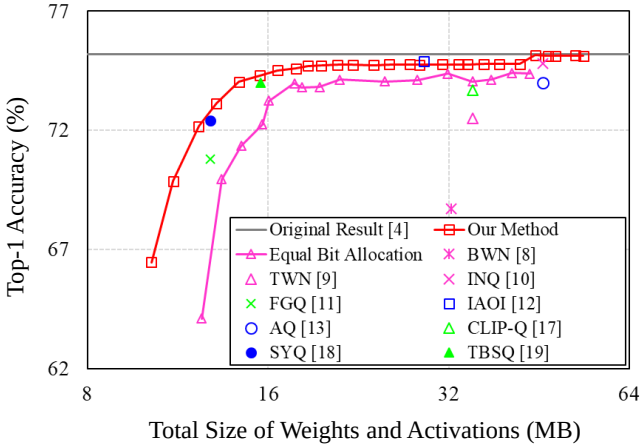

Fig. 3: The results of state-of-the-art methods on ResNet-50 [4] over ImageNet [28] dataset. The total size of weights and activations of original ResNet-50 is 116 MB.

$$\frac{\partial D_{\mathcal{F}}^{W_i}}{\partial R^{W_i}} = \frac{\partial D_{\mathcal{F}}^{A_j}}{\partial R^{A_j}} = \lambda \tag{5}$$

for all $1 \leq i, j \leq L$. The above equation expresses the intuitively pleasing insight that the slopes of output error vs rate functions for weights and activations of each layer have to be equal. This is exactly the well-known Pareto condition for optimal resource allocation in economics. (5) holds, as long as the output error contributions $D_{\mathcal{F}}^{W_i}$ and $D_{F}^{A_i}$ are additive, rates R^{W_i} and R^{A_i} are additive, the functions $D_{\mathcal{F}}^{W_i}(R^{W_i})$ and $D_{\mathcal{F}}^{A_i}(R^{A_i})$ are convex and the resulting $R^{W_i}, R^{A_i} > 0$. It does not make assumptions about the specific quantization or coding scheme used.

4. EXPERIMENTS

We evaluate our method on two state-of-the-art neural networks VGG-16 [2] and ResNet-50 [4] over the ImageNet [28] validation dataset with 224x224 input image resolution. The total number of values to be stored and loaded from memory for VGG-16 is 151M with 138M weights and 13M activations. The total number of weights and activations in ResNet-50 is 30M with 22M weights and 8M activations.

The comparisons in our experiments include Binarized Weight Network (BWN) [8], Ternary Weight Network (TWN) [9], Incremental Network Quantization (INQ) [10], Fine-grained Quantization (FGQ) [11], Integer Arithmetic-only Inference (IAOI) [12], Adaptive Quantization (AQ) [13], Compression Learning by In-parallel Quantization (CLIP-Q) [17], Symmetric Quantization (SYQ) [18], and Two-bit Shift Quantization (TBSQ) [19].

We adopt the standard Tensorflow implementation of VGG-16 and ResNet-50 provided by the SLIM library. The original VGG-16 and ResNet-50 in our method have 70.8% and 75.2% top-1 image classification accuracy on the ImageNet validation dataset, respectively.

4.1. Effectiveness of Unequal Bit Allocation

We show the performance of equal bit allocation as a baseline to evaluate the effectiveness of the proposed unequal bit allocation strategy. Equal bit allocation uses the same bit-width for all layers' weights and activations. For a fair comparison, we also perform entropy coding to encode the quantization index for equal bit allocation. As shown in Figs. 2 and Figure 3, there is a large gap between the results of our method and the results of equal bit allocation. Unequal bit allocation is much more effective.

4.2. Comparison with State-of-the-art Methods

Figure 2 and Figure 3 show the results of state-of-the-art methods on VGG-16 [2] and ResNet-50 [4] over ImageNet [28] dataset. Our method achieves excellent results on both VGG-16 and ResNet-50. On VGG-16, our method has no accuracy loss at 4x compression rate when weights and activations are 8 bits on average and only loses 0.2% accuracy at 8x compression rate when weights and activations are 4 bits on average. In particular, the accuracy loss on VGG-16 of our method is less than 1% even though weights and activations are compressed into 3 bits on average. On ResNet-50, our method loses 0.4% at 4x compression rate when weights and activations are 8 bits on average and loses 1.0% at 8x compression rate when weights and activations are 4 bits on average. Our method outperforms others except TBSQ [19] on VGG-16. After re-training, TBSQ achieves 71.7% top-1 accuracy on VGG-16, which is higher than the result of original model (70.8%). But on ResNet-50, our method is better than TBSQ. Without re-training, our method achieves

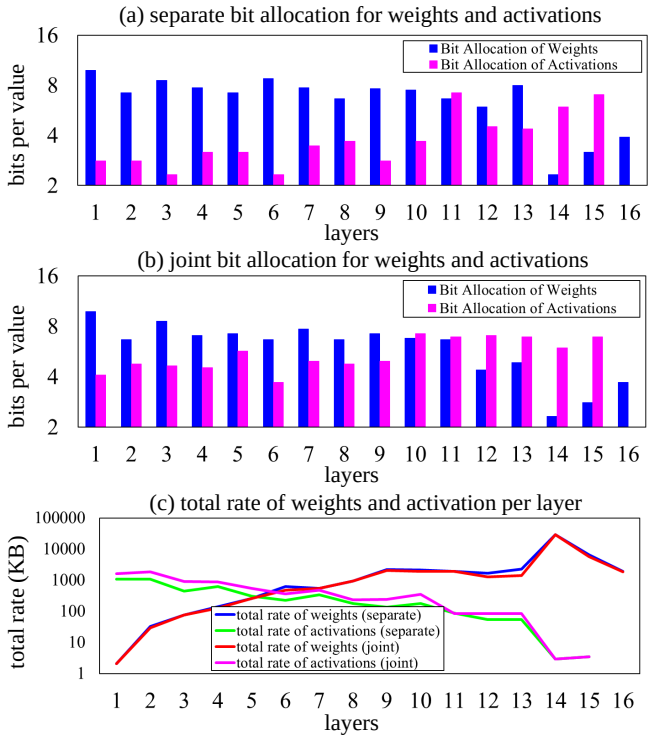

Fig. 4: The bit allocation on VGG-16 [2] when weights and activations are compressed into 3 bits on average. The output error in (a) and (b) are 3.71 and 3.57, which is the result on five hundred testing images randomly selected from the ImageNet [28] dataset.

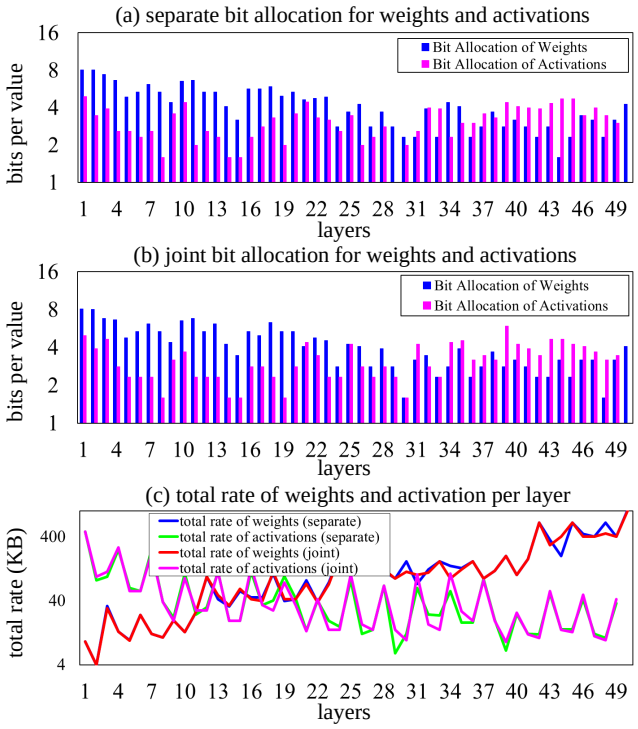

Fig. 5: The bit allocation on ResNet-50 [4] when weights and activations are compressed into 3 bits on average. The output error in (a) and (b) are 1.97 and 1.88, which is the result on five hundred testing images randomly selected from the ImageNet [28] dataset.

high compression rate on VGG-16 [2] and ResNet-50 [4] and outperforms other state-of-the-art methods.

4.3. Bit Allocation across Layers

Figs. 4 and 5 show the bit allocation on VGG-16 [2] and ResNet-50 [4] when weights and activations are compressed to 3 bits on average. We do not quantize the last layer of activations. In Figure 4(a) and Figure 5(a), bit allocation for weights and activations is performed separately, and in Figure 4(b) and Figure 5(b), bit allocation for weights and activations is performed jointly.

Joint bit allocation for both weights and activations can obtain smaller output error. For example, the output error in Figure 4(a) is 3.71 when the bit allocation for weights and activations is performed separately. While the output error in Figure 4(b) drops to 3.57 when the bit allocation for weights and activations is performed jointly. Besides, we can see that the bit-rates of weights in Figure 4(b) are slightly smaller than the bit-rates of weights in Figure 4(a). Conversely, the bit-rates of activations in Figure 4(b) are slightly larger than the bit-rates of activations in Figure 4(a). This shows that the joint bit allocation framework adjusts the bit-rates across the layers of weights and activations to lower the overall output error. Figure 4(c) and Figure 5(c) show the total rate of weights and activations per layer when performing separate bit allocation and joint bit allocation.

5. CONCLUSIONS

In this paper, we have proposed a novel bit allocation framework for compression of both weights and activations of deep neural network. The output error due to the quantization of individual layer's weights and activations is additive, as are the bits used to represent each layer. Building on this observation, our framework adjusts the bit-rates across the layers of weights and activations to minimize the total output error of the neural network and finds the Pareto-optimum bit allocation by using a Lagrangian formulation. This unequal bit allocation strategy is not only computationally straightforward and very fast, it is also much more effective than equal bit allocation and can obtain excellent results. Unlike competing methods, it does not require re-training, which is computationally very expensive and might not even be possible, if the original training data are not available. On both VGG-16 and ResNet-50, our method obtains more than 8x compression rate with only minor accuracy loss (less than 1%) and outperforms other state-of-the-art methods.

6. REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," in *NIPS*, 2012.

[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *ICLR*, 2015.

[3] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *CVPR*, 2015.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *ECCV*, 2016.

[6] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *NIPS*, 2015.

[7] S. Han, H. M., and W. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," in *ICLR*, 2016.

[8] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-NET: Imagenet classification using binary convolutional neural networks," in *arXiv preprint arXiv:1603.05279*, 2016.

[9] F. Li, B. Zhang, and B. Liu, "Ternary weight networks," in *arXiv preprint arXiv:1605.04711*, 2016.

[10] A. Zhou, A. Yao, Y. Guo, L. Xu, and Y. Chen, "Incremental network quantization: Towards lossless cnns with low-precision weights," in *arXiv preprint arXiv:1702.03044*, 2017.

[11] N. Mellempudi, A. Kundu, D. Mudigere, D. Das, B. Kaul, and P. Dubey, "Ternary neural networks with fine-grained quantization," in *arXiv preprint arXiv:1705.01462*, 2017.

[12] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *CVPR*, 2018.

[13] Y. Zhou, Seyed-Mohsen Moosavi-Dezfooli, Ngai-Man Cheung, and P. Frossard, "Adaptive quantization for deep neural network," in *AAAI*, 2018.

[14] B. Zhuang, C. Shen, M. Tan, L. Liu, and I. Reid, "Towards effective low-bitwidth convolutional neural networks," in *CVPR*, 2018.

[15] M. Courbariaux, Y. Bengio, , and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *NIPS*, 2015.

[16] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, "Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients," in *arXiv preprint arXiv:1606.06160*, 2016.

[17] F. Tung and G. Mori, "CLIP-Q: Deep network compression learning by in-parallel pruning quantization," in *CVPR*, 2018.

[18] J. Faraone, N. Fraser, M. Blott, and P. Leong, "SYQ: Learning symmetric quantization for efficient deep neural networks," in *CVPR*, 2018.

[19] C. Leng, H. Li, S. Zhu, and R. Jin, "Extremely low bit neural network: Squeeze the last bit out with ADMM," in *AAAI*, 2018.

[20] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," in *NIPS*, 2016.

[21] D. Das et al., "Mixed precision training of convolutional neural networks using integer operations," in *ICLR*, 2018.

[22] L. Hou and J. Kwok, "Loss-aware weight quantization of deep networks," in *ICLR*, 2018.

[23] S. Khoram and J. Li, "Adaptive quantization of neural networks," in *ICLR*, 2018.

[24] U. Koster et al., "Flexpoint: An adaptive numerical format for efficient training of deep neural networks," in *NIPS*, 2017.

[25] E. Park, D. Kim, S. Yoo, and P. Vajda, "Precision highway for ultra low-precision quantization," in *arXiv:1812.09818*, 2018.

[26] J. McKinstry et al., "Discovering low-precision networks close to full-precision networks for efficient embedded inference," in *arXiv:1809.04191*, 2018.

[27] D. Zhang, J. Yang, D. Ye, and G. Hua, "LQ-Nets: Learned quantization for highly accurate and compact deep neural networks," in *ECCV*, 2018.

[28] J. Deng et al., "ImageNet: A large-scale hierarchical image database," in *CVPR*, 2009.