ON QUANTIZATION OF IMAGE CLASSIFICATION NEURAL NETWORKS FOR COMPRESSION WITHOUT RETRAINING

Marcos Tonin and Ricardo L. de Queiroz

Eletrical Engineering and Computer Science Departament
Universidade de Brasília, Brazil
marcosvtonin@gmail.com and queiroz@ieee.org

ABSTRACT

We studied the quantization of neural networks for their compression and representation without retraining. The goal is to facilitate neural network representation and deployment in standard formats so that general networks may have their weights quantized and entropy coded within the deployment format. We relate weight entropy and model accuracy and try to evaluate distribution of weights against known distributions. Many scalar quantization strategies were tested. We have found that weights are typically approximated by a Laplacian distribution for which optimal quantizers are approximated by entropy-coded uniform quantizers with deadzones. Results indicate that it is possible to reduce 8-fold the size of the popular image classification networks with accuracy losses near 1%.

Index Terms— Neural network compression, weight quantization, ONNX file compression.

1. INTRODUCTION

The use of artificial intelligence (AI) to solve problems in diverse areas of knowledge is increasingly frequent and intense. One of the reasons for this high frequency is the applicability in different contexts and the ability to respond to real challenges [1], such as safe driving in autonomous cars [2].

AI often uses neural networks (NN) [1], which have been used in different applications, such as image classification, object detection, body analysis, machine comprehension, machine translation. NN models are made of weights, biases, computational units, and layers [1].

In Internet of Things (IoT) [3] and Edge Computing [4], typical equipments have limited resources and energy [5, 6]. Hence, the NN models should be compacted and reduced. These constraints have given rise to interest in NN compression. Since NN is used in the most diverse devices and varied occasions, the interoperability of the NN becomes a critical point in their development. The Open Neural Network eXchange (ONNX) format allows interoperability between frameworks to train the model in one tool and use another for inference and prediction [7]. ONNX works by defining a standard set of operators and default data types, in addition to weights and biases. The weights can be represented as 32-bit floating-point as specified in IEEE 754 [8], a *float*. There is also the possibility to store the parameters as a 64-bit floating-point, a *double*. There are ONNX models for object detection, gesture analysis, text translation, image classification, etc., for example, MobileNet, AlexNet, GoogLenet and VGG [7].

There are several works on NN compression. However, most involve retraining the model or changing its structure (e.g., insertion or removal of layers), such as the methods that use pruning [9, 10, 11, 12, 13], layer quantization [14], weight sharing [15] and quantization in general [16]. We are interested in compression without retraining. For example, Seo and Kim use a hybrid method with uniform compression followed by K-means clustering [17]. Dupuis et al. achieve compression by sharing weights among layers [18].

Among the compression with retraining efforts, we have the MPEG's call for NN compression (MPEG–NNR), which aims at defining a compressed, interpretable and interoperable representation for trained NN [19]. Also, MPEG-NNR recommends interoperable formats like ONNX and NNEF (Neural Network Exchange Format) for a compressed representation of NN. MPEG-NNR's call for proposal has the following requirements [19]: efficient representation of the model (the size of the compressed model has to be at least 30% smaller than the original model); to support different types of NN (CNN, RNN and others); the compressed representation may contain all the parameters and weights of the NN; the possibility of performing the inference of the compact model; the method to compress the NN independently of the dataset used to train the original model; low computational power and memory consumption to perform decoding.

We, however, are interested in reducing the size of networks in ONNX format without retraining them and without imposing a significant performance impact.

2. WEIGHT QUANTIZATION

If one is to look for data inter-dependence within NN, there must be an implicit ordering among the data. Weights and

biases of the NN model are separated into layers, which are usually ordered from the input to the output one. However, there is no exact ordering, and we could not find clear peaks in spectral analysis or correlation functions to warrant vector quantization or transformation. Therefore, as we found no dependence between the coefficients, we focus on scalar quantization.

Table 1. Some ONNX's models.

Models	File size (MB)	Percentage ratio of weights and biases in the file(%)
Caffenet	232.57	99.999
Efficientnet-lite4	49.54	99.841
GoogLenet	26.72	99.907
Mobilenet	13.59	99.366
Resnet	170.40	99.930
Shufflenet	5.46	99.246
Squeezenet	4.73	99.713
VGG	548.15	99.997

Most of the data in an ONNX file are weights and biases, as shown in Table 1. Their amplitude distribution is usually concentrated around zero and is approximately symmetric around zero. The weights distribution of Googlenet and the Shufflenet models are shown in Fig. 1.

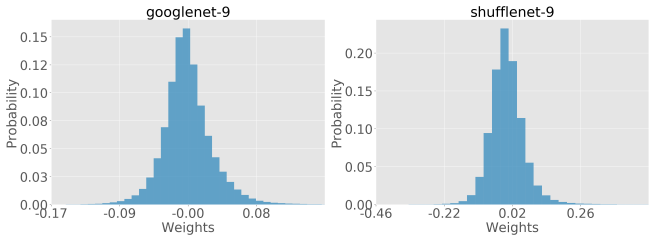

Fig. 1. Weights histogram for Googlenet and Shuffletnet models.

We analyzed the weight histograms and compared them to known probability density functions (PDF), such as Alpha, Cauchy, Exponential, Logistic, Gamma, Laplace, Gaussian, and Uniform [20, 21, 22]. In order to evaluate distribution distances, we used two distance metrics: sum of squared errors (SSE) and Kullback-Leibler divergence (KLD). Let $P(i)$ be samples of the model distribution, while $Q(i)$ be similar samples of the reference, standard distribution. Then

$$SSE = \sum_{i=1}^{n} (P(i) - Q(i))^2, \tag{1}$$

$$KLD(P||Q) = \sum_{i=1}^{n} P(i) \log \frac{P(i)}{Q(i)}. \tag{2}$$

Both SSE and KLD measure the similarity between two distributions. The KLD measures the information lost when

Table 2. SSE between standard and NN model distributions.

Models	1^{st} most similar dist.	2^{nd} most similar dist.	3^{rd} most similar dist.
Caffenet	Laplacian	Logistic	Gaussian
efficientnet	Laplacian	Logistic	Gaussian
GoogLenet	Laplacian	Logistic	Gaussian
Mobilenet	Laplacian	Gaussian	Logistic
Resnet	Laplacian	Logistic	Gaussian
Shufflenet	Cauchy	Laplacian	Logistic
Squeezenet	Logistic	Laplacian	Gaussian
VGG	Alfa	Laplacian	Logistic

Table 3. KLD between standard and NN model distributions.

Models	1^{st} most similar dist.	2^{nd} most similar dist.	3^{rd} most similar dist.
Caffenet	Gaussian	Laplacian	Cauchy
Efficientnet	Laplacian	Logistic	Gaussian
GoogLenet	Laplacian	Logistic	Gaussian
Mobilenet	Laplacian	Logistic	Alfa
Resnet	Laplacian	Logistic	Gaussian
Shufflenet	Cauchy	Laplacian	Logistic
Squeezenet	Laplacian	Logistic	Gaussian
VGG	Alfa	Laplacian	Logistic

Q is used to estimate P, and SSE is the norm of the error in between distributions.

Tables 2 and 3 rank PDFs that are most similar to the weights histogram of NN models. In Table 2, referring to the SSE, we found that the Laplacian distribution is a good approximation since the first or second distribution best approximates the models. Moreover, the KLD metric (Table 3) has the Laplacian distribution always among the three best approximations to the models. Hence we can conclude that the Laplacian distribution is a reasonable approximation. Thus, we can use Sullivan's results [23] for entropic coding of quantized values, which reports that, for Laplacian distributions, the optimal quantization is approximated by uniform quantization with dead-zones.

Uniform quantization can be *midtread* or *midrise*, as described in Table 4. With a dead-zone, the level around zero has a different range, as described in Table 4. In Table 4, X represents the value of a weight to be quantized, X_q the value passed to the decoder, and $\widehat{X}$ the value reconstructed by the decoder. Δ is the size of the quantization steps and $s(X)$ is the sign function returning 1 if the number is positive and -1 otherwise. σ defines the relative size of the dead-zone related to the quantization step. Note that $\sigma = 0$ implies the *midrise* quantizer, while the *midtread* quantizer has $\sigma = 0.5$. The $\lceil X \rceil$ operator is the ceiling (top) rounding of X.

In non-uniform quantization, step sizes are unequal, for

Table 4. Uniform Quantization Formulas.

	X_q	$\widehat{X}$
Midtread	round($\frac{X}{\Delta}$)	ΔX_q
Midrise	$s(X)\lceil\frac{X}{\Delta}\rceil$	$s(X_q)\Delta(X_q - \frac{1}{2})$
Dead-zone	$0, \lvert X\rvert < \sigma\Delta$ $s(X)\lceil(\frac{\lvert X\rvert}{\Delta} - \sigma)\rceil$	$0, X_q = 0$ $s(X_q)\Delta(\lvert X_q\rvert + \sigma - 0.5)$

example, using logarithmic functions. Usually, steps near the origin are smaller. We performed tests with the non-uniform "μ-law" quantization. Another way to perform non-uniform quantization is to use different floating-point formats. The standard format is the 32-bit *float* [8]. There are IEEE-defined versions for 16-bits and 8-bits (*half-precision and* minifloat, respectively). Other float versions can be defined using the format:

$$s_0\, e_0\, e_1 \ldots e_{p-1}\, m_0\, m_1 \ldots m_{q-1}, \tag{3}$$

where the sign s_0 is a bit, followed by p exponent bits and q mantissa bits.

3. RESULTS

Tests were performed using the ImageNet Large Scale Visual Recognition Challenge 2012 (ILSVRC 2012) validation dataset, composed of 50000 images [24]. These images have a classification among 1000 possible classes. The networks used here are presented in Table 1 (all of them are CNN type in ONNX format). In order to evaluate the performance of the image classification models, we use the accuracy Top 1 (acc_1) and the accuracy Top 5 (acc_5). acc_1 computes 1 if the top class of the model matches the ground truth and 0 otherwise; acc_5 computes 1 if any of the top 5 highest classes outputs of the network matches the ground truth and 0 otherwise. So, for the 50000 validation images, acc_1 indicates how many images the model was able to correctly predict, just considering the top class, while acc_5 indicates how many images the model was able to correctly predict, considering the top 5 classes. We have also computed the entropy of the NN weights, which estimates how many bits in average an encoder would spend to encode each quantized weight.

Table 5 shows entropy and accuracy results for the unquantized models. We evaluate the uniform quantization (*midrise* and *midtread*), non-uniform quantization (*floats* representations and "μ-law") and, also dead-zone quantization (σ as 0.1, 0.25, 0.4, and 0.7). Note that *midrise* is the case $\sigma = 0$ and *midtread* is the case $\sigma = 0.5$. For the *midrise*, *midtread* and dead-zone quantizations, we vary the number of bits from 24 to 2 bits ($2^b, 2 \leq b \leq 24$), with the weights ranging from -1 to $+1$; As for the non-uniform (μ-law) quantization, we vary μ from 2^{24} to 2^2. The *float* representation was chosen with 16, 12, 10, 8, 7, 6, 5, 4 and 3 bits according to Eq. 3 and IEEE 754.

Table 5. Entropy and accuracy results for uncompressed models.

Models	Entropy	acc_5 (%)	acc_1 (%)
caffenet	25.213	79.522	56.264
efficientnet-lite	23.513	93.684	77.734
googLenet	22.653	88.34	67.774
Mobilenet	21.673	88.934	69.3
Resnet	24.734	93.614	77.214
shufflenet	20.391	68.134	42.422
squeezenet	20.221	77.38	53.77
VGG	25.661	91.816	73.646

Figure 2 compares the acc_1 results for uniform quantization schemes. The RD results of the acc_5 metric has qualitatively similar behaviors to acc_1, but they are not quantitatively identical. Thus, when acc_1 tends to decrease, acc_5 also decreases at a slower rate. For simplicity, The results for the acc_5 metric were not shown. In Fig. 2, except for the Mobilenet case, a better result is achieved for *midrise*, *midtread*, or dead-zone with $\sigma = 0.4$. For most networks, the results do not vary much. The Mobilenet network is the most sensitive to dead-zone step size. For this model, $\sigma = 0.7$ yields the best results.

Table 6. Best RD results for a 1% accuracy drop.

Models (Method)	**Level Bits**	**Entropy achieved**	**acc_5 (%) achieved**	**acc_1 (%) achieved**
caffenet (*Midrise*)	8	1.828	78.958	55.5
efficientnet ($\sigma = 0.7$)	9	6.528	93.374	77.194
googLenet ($\sigma = 0.4$)	8	3.06	87.938	67.03
mobilenet ($\sigma = 0.7$)	14	11.914	88.482	68.304
resnet ($\sigma = 0.7$)	13	7.736	93.322	76.632
shufflenet (*Midrise*)	9	6.078	67.512	41.77
squeezenet (*Midtread*)	8	4.67	76.578	53.2
VGG ($\sigma = 0.7$)	9	2.653	91.698	73.576

Table 6 refers to the point with the best performance shown in Fig. 2. The best performance point is the point with the lowest entropy, such that there is a maximum difference of 1% in the acc_1 and acc_5 metrics compared to the model's initial accuracy.

Figure 2 and Table 6, indicate that a better result is achieved for *midrise*, *midtread*, or dead-zone with $\sigma = 0.4$ or $\sigma = 0.7$. For most networks, the results do not vary much

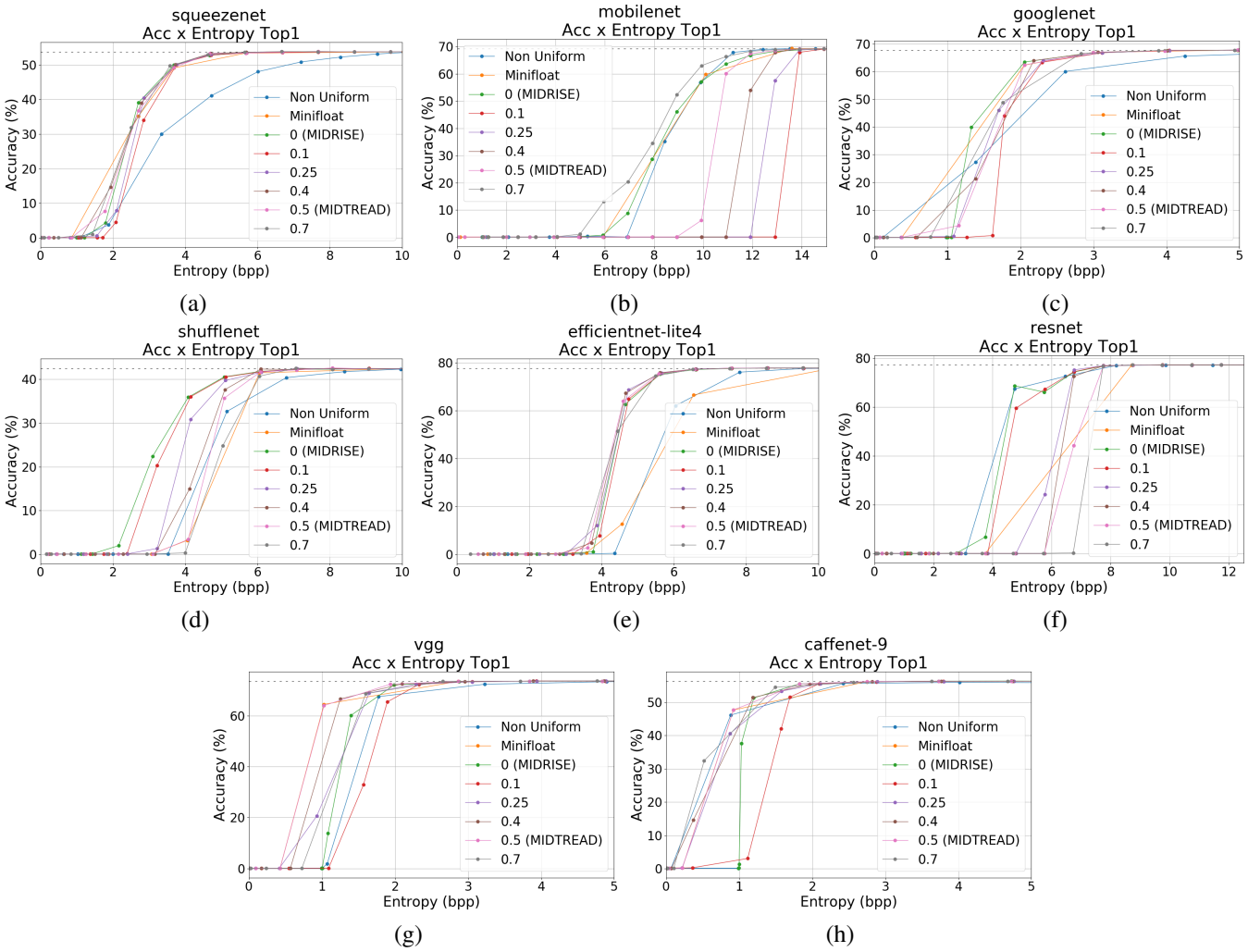

Fig. 2. Rate and distortion (RD) comparisons (entropy $\times$ accuracy) among uniform and non-uniform quantization for different networks. Results for the acc_1 metric are shown, since graphics for the acc5 are very similar.

(considering *midrise*, *midtread* and dead-zone methods). The network that shows the most divergence of results is Mobilenet, which has a noticeable difference in behaviour, when comparing dead-zone methods against non-uniform ones, *midrise* and *midtread*. In this case, the dead-zone quantizer with $\sigma = 0.7$ yields the best performance.

4. CONCLUSIONS

Results in Table 6 and Fig. 2 show us that, for most networks, it is possible to get rates close to 5 bits by weight without causing significant losses. For acc_1, as for acc_5, a maximum accuracy of 1.0% was achieved. After choosing the best quantizer and step size, an average of 5.62 bits/weight was achieved, representing a 5.6$\times$ reduction in the size of the NN originally with 32 bits. The proposed method can be useful to achieve the objectives proposed in the MPEG-NNR call [19]. For simplicity, only 8 NN models have been shown. We have tested others models, including non-image-related ones, with similar results. We hope these results could be characteristic and provide a general trend.

Future work may include tests with more NNs, and combine quantization with retraining methods.

5. REFERENCES

[1] M. Mohammed, M. B. Khan, and E. Bashier, *Machine Learning: Algorithms and Applications*, CRC, 2017.

[2] D. Feng, L. Rosenbaum, and K Dietmayer, "Towards Safe Autonomous Driving: Capture Uncertainty in the Deep Neural Network For Lidar 3D Vehicle Detection," in *2018 21st Intl. Conf. on Intelligent Transportation Systems (ITSC)*, 2018, pp. 3266–3273.

[3] M. Mahdavinejad et al, "Machine learning for internet

of things data analysis: A survey," *Digital Communications and Networks*, vol. 4, no. 3, pp. 161–175, 2018.

[4] J. Ren, Y. Pan, A. Goscinski, and R. A. Beyah, "Edge computing for the internet of things," *IEEE Network*, vol. 32, no. 1, pp. 6–7, 2018.

[5] Y. Zhang, W. Ding, and C. Liu, "Summary of convolutional neural network compression technology," in *2019 IEEE Intl. Conf. on Unmanned Systems (ICUS)*. IEEE, 2019, pp. 480–483.

[6] V. Sze, Y. Chen, T. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proc. of the IEEE*, vol. 105, no. 12, pp. 2295–2329, 2017.

[7] J. Bai et al, "ONNX: Open neural network exchange," https://github.com/onnx/onnx, 2019.

[8] Institute of Electrical and Electronics Engineers, "IEEE Standard for Floating-Point Arithmetic," *IEEE Std 754-2008*, pp. 1–70, 2008.

[9] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[10] L. Li, Z. Li, Y. Li, B. Kathariya, and S. Bhattacharyya, "Incremental deep neural network pruning based on hessian approximation," in *2019 Data Compression Conf. (DCC)*. IEEE, 2019, pp. 590–590.

[11] X. Dong, S. Chen, and S. Pan, "Learning to prune deep neural networks via layer-wise optimal brain surgeon," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[12] W. B. Zhao, Y. Li, and L. Shang, "Fuzzy pruning for compression of convolutional neural networks," in *2019 IEEE Intl. Conf. on Fuzzy Systems (FUZZ-IEEE)*. IEEE, 2019, pp. 1–5.

[13] T. Serra, A. Kumar, and S. Ramalingam, "Lossless compression of deep neural networks," in *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*. Springer, 2020, pp. 417–430.

[14] X. Zhu, W. Zhou, and H. Li, "Adaptive layerwise quantization for deep neural network compression," in *2018 IEEE Intl. Conf. on Multimedia and Expo (ICME)*. IEEE, 2018, pp. 1–6.

[15] J. Kim, M. Lee, J. Kim, B. Kim, and J. Lee, "An efficient pruning and weight sharing method for neural network," in *2016 IEEE Intl. Conf. on Consumer Electronics-Asia (ICCE-Asia)*. IEEE, 2016, pp. 1–2.

[16] J. Faraone et al, "Syq: Learning symmetric quantization for efficient deep neural networks," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2018, pp. 4300–4309.

[17] S. Seo and J. Kim, "Hybrid approach for efficient quantization of weights in convolutional neural networks," in *2018 IEEE Intl. Conf. on Big Data and Smart Computing (BigComp)*. IEEE, 2018, pp. 638–641.

[18] E. Dupuis, D. Novo, I. O'Connor, and A. Bosio, "Sensitivity analysis and compression opportunities in dnns using weight sharing," in *2020 23rd Intl. Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS)*. IEEE, 2020, pp. 1–6.

[19] "Core Experiments for Incremental Neural Network Compression," Oct. 2021, ISO/IEC JTC 1/SC 29/WG 04, output document WG04N151.

[20] A. Papoulis and U. Pillai, *Probability, random variables and stochastic processes*, McGraw-Hill, 4th edition, Nov. 2001.

[21] A. A. Salvia, "Reliability application of the alpha distribution," *IEEE Transactions on Reliability*, vol. R-34, no. 3, pp. 251–252, 1985.

[22] J. S. deCani and R. A. Stine, "A note on deriving the information matrix for a logistic distribution," *The American Statistician*, vol. 40, no. 3, pp. 220–222, 1986.

[23] G. J. Sullivan, "Optimal entropy constrained scalar quantization for exponential and laplacian random variables," in *IEEE Intl. Conf. on Acoustics, Speech and Signal Processing*. IEEE, 1994, pp. V–265.

[24] O. Russakovsky et al, "Imagenet large scale visual recognition challenge," *Int. J. Comput. Vision*, vol. 115, no. 3, pp. 211–252, Dec. 2015.