Codificação de Redes Neurais sem Retreino

Marcos Tonin e Ricardo L. de Queiroz

***Resumo*— Aprendizado de máquinas é utilizado em várias áreas de atuação para diversos problemas. Com isso, o desenvolvimento de redes neurais cresce, e o tamanho dessas redes neurais também aumenta. O objetivo deste artigo é reduzir o tamanho das redes sem retreiná-las, relacionando entropia dos pesos e acurácia dos modelos de redes neurais para vários métodos de quantização. Após análises, não é aparente uma correlação entre os pesos que permita codificação mais sofisticada que a quantização escalar agressiva (baixas taxas), seguida de codificações entrópicas dos pesos. Nossos estudos indicam que é possível diminuir a entropia dos pesos, em cerca de 4 vezes, sem ter um prejuízo significante no funcionamento da mesma. A quantização e a codificação recomendadas podem ser incorporadas a um formato de distribuição de redes neurais.**

***Palavras-Chave*— Compressão de redes neurais; ambientes de recursos limitados; Quantização; Distribuições de pesos; Open Neural Network Exchange (ONNX).**

***Abstract*— Machine learning is used in many areas on many problems. Hence, neural network development continuously grows and their sizes steadily increase. This work intends to reduce the networks file sizes without re-training. We relate weight entropy and accuracy for several quantization methods. After analysis, it is not apparent any correlation among the weights that would enable a more sophisticated coding than aggressive scalar quantization (low rates) followed by entropy coding of the weights. Our studies indicate that it is possible to reduce fourfold the network size without significantly impacting its performance. The recommended quantization and encoding may be incorporated into a format for the deployment of neural networks.**

***Keywords*— Compression of neural networks; constrained resource; Quantization; Weight distributions; Open Neural Network Exchange (ONNX).**

I. Introdução

Redes neurais têm sido usadas para os mais diversificados tipos de problemas, ocasionando uma maior procura e uso desses modelos computacionais. Podemos dividir as redes neurais em dois tipos: redes neurais convolucionais (CNN's) e redes neurais recorrentes (RNN's) [1]. A principal diferença entre elas é a capacidade das RNN's de processar dados temporais e sequenciais, enquanto que as CNN's são conhecidas por sua estrutura hierárquica [1].

As RNN's e CNN's são usadas nos mais diversos dispositivos aumentando as preocupações com a interoperabilidade entre os sistemas para que as redes possam ser usadas nos mais diversos equipamentos. Existe, inclusive, uma preocupação com o tamanho que essas redes podem alcançar e com o processamento computacional necessário. Isso fez surgir o interesse por compressão dessas redes.

O formato ONNX (Open Neural Network eXchange) [2] provê um arquivo de fonte aberto com a intenção de ser comum às redes neurais e independentes de *frameworks* específicos (*PyTorch* [3], *Keras* [4], *Tensorflow* [5]). Portanto, o ONNX é feito para permitir a interoperabilidade entre *frameworks*, sendo possível treinar o modelo em uma ferramenta e usar outra para inferência e predição. O ONNX funciona definindo um conjunto comum de operadores e os tipos de dados padrão, além dos pesos e vieses. Os pesos são representados como ponto flutuante de 32 bits, no padrão definido em IEEE 754 [6], denominado *float*. Há, ainda, a possibilidade deles serem armazenados como ponto flutuante de 64 bits, no padrão definido em IEEE 754 [6], conhecido como *double*.

O ONNX provê modelos de detecção de objeto, análise de gesto, tradução de texto e classificação de imagem, entre outros. Alguns modelos de classificação de imagens são as redes: *MobileNet* [7], *AlexNet* [8], *GoogLenet* [9] e *VGG* [10].

Há vários trabalhos acerca da compressão de redes neurais, no entanto, a maioria envolve retreino do modelo ou uma mudança da estrutura do modelo (inserção, remoção de camadas, mudanças de nós) após a compressão. Nesse modo de realizar compressão, podemos destacar os métodos que utilizam *pruning* [11], [12], [13], quantização de camadas [14], compartilhamento de pesos [15] e quantização em geral [16].

Por outro lado, os trabalhos sobre compressão sem retreinar pesos e sem a mudança de estrutura do modelo são menos abundantes. Seo e Kim usam um método híbrido com compressão uniforme seguido de agrupamento por K-*means* [17], porém, esse não se atenta à distribuição dos dados, além de realizar os testes apenas para uma rede. Dupuis et al. empregam compressão por compartilhamento de pesos entre as camadas [18], mas tem a necessidade de realizar o agrupamento para cada camada da rede. Assim, o objetivo desse artigo é reduzir o tamanho das redes no formato ONNX sem retreiná-las.

II. Correlação entre pesos

Para encontrar a melhor forma de realizar a quantização dos pesos é fundamental procurar correlações entre os pesos da rede. Assumindo que um elemento da k-ésima camada seja $x_k[n]$, podemos encontrar o coeficiente *Pearson* (ρ_c) [19] entre os pesos das camadas $x_i[n]$ e $x_j[n]$. Podemos, ainda, procurar a correlação entre os pesos da mesma camada e isso poderia ser estimado através da função de autocorrelação $r_x[k]$ de $\{x_i[n]\}$ definida como $r_x(k) = E\{x_i[n]x_i[n-k]\}$ [20]. A autocorrelação de k *lags* (atrasos) significa a correlação entre os pesos que estão separados por k. O coeficiente de *Pearson* pode ser separado em fraco e forte:

- Fraco: $0 \leq |\rho_c| < 0,5$.
- Forte: $0,5 \leq |\rho_c| < 1,0$.

Para analisar a correlação entre cada camada dos modelos foram utilizadas as redes: *Densenet*, *Efficientnet*, *googLenet*, *Mobilenet*, *Shufflenet* e *Squeezenet*, sendo que as três últimas foram usadas também para a medida de autocorrelação.

Marcos Tonin, Dept. Engenharia Elétrica, Universidade de Brasília, Brasília-DF, e-mail: 190122617@aluno.unb.br; Ricardo L. de Queiroz, Dept. Ciências da Computação, UnB, Brasília-DF, e-mail: queiroz@ieee.org.

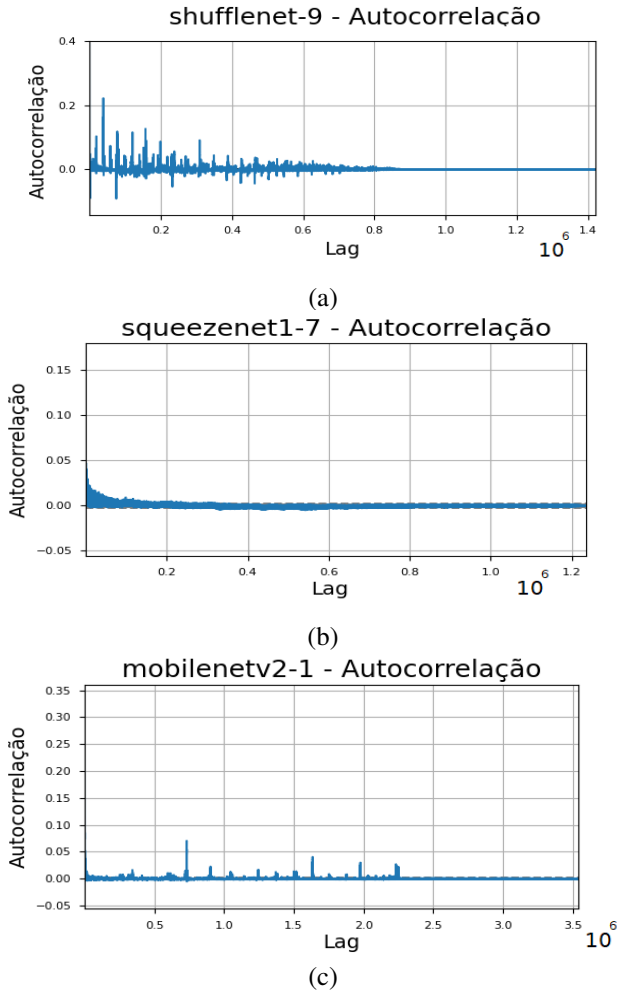

Fig. 1. Função de autocorrelação dos pesos de várias redes.

A partir da Fig. 1 podemos averiguar que a grande maioria dos $r_x[k]$ são fracos, próximos de zero para todos os pesos das redes: *Shufflenet, Squeezenet* e *Mobilenet*.

TABELA I

PERCENTUAL DE CLASSIFICAÇÃO DO ρ_c ENTRE CAMADAS POR REDE.

	Dense net	*Efficient net*	*Lenet*	*Mobile net*	*Shuffle net*	*Squeeze net*
Fraco	99,53%	99,95%	99,73%	99,26%	99,54%	99,40%
Forte	0,47%	0,005%	0,27%	0,74%	0,46%	0,60%

A Tabela I mostra o percentual de coeficientes fracos e fortes, com a finalidade de verificar se há predominância da correlação entre as camadas. Nela, vemos que a maioria dos coeficientes de correlações são fracos; os fortes são esporádicos, menores que 1%. Com isso, chegamos à conclusão que os pesos são praticamente descorrelacionados e, dessa maneira, não se indica realizar a compressão por transformadas, predição ou quantização vetorial.

III. QUANTIZAÇÃO ESCALAR DOS PESOS

A Fig. 2 apresenta a função densidade de probabilidade de todos os pesos da rede *Squeezenet*. O gráfico tem o intuito de mostrar como esses são distribuídos em uma rede neural. Nele é possível ver que os valores estão concentrados em sua maioria perto do zero. As distribuições dos pesos das outras redes têm comportamento similar.

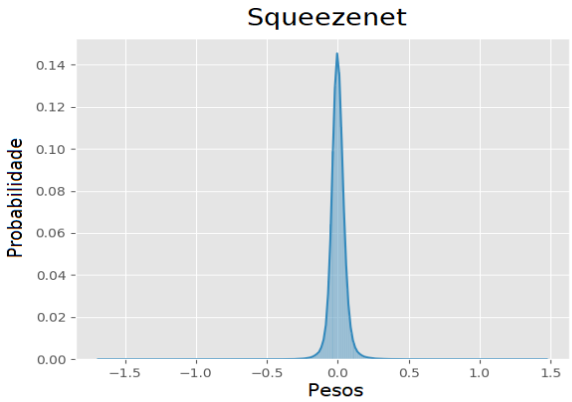

Fig. 2. Distribuição dos pesos do modelo *Squeezenet*.

Analisamos a similaridade das distribuições dos pesos dos modelos com outras conhecidas, tais como: *Alpha, Cauchy, Exponencial, Logística, Gamma, Laplace, Gaussiana* e *Uniforme* [20], [21], [22]. Para examinar qual distribuição se aproxima mais da rede neural foram usadas duas métricas: soma de erros quadrado (SSE) e *divergência de Kullback-Leibler (D_{KL})*.

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

$$D_{KL}(P||Q) = \sum_{i=1}^{n} P(i) \log \frac{P(i)}{Q(i)}, \tag{2}$$

na Equação 1, y_i é um ponto referente à distribuição do modelo e y_i^d refere-se às distribuições conhecidas. Na segunda (Eq. 2), $P(i)$ é a distribuição de probabilidade dos modelos conhecidos e $Q(i)$ é a probabilidade de distribuição do modelo de cada rede. Tanto a SSE, quanto a D_{KL} medem a semelhança entre duas distribuições, quanto mais semelhantes as distribuições, menores os valores de SSE e D_{KL}.

TABELA II

SSE ENTRE AS DISTRIBUIÇÕES PADRÃO E A DE CADA MODELO.

Redes	1^{st} **Match**	2^{nd} **Match**	3^{rd} **Match**
Caffenet	Laplaciana	Logística	Gaussiana
Densenet	Laplaciana	Alpha	Gaussiana
efficientnet	Laplaciana	Logística	Gaussiana
googLenet	Laplaciana	Logística	Gaussiana
Mobilenet	Laplaciana	Gaussiana	Logística
rcnn	Gaussiana	Laplaciana	Alpha
resnet	Laplaciana	Logística	Gaussiana
Shufflenet	Cauchy	Laplaciana	Logística
Squeezenet	Logística	Laplaciana	Gaussiana

TABELA III

D_{KL} ENTRE AS DISTRIBUIÇÕES PADRÃO E A DE CADA MODELO.

Redes	1^{st} **Match**	2^{nd} **Match**	3^{rd} **Match**
Caffenet	Gaussiana	Laplaciana	Cauchy
Densenet	Cauchy	Laplaciana	Alpha
efficientnet	Laplaciana	Logística	Gaussiana
googLenet	Laplaciana	Logística	Gaussiana
Mobilenet	Laplaciana	Logística	Alpha
rcnn	Gaussiana	Laplaciana	Alpha
resnet	Laplaciana	Logística	Gaussiana
Shufflenet	Cauchy	Laplaciana	Logística
Squeezenet	Laplaciana	Logística	Gaussiana

As Tabelas II e III mostram as três distribuições que mais se aproximam dos modelos de redes. A coluna 1^{st} *Match* é a distribuição com melhor resultado; a 2^{nd} *Match* é a

segunda melhor e a 3^{rd} *Match* é a terceira mais adequada. Ao analisar as informações referentes à SSE, é possível verificar que a distribuição *Laplace* é uma boa aproximação, uma vez que ela possui a primeira (em 55,56% dos modelos) ou segunda melhor aproximação (em 44,44% dos 9 modelos). A métrica D_{KL}, de maneira análoga à SSE, possui a distribuição *Laplaciana* entre os melhores modelos: em 6, a *Laplaciana* possui melhor aproximação e, em 3 é a segunda melhor. Com a codificação entrópica dos valores quantizados podemos utilizar os resultados de Sullivan. O autor relata que para distribuições *Laplacianas*, a quantização ótima é aproximada pela quantização uniforme com zonas mortas (*deadzone*) [23].

A quantização é usada para fornecer uma quantia discreta para um valor, de forma que seja possível reconstruir aproximadamente o valor original. A quantização uniforme possui um espaçamento igual entre cada nível existente. Podemos dividi-la em: *midtread* e *midrise* [24], modificando como o valor é reconstruído, tal qual descrito na Tabela IV. A versão com *deadzone* não é exatamente uma quantização uniforme, uma vez que nem todos os intervalos são iguais. Nessa, o nível em torno do zero tem um intervalo diferente, como descrito na Tabela IV.

TABELA IV

FÓRMULAS PARA QUANTIZAÇÃO UNIFORME.

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

Na Tabela IV, X representa o valor de um peso a ser quantizado, X_q o valor que será passado para o codificador e $\widehat{X}$ o valor reconstruído pelo codificador. Δ é o tamanho do passo de quantização. A função s(X) retorna 1 se o número for positivo e, caso contrário, -1. σ define o tamanho relativo de *deadzone* em relação ao passo de quantização. Note, $\sigma = 0.0$ é quantizador *midrise*, enquanto que o quantizador *midtread* tem $\sigma = 0.5$. O operador $\lceil X\rceil$ é conhecido por teto e retorna o arredondamento superior de X.

Em relação à quantização não uniforme, o tamanho de cada passo é desigual. Uma das formas mais conhecidas de fazê-las é usando funções logarítmicas. Na quantização não uniforme, os passos perto da origem são menores e, ao se distanciarem, serão maiores. Assim, pretende-se priorizar os valores perto da origem em detrimento dos mais distantes. Realizamos testes envolvendo a quantização não uniforme "lei μ"[24].

Outra forma de realizar a quantização não uniforme é com a utilização de diferentes formatos de ponto flutuante. O formato padrão tem 32 bits, no entanto, há a versão de 16 bits definida em IEEE 754 [6] (*halfprecision*) e a de 8 bits, conhecida como *minifloat*. Então, para cada ponto flutuante representado em 32 bits, foi identificado o valor correspondente para 16 e 8 bits. Além das versões *halfprecision* e *minifloat*, foram definidas versões de *floats* com diferentes números de bits (12, 10, 7, 6, 5, 4 e 3) seguindo o formato de ponto flutuante:

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

em que o sinal é representado em um bit, o expoente em p bits e, por fim, a mantissa usa q bits.

IV. RESULTADOS

Os testes foram realizados com o uso do *dataset* de validação do *ImageNet Large Scale Visual Recognition Challenge 2012* (*ILSVRC 2012*) que é composto por 50000 imagens [29], essas possuem uma classificação dentre 1000 possíveis. As redes utilizadas foram: *squeezenet* [25], *shufflenet* [26], *mobilenet* [27], *caffenet* [27], *GoogLenet* [9] e *efficientnet* [28] (todas do tipo CNN e no formato ONNX). Para analisar o funcionamento da rede foram usadas as medidas de acurácia Top 1 (verifica se a saída mais provável produzida pela rede corresponde ao *ground truth*) e acurácia Top 5 (verifica se uma das 5 saídas mais prováveis produzidas pela rede corresponde ao *ground truth*). Outra grandeza importante observada foi a entropia: uma maneira de medir o grau de incerteza de uma fonte e estimar quantos bits um codificador gastaria para codificar cada peso [24].

TABELA V

ENTROPIA E RESULTADOS DAS MÉTRICAS PARA OS MODELOS.

Modelo	Entropia	Acc Top 5(%)	Acc Top 1(%)
caffenet	25,213	79,522	56,264
efficientnet-lite	23,513	93,684	77,734
googLenet	22,653	88,34	67,774
mobilenet v2	21,673	88,934	69,3
shufflenet	20,391	68,134	42,422
squeezenet	20,221	77,138	53,77
Média	**22,277**		

A Tabela V mostra os resultados de entropia e acurácias (em porcentagem) para cada modelo original de rede neural, nessa Tabela vemos que a média de 22,28 bits/peso. Esses resultados foram usados como base para a comparação após a quantização. O primeiro teste consistiu na comparação entre a quantização uniforme (*midrise* e *midtread*) e a quantização não uniforme (representações de *floats* e a função "lei μ").

Ao analisar os gráficos das Fig. 3, percebe-se que a quantização *midrise* e *midtread* tem melhor performance em relação às quantizações não uniformes. Constata-se que a *midrise* tem um pequeno ganho em relação a *midtread*. O caso especial é a rede *mobilenet* (Figs. 3(k) e 3(l)) em que a quantização não uniforme (lei μ) tem um melhor desempenho.

Depois disso, foram contrapostos os modelos de quantização uniforme citados acima com os modelos *deadzone*, utilizando o coeficiente σ como: 0.1, 0.25, 0.4 e 0.7. Lembrando que *midrise* temos *deadzone* com $\sigma = 0.0$ e, *midtread*, $\sigma = 0.5$.

Na Fig. 4, excetuando-se o caso *Mobilenet*, um melhor resultado é alcançado para *midrise*, *midtread*, ou *deadzone* com coeficiente 0.4 ou 0.7. Para a maioria das redes, os resultados não variam muito, elas se comportam de maneira parecida com as variações de σ. A rede *Mobilenet* possui maior diferença entre os comportamentos de acordo com o coeficiente σ. Nesse caso, constata-se melhor um resultado para *deadzone* 0.7, mesmo comparado à quantização não-uniforme seguindo a "lei μ".

Em síntese, nos resultados das Figs. 3 e 4, temos: a *caffenet*, com método o *midrise*, com uma taxa entrópica de 1,8 e uma diferença de no máximo 0,8% na acurácia Top 5. Para a *efficientnet-lite*, com método *deadzone* $\sigma = 0.7$, houve um prejuízo de 0,75% na acurácia Top 1 e as taxas reduziram para 6,5 bits/peso. A *googLenet* alcança, pelo método *deadzone* $\sigma = 0.4$, taxas próximas a 3 bits/peso, e com uma degradação

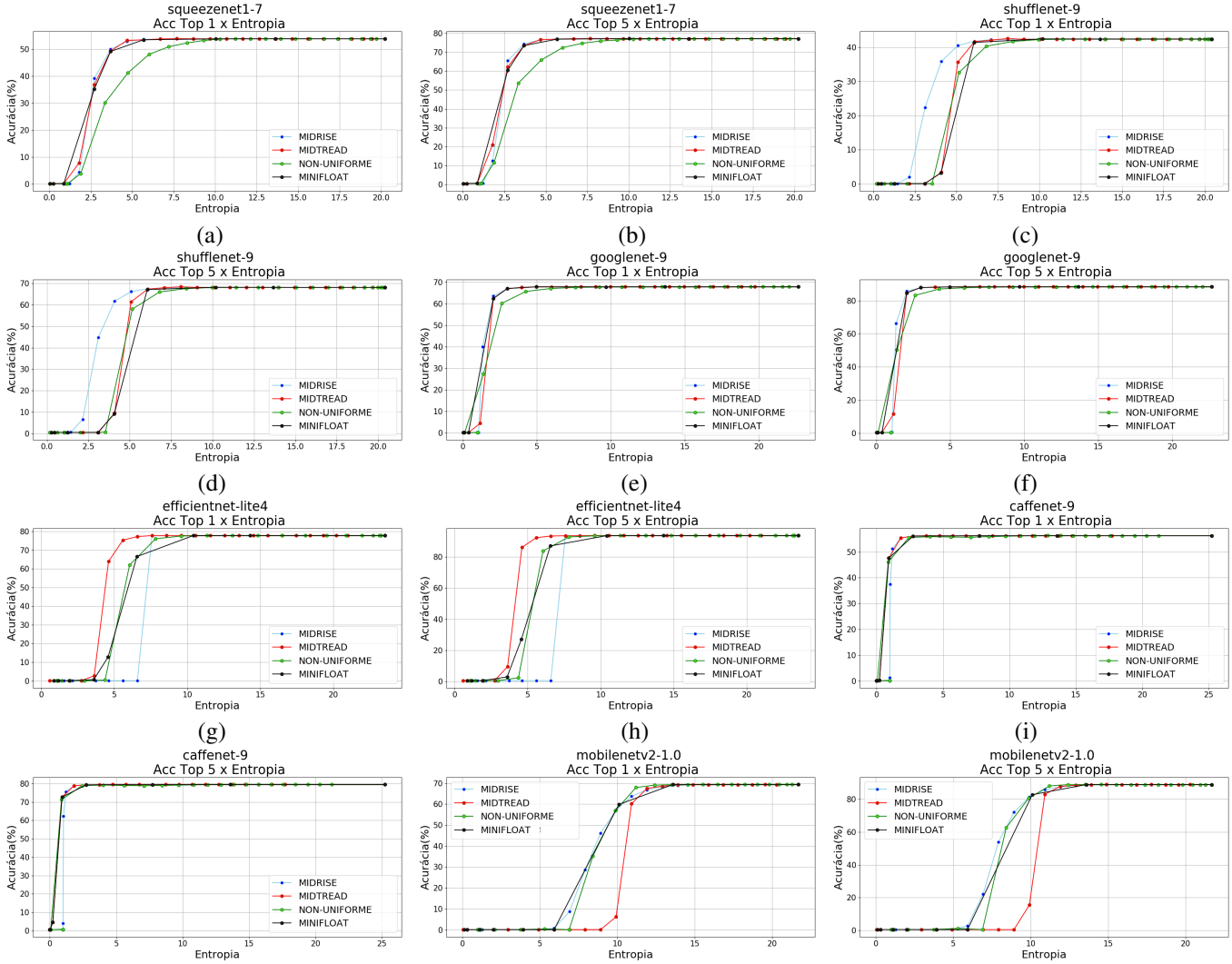

Fig. 3. Comparações taxa e distorção (entropia × acurácia) entre formas de quantização uniforme e não uniforme para diversas redes.

de 0,75% da rede. A rede mais discrepante, *mobilenet*, teve um bom resultado alcançado por *deadzone* $\sigma = 0.7$, com um prejuízo máximo de 1% para a acurácia Top 1 e taxas reduzidas para 11,9 bits/peso. Quando utilizado o método *midtread*, a *Shufflenet* teve a acurácia Top 5 prejudicada em 0,65% e sua entropia indo para perto de 6 bits. Por último, a *Squeezenet* tem um melhor resultado por meio da *midrise*, chegando a entropia de 4,67, piorando os resultados da rede em, no máximo, 0,56%.

V. Conclusões

Os resultados finais mostram que para a maioria das redes é possível chegar em taxas perto de 5 bits por peso sem acarretar grandes perdas, entre 1% e 2% tanto para acurácia Top 1 quanto Top 5. Nas versões originais das redes estudadas havia em média 22,27 bits/peso de entropia, conforme Tabela V; podendo, portanto, chegar a uma entropia por pesos 4 vezes menores. Vimos que é possível utilizar menos bits para representar os pesos sem comprometer o funcionamento da rede. Com isso, podemos estabelecer formatos de distribuição e intercâmbio mais compactos.

Por outro lado, o método escolhido não impede a combinação com outros. Poderíamos, por exemplo, embora não seja o intuito deste artigo, aliá-lo a métodos que envolvam retreino, como quantização em camadas [14], *pruning* [11], [12], [13], ou combiná-lo com *K-means* [17]. Por fim, a reconstrução sub-ótima utilizada nos quantizadores uniformes da Tabela IV será substituída por reconstrução otimizada em trabalhos futuros.

Referências

[1] I. Banerjee et al, "Comparative effectiveness of convolutional neural network (CNN) and recurrent neural network (RNN) architectures for radiology text report classification", *Artificial intelligence in medicine 97*, pp. 79–88, 2019.

[2] F. L. Bai, "Onnx: Open neural network exchange", https://github.com/onnx/onnx, 2019.

[3] A. Paszke et al, "Automatic differentiation in PyTorch", 2017.

[4] F. Chollet et al, "Keras", https://keras.io, 2015.

[5] M. Abadi et al, *TensorFlow: Large-scale machine learning on heterogeneous systems*, https://tensorflow.org, 2015.

[6] "IEEE Standard for Floating-Point Arithmetic", *IEEE Std 754-2008*, pp.1–70, 2008.

[7] M. Sandler et al, "MobileNetV2: Inverted Residuals and Linear Bottlenecks", *Proc. of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520, 2018.

[8] K., Alex, I. Sutskever, e G. E. Hinton, "Imagenet classification with deep convolutional neural networks", *Advances in neural information processing systems*, pp. 1097-1105, 2012.

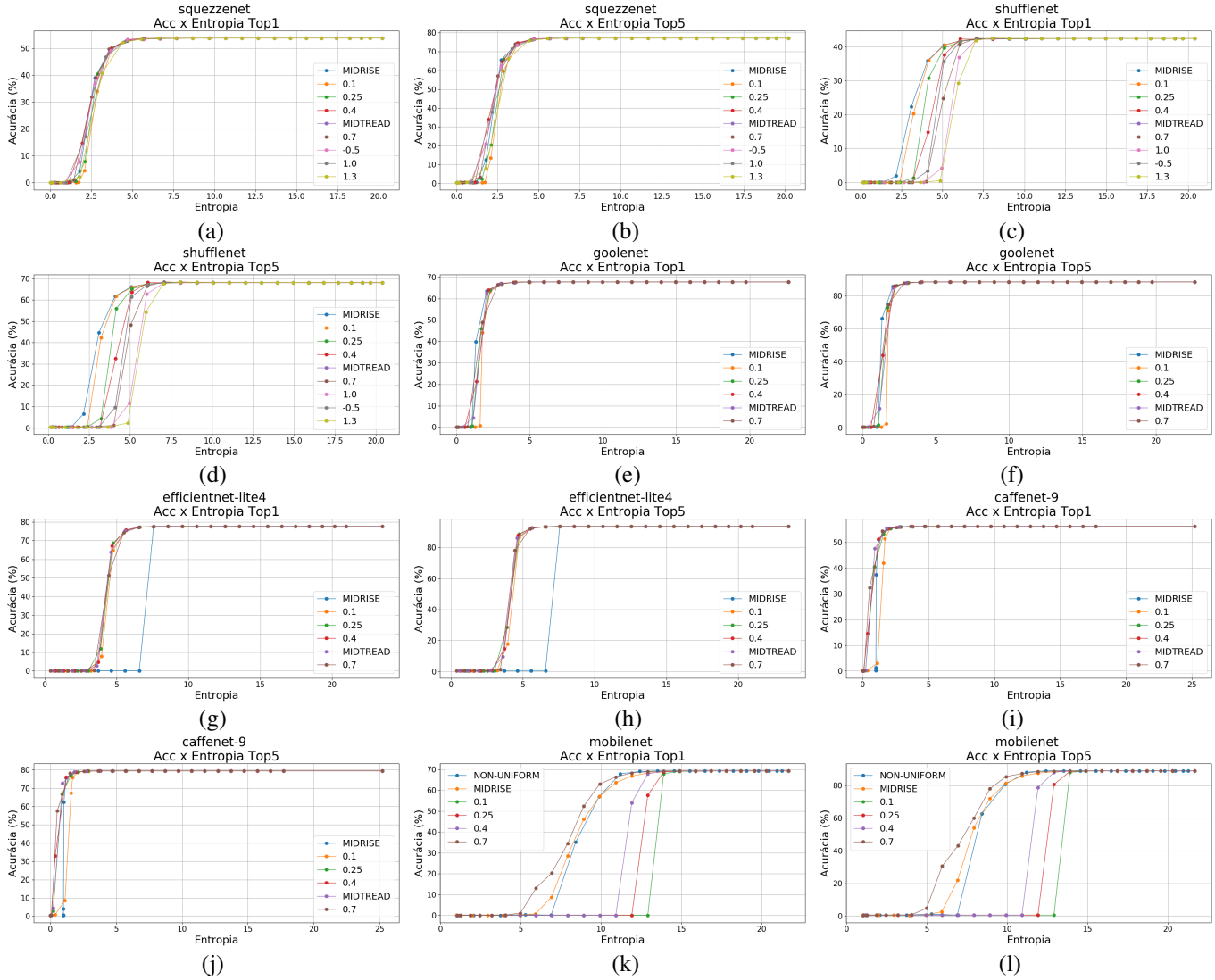

Fig. 4. Comparações taxa e distorção (entropia × acurácia) entre diferentes tamanhos de *deadzone* para várias redes.

[9] C. Szegedy, et al, "Going deeper with convolutions", *Proc. of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

[10] K. Simonyan e A. Zisserman, "Very deep convolutional networks for large-scale image recognition", *arXiv:1409.1556*, 2016.

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

[12] L. Li, Z. Li, Y. Li, B. Kathariya e S. Bhattacharyya, "Incremental Deep Neural Network Pruning based on Hessian Approximation," *Data Compression Coference*, p. 590–590, 2019.

[13] W. B. Zhao e Y. Li, "Fuzzy Pruning for Compression of Convolutional Neural Networks", *IEEE International Conference on Fuzzy Systems*, IEEE, pp. 1–5, 2019.

[14] X. Zhu, W. Zhou e H. Li, "Adaptive Layerwise Quantization for Deep Neural Network Compression", *IEEE International Conference on Multimedia and Expo (ICME)*, pp. 1–6, 2018.

[15] J.-K. Kim, M.-Y. Lee, B.-J. Kim e J.-H. Lee, "An Efficient Pruning and Weigth Sharing Method for Neural Network", *IEEE Conference o Consumer Eletronics-Assia (ICCE-Asia)*, pp 1–2, 2016.

[16] J. Faraone, N. Fraser, M. Blott e P. H. W. Leong, "SYQ: Learning Symmetric Quantization For Efficient Deep Neural Networks", *The IEEE Conference on Computer Vision and Pattern Reciognition*, pp. 4300-4309, 2018

[17] S. Seo e J. Kim, "Hybrid Approach for Efficient Quantization of Weights in Convolutional Neural Networks", *IEEE International Conference on Big Data and Smart Computing*, IEEE, pp. 638–641 , 2018.

[18] E. Dupuis, D. Novo, I. O'Connor e A. Bosio, "Sensitivity Analysis and Compression Opportunities in DNNs Using Weight Sharing", *23rd International Symposium on Design and Diagnostics of Electronic Circuits & Systems*, IEEE, pp. 1–6, 2020.

[19] L. Sheugh e S. H. Alizadeh, "A note on pearson correlation coefficient as a metric of similarity in recommender system", *2015 AI & Robotics (IRANOPEN)*, pp. 1–6, 2015.

[20] A. Papoulis, *Probability, Random Variables, and Stochastic Processes* New York: McGraw-Hill, pp. 92-104, 1984.

[21] A. A. Salvia, "Reliability applications of the Alpha Distribution", *IEEE Transactions on Reliability*, Vol. R-34, No. 3, pp. 251–252, 1985.

[22] J. DeCani e R. Stine, *A Note on Deriving the Information Matrix for a Logistic Distribution*, The American Statistician, pp. 220–222, 1986.

[23] G. J. Sullivan, "Efficient scalar quantization of exponential and Laplacian random variables", *IEEE Transactions on Information Theory*, IEEE, vol. 42, no. 5, pp. 1365–1374, 1996.

[24] W.A Pearlman e A. Said, *Digital Signal Compression: Principles and Practice*. Cambridge University Press, 2011.

[25] F. N. Iandola et al, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and < 0.5 MB model size", *arXiv:1602.07360*, 2016.

[26] N. Ma, X. Zhang, H. T. Zheng e J. Sun, "Shufflenet v2: Practical guidelines for efficient cnn architecture design", *Proceedings of the European conference on computer vision*, ECCV, pp. 116–131, 2018.

[27] Y. Jia et al, "Caffe: Convolutional architecture for fast feature embedding",*Proceedings of the 22nd ACM international conference on Multimedia*, pp. 675–678, 2014.

[28] M. Tan e Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks", *International Conference on Machine Learning*, PMLR, pp. 6105-6114, 2019.

[29] O. Russakovsky et al, "Imagenet large scale visual recognition challenge", *International journal of computer vision*, pp. 211–252, 2015.