# Nova estratégia de treinamento para compressão de modelo de aprendizado profundo aplicada a classificação automática de modulações

Mateus A. S. de S. Goldbarg e Marcelo A. C. Fernandes

***Resumo*— Técnicas de aprendizado profundo, como redes neurais profundas (DNNs), têm sido utilizadas com sucesso em muitos problemas de classificação automática de modulações. No entanto, os algoritmos de aprendizado profundo exigem um grande esforço computacional e isto pode ser um gargalo para detecção de modulações em tempo real. Assim, este artigo propõe uma nova estratégia de treinamento que minimiza simultaneamente as perdas de poda e quantização no treinamento de modelos compactados para reduzir a complexidade computacional de DNNs. A estratégia de treinamento proposto foi aplicada no treinamento de classificação automática de sete modulações expressas como OOK, 4ASK, 16PSK, 64APSK, AM-DSB-WC, AM-DSB-SC e FM. Uma redução substancial de pesos e operações da DNN é demonstrada, mantendo alta precisão de classificação.**

***Palavras-Chave*— Aprendizagem profunda, DNN, Detecção automática de modulação, Quantização consciente, Poda consciente.**

***Abstract*— Deep learning techniques, such as deep neural networks (DNNs), have been used successfully in many automatic modulation classification problems. However, deep learning algorithms require a significant computational effort, and this can be a bottleneck to detect modulations in real time. So, this article proposes a new training strategy that minimizes simultaneously pruning and quantization losses in the training of compressed models to reduce the computational complexity of DNNs. Furthermore, the training strategy was specifically applied without the training of automatic classification of seven express modulations such as OOK, 4ASK, 16PSK, 64APSK, AM-DSB-WC, AM-DSB-SC, and FM. As a result, a substantial reduction in DNN weights and operations has been demonstrated while maintaining high classification accuracy.**

***Keywords*— Deep Learning, DNN, Automatic Modulation Classification, Aware Quantization, Aware Pruning.**

## I. INTRODUÇÃO

A classificação automática de modulação (*automatic modulation classification* - AMC) é um problema clássico nas comunicações sem fio modernas. Um dos objetivos da AMC é entender e rotular o espectro de rádio em cenários de comunicação não cooperativos, o que facilitará muito a detecção de falhas, monitoramento de interferência de espectro e acesso dinâmico ao espectro. A AMC é uma tecnologia em rápida evolução, que pode ser empregada em estruturas de rádio definidas por software, especialmente na tecnologia 5G e 6G [1], [2], [3], [4], [5]. Recentemente, métodos AMC baseados em aprendizagem profunda (*deep learning* - DL) têm atraído atenção significativa devido ao seu desempenho. A técnicas baseadas em DL fornecem uma solução altamente promissora para lidar com dados brutos e de alta dimensão, em vez da estratégia de extração de características feita normalmente em técnicas tradicionais de AMC [1], [2], [3], [4], [5], [6].

Acelerar a computação de Redes Neurais Profundas (*Deep Neural Networks* – DNN) tem sido um foco recente no aprendizado de máquina, e sua literatura tem crescido rapidamente. As estratégias utilizadas para aceleração de inferência podem ser classificadas nas baseadas em algoritmos e baseadas em sistema. As abordagens de aceleração baseadas em algoritmo incluem processamento paralelo (paralelismo de dados e modelo), compressão de modelo (remoção de pesos e quantização de valores de peso e ativação), exploração de esparsidade (esparsidade em nível de valor e em nível de bit) e redução de modelo por meio de aproximação [7], [8], [9], [10], [11], [12], [13], [14], [15]. As abordagens de aceleração baseadas em sistema incluem sistemas de memória de alta largura de banda para minimização do tempo de acesso, aceleradores de hardware (ASIC e FPGA), sistemas de computação distribuída [16], [17], [18] e otimizações de compilador, como eliminação de expressões comuns [19].

A estratégia baseada em algoritmos usando compressão de modelos (ou dos pesos), foco deste trabalho, é uma forma comum de reduzir o número de operações DNN e, ao mesmo tempo, ter um impacto mínimo no resultado final, ou seja, na precisão da classificação. A abordagem também chamada de compressão consciente pode melhorar o desempenho de DNN em situações de tempo real para classificação automática de modulação.

A compressão convencional aborda a poda [7], [8] e quantiza os pesos do modelo [9], [10], com poda seguida de quantização sobre as épocas de treinamento [20], [21], [15]. Neste artigo, em contraste, propomos um novo esquema de compressão de modelo que minimiza simultaneamente as perdas de poda e quantização durante o treinamento com poda seguida de quantização ($P \rightarrow Q$ *iteration*) para cada iteração de treinamento.

O esquema de estratégia de treinamento proposto foi aplicado para treinar modelos compactados para classificação automática de modulações. Os resultados mostraram uma redução substancial dos pesos e operações da DNN para vários tamanhos de quantização e poda mantendo a acurácia da

Mateus A. S. de S. Goldbarg, Laboratório de Aprendizagem de Máquina e Instrumentação Inteligente (LAMII/IMD), Universidade Federal do Rio Grande do Norte, Natal-RN, e-mail: mateus.goldbarg@dca.ufrn.br; Marcelo A. C. Fernandes, Departamento de Engenharia da Computação e Automação, Universidade Federal do Rio Grande do Norte, Natal-RN, e-mail: mfernandes@dca.ufrn.br.

classificação.

## II. Proposta

### A. Compressão dos pesos por quantização consciente

A Figura 1 mostra a arquitetura do treinamento com compressão dos pesos baseado quantização consciente (*aware quantization*) [9], [10]. No esquema, $n$ representa a $n$-ésima iteração, $z^{-1}$ é um atraso unitário, $\mathbf{W}(n)$ representa os pesos CNN de todas as camadas, $\mathbf{X}(n)$ é a entrada da CNN, $\mathbf{Y}_{ref}(n)$ é a saída desejada (rótulo), $\mathbf{Y}(n)$ é a saída da CNN, $\mathbf{E}(n)$ é uma métrica de erro (perda) entre $\mathbf{Y}(n)$ e $\mathbf{Y}_{ref}(n)$ e $\mathbf{G}(n)$ representa a métrica de gradiente de todas as camadas. Em cada mini-batch uma cópia dos pesos, $\mathbf{W}(n)$, é quantizada e passada para o estágio *forward* através da variável $\mathbf{C}(n)$.

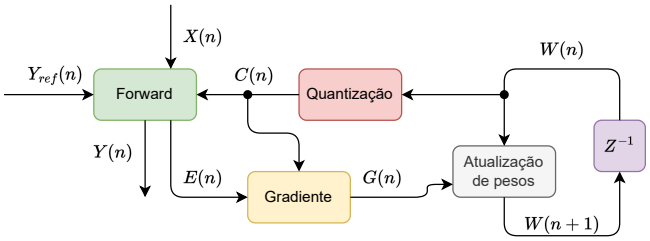



Fig. 1: Arquitetura do esquema de treinamento com compressão dos pesos baseado na quantização consciente (*aware quantization*).

A compressão por quantização produz inteiros de largura de bits baixa para representar pesos. Ou seja, $\mathbf{W}(n)$ é representado por um dos $M$ níveis discretos possíveis que podem ser representados por $b$ bits. Existem muitas técnicas de quantização na literatura, como apresentado em [9], [10], [14], [13]. Um esquema comum é a quantização uniforme, com $M = 2^b - 1$. Neste caso, a variável $\mathbf{C}(n)$ de cada $k$-ésima camada é expressa como

$$\mathbf{C}_k(n) = Q\left(\mathbf{W}_k(n), q_k\right) = \left\lceil \frac{\mathbf{W}_k(n)}{q_k} \right\rfloor \times q_k \tag{1}$$

onde $Q(\cdot,\cdot)$ é a função de quantização, $\lceil \cdot \rfloor$ é a operação de rodada, e $q_k$, o fator de escala, pode ser expresso como

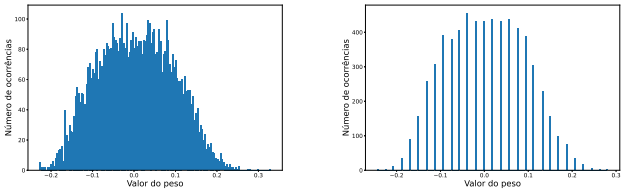
$$q_k = \frac{\max\left\{|\mathbf{W}_k(n)|\right\}}{2^{b-1} - 1}. \tag{2}$$

As figuras 2(a) e 2(b) ilustram um exemplo de pesos de uma dada camada, $\mathbf{W}_k(n)$, de uma CNN. Os pesos são apresentados antes e depois da quantização com $b = 5$ ($M = 31$), respectivamente.

### B. Compressão dos pesos por poda consciente

Na poda (ver Figura 3), os pesos para cada $k$-ésima camada, $\mathbf{W}_k(n)$, com valores pequenos (abaixo de um limite) são zerados, ou seja ,

$$\mathbf{C}_k(n) = P\left(\mathbf{W}_k(n), \beta_k\right) = \begin{cases} \mathbf{W}_k(n) & \text{if } |\mathbf{W}_k(n)| \geq \beta_k \\ 0 & \text{if } |\mathbf{W}_k(n)| < \beta_k \end{cases} \tag{3}$$




(a) Pesos antes da compressão. (b) Pesos quantizados com 5 bits.

Fig. 2: Histograma do valores dos pesos associado a um exemplo de compressão de pesos por quantização consciente com $b = 5$ ($M = 31$) aplicada à uma camada de uma rede CNN.

onde $P(\cdot,\cdot)$ é a função de poda e $\beta_k$ é o limite de corte de poda da $k$-ésima camada, que pode ser expresso como

$$\beta_k = \alpha_k \times \sigma_k \tag{4}$$

onde $\sigma_k$ representa o desvio padrão do $\mathbf{W}_k(n)$. As figuras 4(a) e 4(b) ilustram um exemplo de pesos de uma $k$-ésima camada de uma CNN, $\mathbf{W}_k(n)$, antes de depois da poda consciente. Os pesos são apresentados antes e após a poda, respectivamente, com o limite de corte da poda, $\beta_k = 0,5 \times \sigma_k$ ($\alpha_k = 0.5$).

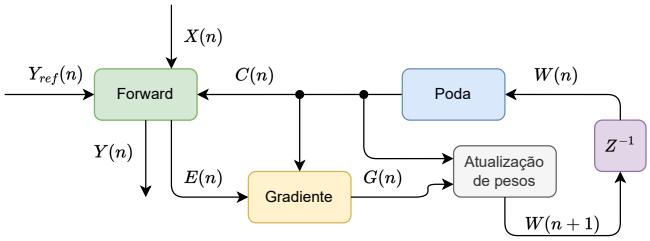
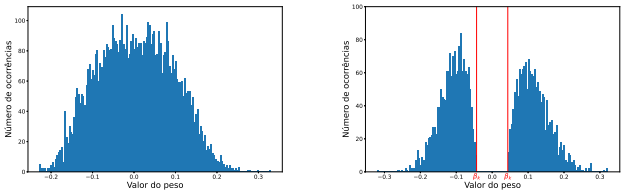



Fig. 3: Arquitetura do esquema de treinamento com compressão dos pesos baseado na poda consciente (*aware pruning*).




(a) Pesos antes da compressão (b) Poda consciente com $\alpha = 0.5$

Fig. 4: Histograma do valores dos pesos associado a um exemplo de compressão de pesos por poda consciente com $\alpha_k = 0.5$ ($\beta_k = 0.5 \times \sigma_k$) aplicada à uma camada de uma rede CNN.

### C. Proposta de compressão consciente com poda seguida de quantização por iteração ($P \rightarrow Q$ iteration)

A compressão proposta por este trabalho baseada na poda seguida de iteração de quantização, $P \rightarrow Q$ *iteration*, pode ser expressa como

$$\mathbf{C}_k(n) = Q\left(P\left(\mathbf{W}_k(n), \beta\right), q'_k\right) \tag{5}$$

Onde

$$q'_k = \frac{\max\{|\mathbf{W}_k(n)|\} - \beta_k}{2^{b-1}-1}. \quad (6)$$

O diagrama apresentado na Figura 5 ilustra o esquema de treinamento para a proposta de $P \rightarrow Q$ *iteration*, e as Figuras 6(a) e 6(b) mostram um exemplo de pesos de uma $k$-ésima camada, $\mathbf{W}_k(n)$, da um CNN. Os pesos são apresentados antes e depois do $P \rightarrow Q$ *iteration*, respectivamente.

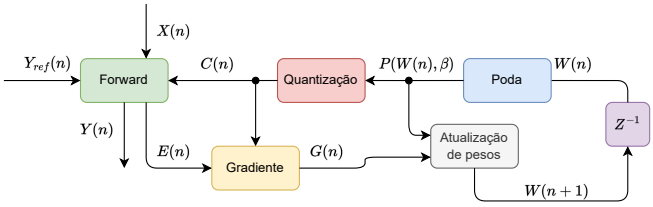
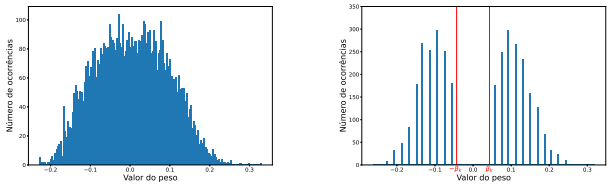



Fig. 5: Arquitetura do esquema de treinamento com compressão consciente baseado em poda seguida de quantização a cada iteração, $P \rightarrow Q$ *iteration*.




(a) Pesos antes da compressão

(b) Poda seguida de quantização com $\alpha = 0.5$ e 5 bits

Fig. 6: Histograma do valores dos pesos associado a um exemplo de compressão consciente dos pesos por poda seguida de quantização a cada iteração com $\alpha_k = 0.5$ e 5 bits aplicada à uma camada de uma rede CNN.

## III. Metodologia

### A. Dataset

Este estudo fez uso de um *dataset* de modulações apresentado em [22], [23], [6]. Este *dataset* inclui efeitos de canal simulados sintéticos e gravações pelo ar de 24 tipos de modulação digital e analógica com 26 níveis de relação sinal ruído, SNR, (entre $-20$ dB e 30 dB com um passo de 2). Os dados são armazenados no formato hdf5 como valores complexos (valores em fase e quadratura) em ponto flutuante, com 2 milhões de exemplos, cada um com 1024 amostras. As 24 modulações armazenadas no dataset são OOK, 4ASK, 8ASK, BPSK, QPSK, 8PSK, 16PSK, 32PSK, 16APSK, 32APSK, 64APSK, 128APSK, 16QAM, 32QAM, 64QAM, 128QAM, 256QAM, AM-SSB-WC, AM-SSB-SC, AM-DSB-WC, AM-DSB-SC, FM, GMSK, OQPSK. Para cada par de modulação e SNR existem 4096 realização, chamadas de *frames*, e cada *frame* possui 1024 amostras de I/Q ($2\times1024$). O *dataset* possui 2,555,904 *frames* no total.

A partir do *dataset* original descrito acima e detalhado em [22], [23] foi construido um segundo *dataset* formado por subconjunto de dados do *dataset* original. O novo dataset foi composto com 7 das modulações do *dataset* original (OOK, 4ASK, 16PSK, 64APSK, AM-DSB-WC, AM-DSB-SC e FM) a uma SNR $=$ 30 dB. Todos os resultados apresentados no artigo foram baseados neste novo *dataset* que possui 4096 *frames*, a uma SNR $=$ 30 dB para cada modulação. Cada *frame* possui 1024 amostras de I/Q ($2 \times 1024$).

### B. Arquitetura e Treinamento da DNN

Este trabalho fez uso de uma DNN do tipo CNN cuja a arquitetura está descrita na Tabela I. Esta arquitetura foi baseada em um dos modelos utilizados na validação do dataset apresentado em [22], [23], [6].

TABELA I: Arquitetura da CNN usada neste artigo com 7 camadas convolucionais e 3 camadas totalmente conectadas.

| **Camada** | **Descrição** | **Valores** |
|---|---|---|
| 1 | Entrada | $1024 \times 2$ |
| 2 | Conv1D($K_1 \times S_1$) | $K_1 = 40$ e $S_1 = 4$ |
| 3 | Batch Norm | - |
| 4 | ReLU | - |
| 5 | MaxPool1D($S_1$) | $S_1 = 2$ |
| 6 | Conv1D($K_2 \times S_2$) | $K_2 = 40$ e $S_2 = 4$ |
| 7 | Batch Norm | - |
| 8 | ReLU | - |
| 9 | MaxPool1D($S_2$) | $S_2 = 2$ |
| 10 | Conv1D($K_3 \times S_3$) | $K_3 = 40$ e $S_3 = 4$ |
| 11 | Batch Norm | - |
| 12 | ReLU | - |
| 13 | MaxPool1D($S_3$) | $S_3 = 2$ |
| 14 | Conv1D($K_4 \times S_4$) | $K_4 = 40$ e $S_4 = 4$ |
| 15 | Batch Norm | - |
| 16 | ReLU | - |
| 17 | MaxPool1D($S_4$) | $S_4 = 2$ |
| 18 | Conv1D($K_5 \times S_5$) | $K_5 = 40$ e $S_5 = 4$ |
| 19 | Batch Norm | - |
| 20 | ReLU | - |
| 21 | MaxPool1D($S_5$) | $S_5 = 2$ |
| 22 | Conv1D($K_6 \times S_6$) | $K_6 = 40$ e $S_6 = 4$ |
| 23 | Batch Norm | - |
| 24 | ReLU | - |
| 25 | MaxPool1D($S_6$) | $S_6 = 2$ |
| 26 | Conv1D($K_7 \times S_7$) | $K_7 = 40$ e $S_7 = 4$ |
| 27 | Batch Norm | - |
| 28 | ReLU | - |
| 29 | MaxPool1D($S_7$) | $S_7 = 2$ |
| 30 | Flatten | - |
| 31 | Dense($S_1$) | $S_1 = 128$ |
| 32 | Batch Norm | - |
| 33 | ReLU | - |
| 34 | Dense($S_2$) | $S_2 = 128$ |
| 35 | Batch Norm | - |
| 36 | ReLU | - |
| 37 | Dense($S_3$) | $S_3 = 7$ |
| 38 | Softmax | - |

Para o treinamento do modelo foi definido o *batch* de tamanho 64 com um otimizador do tipo SGD no qual 70% das amostras foram utilizadas para treinamento e 30% para teste. A taxa de trainamento (*learning rate*) foi de 0.05 o *momentum* de 0.9. O critério de *categorical crossentropy* foi utilizado como *Loss*.

## IV. Resultados e Discussões

Com o intuito de demonstrar as vantagens de se utilizar técnicas de compressão consciente de redes neurais profundas por poda, quantização e poda seguida de quantização, foram

obtidos resultados referentes a acurácia de teste, esparsidade e tamanho final do modelo para classificação automática de modulação. Esses resultados foram comparados em 20 diferentes cenários, onde os parâmetros do modelo foram quantizados em 32, 16, 8,4 e 3 bits e variando-se o valor do corte da poda, ou seja, $\alpha$ em 0.00, 0.25, 0.50 e 0.75.

A Tabela II apresenta os resultados das acurácias de teste de cada estratégia apresentada. A Tabela III mostra o resultado da esparsidade total do modelo ao final do treinamento. Sendo essa esparsidade obtida pela divisão entre quantidade de parâmetros iguais a zero e a quantidade total de parâmetros do modelo. A tabela IV contém os tamanho de cada um dos modelos (em Megabits) ao final do treinamento. O tamanho foi calculado a partir do tamanho, em bits, de cada parâmetro, multiplicados pela quantidade de parâmetros diferentes de zero.

TABELA II: Acurácias obtidas a partir do modelo comprimido para vários valores de tamanho de bits e $\alpha$.

| $\alpha$ \ bits | 32 bits | 16 bits | 8 bits | 4 bits | 3 bits |
|---|---|---|---|---|---|
| **0.00** | 97.06% | 96.93% | 97.00% | 84.18% | 83.89% |
| **0.25** | 96.13% | 96.68% | 95.77% | 92.03% | 84.00% |
| **0.50** | 96.70% | 96.41% | 96.43% | 84.31% | 83.41% |
| **0.75** | 94.46% | 95.20% | 95.18% | 90.55% | 83.36% |

TABELA III: Valores de esparcidade obtidos a partir do modelo comprimido para vários valores de tamanho de bits e $\alpha$.

| $\alpha$ \ bits | 32 bits | 16 bits | 8 bits | 4 bits | 3 bits |
|---|---|---|---|---|---|
| **0.25** | 42.17% | 47.31% | 48.11% | 45.85% | 38.50% |
| **0.50** | 66.26% | 60.82% | 61.28% | 60.61% | 41.60% |
| **0.75** | 74.78% | 69.75% | 76.11% | 68.41% | 62.44% |

TABELA IV: Tamanho do modelo comprimido para vários valores de tamanho de bits e $\alpha$.

| $\alpha$ \ bits | 32 bits | 16 bits | 8 bits | 4 bits | 3 bits |
|---|---|---|---|---|---|
| **0.00** | 3.18Mb | 1.59Mb | 0.80Mb | 0.40Mb | 0.30Mb |
| **0.25** | 1.84Mb | 0.84Mb | 0.41Mb | 0.26Mb | 0.18Mb |
| **0.50** | 1.08Mb | 0.62Mb | 0.31Mb | 0.16Mb | 0.17Mb |
| **0.75** | 0.80Mb | 0.48Mb | 0.19Mb | 0.13Mb | 0.11Mb |

As figuras 7, 8, 9 apresentam de forma gráfica os resultados obtidos das acurácias de teste, esparsidade dos modelos e seus tamanhos, respectivamente. Percebe-se que foi possivel alcaçar acurácias muito próximas a do modelo não comprimido em diversos cenários de compressão. Para a configuração onde os parâmetros são quantizados em 8 bits e $\alpha = 0.75$, obtve-se uma acurácia apenas $1.88\%$ menor que a do modelo não comprimido mas com um modelo quase 17 vezes menor. Quando os parâmetros foram quantizados novamente em 8 bits, mas com $\alpha = 0.50$, o modelo apresentou uma acurácia apenas $0.46\%$ menor que a do modelo não comprimido, porém mais de 10 vezes menor.

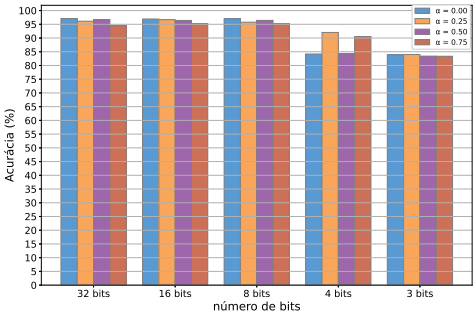



Fig. 7: Acurácias obtidas a partir do modelo comprimido para vários valores de tamanho de bits e $\alpha$.

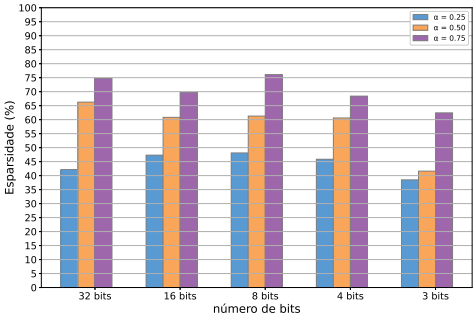



Fig. 8: Valores de esparcidade obtidos a partir do modelo comprimido para vários valores de tamanho de bits e $\alpha$.

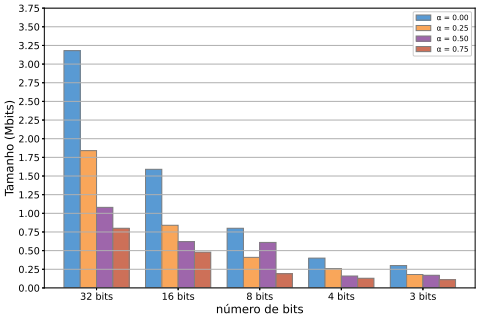
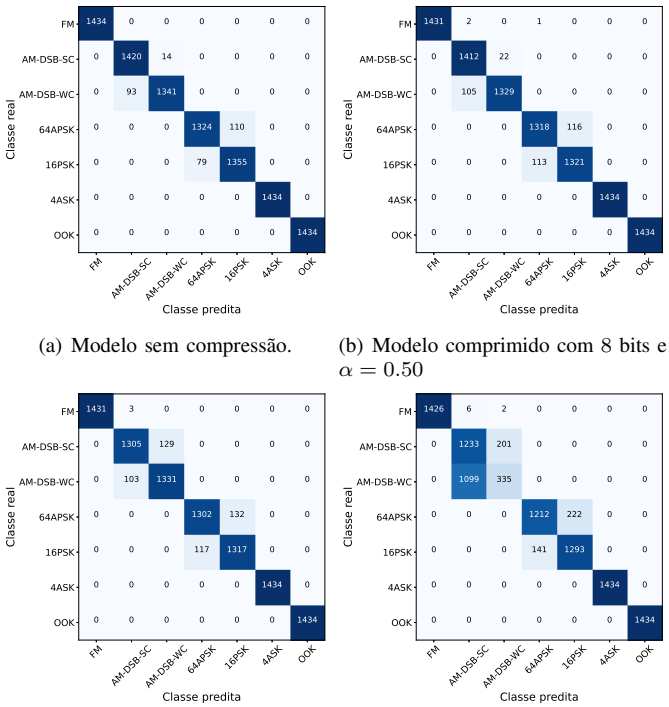



Fig. 9: Tamanho do modelo comprimido para vários valores de tamanho de bits e $\alpha$.

Em contrapartida, também pode-se observar que a compressão excessiva de parâmetros resulta numa queda significativa da acurácia do modelo como visto para o modelo quantizado em 3 bits e $\alpha = 0.75$. A figura 10 apresenta as matrizes de confusão de alguns modelos obtidos discutidos, afim de detalhar a acurácia para modulação.

A Tabela V apresenta um comparativo entre as técnicas mais convencionais de compressão de pesos e a nova estratégia proposta. A acurácia de teste para o modelo 1, comprimido com 8 bits e $\alpha = 0.5$, e para o modelo 2, comprimido com




(a) Modelo sem compressão.

(b) Modelo comprimido com 8 bits e $\alpha = 0.50$

(c) Modelo comprimido com 8 bits e $\alpha = 0.75$

(d) Modelo comprimido com 3 bits e $\alpha = 0.75$

Fig. 10: Matrizes de confusão de alguns modelos obtidos apos o treinamento.

8 bits e $\alpha = 0.75$, utilizando as estratégias mais convencionais ficaram muito próximas as dos modelos utilizando a nova estratégia com os mesmos hiper parâmetros, porém, a compressão utilizando a nova estratégia foi mais eficaz na diminuição do tamanho da rede, resultando no primeiro modelo $10.27\%$ menor e o segundo modelo $11.95\%$ menor em relação ao modelo original.

TABELA V: Comparativo entre a nova estratégia propósta e a estratégia convencional.

| | Estratégia convencional | | Estratégia propósta | |
|---|---|---|---|---|
| | Acurácia | Tamanho | Acurácia | Tamanho |
| Modelo 1 | 96.18% | 0.40Mb | 96.43% | 0.31Mb |
| Modelo 2 | 96.67% | 0.29Mb | 95.18% | 0.19Mb |

## V. Conclusões

Este trabalho propõem um esquema de treinamento para compressão consciente dos pesos de uma DNN para aplicações de classificação automática de modulação. A proposta aplica uma poda seguida de quantização consciente dos pesos a cada iteração. A estratégia proposta foi validada com um dataset composto de realizações de sinais de sete modulações digitais e analógicas. Os Experimentos foram realizados para vários tamanhos de quantização (número de bits) e corte da poda. Os resultados mostraram uma redução significativa dos pesos e operações da DNN para vários tamanhos de quantização e poda mantendo a acurácia na classificação das modulações.

## Referências


[1] L. Huang, W. Pan, Y. Zhang, L. Qian, N. Gao, and Y. Wu, "Data augmentation for deep learning-based radio modulation classification," *IEEE access*, vol. 8, pp. 1498–1506, 2019.

[2] Y. Wang, J. Gui, Y. Yin, J. Wang, J. Sun, G. Gui, H. Gacanin, H. Sari, and F. Adachi, "Automatic modulation classification for mimo systems via deep learning and zero-forcing equalization," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 5, pp. 5688–5692, 2020.

[3] Y. Wang, J. Wang, W. Zhang, J. Yang, and G. Gui, "Deep learning-based cooperative automatic modulation classification method for mimo systems," *Ieee transactions on vehicular technology*, vol. 69, no. 4, pp. 4575–4579, 2020.

[4] Y. Sun and E. A. Ball, "Automatic modulation classification using techniques from image classification," *IET Communications*, 2022.

[5] Y. Tu, Y. Lin, C. Hou, and S. Mao, "Complex-valued networks for automatic modulation classification," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 9, pp. 10 085–10 089, 2020.

[6] T. J. O'Shea, J. Corgan, and T. C. Clancy, "Convolutional radio modulation recognition networks," in *International conference on engineering applications of neural networks*. Springer, 2016, pp. 213–226.

[7] D. Blalock, J. J. G. Ortiz, J. Frankle, and J. Guttag, "What is the state of neural network pruning?" *arXiv preprint arXiv:2003.03033*, 2020.

[8] Q. Huang, K. Zhou, S. You, and U. Neumann, "Learning to prune filters in convolutional neural networks," in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2018, pp. 709–718.

[9] K. Wang, Z. Liu, Y. Lin, J. Lin, and S. Han, "Haq: Hardware-aware automated quantization with mixed precision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2019, pp. 8612–8620.

[10] A. S. Rakin, J. Yi, B. Gong, and D. Fan, "Defend deep neural networks against adversarial examples via fixed and dynamic quantized activation functions," *arXiv preprint arXiv:1807.06714*, 2018.

[11] S. Jung, C. Son, S. Lee, J. Son, J.-J. Han, Y. Kwak, S. J. Hwang, and C. Choi, "Learning to quantize deep networks by optimizing quantization intervals with task loss," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4350–4359.

[12] F. Tung and G. Mori, "Deep neural network compression by in-parallel pruning-quantization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 3, pp. 568–579, 2020.

[13] W. Zhe, J. Lin, V. Chandrasekhar, and B. Girod, "Optimizing the bit allocation for compression of weights and activations of deep neural networks," in *2019 IEEE International Conference on Image Processing (ICIP)*, 2019, pp. 3826–3830.

[14] H. Kung, B. McDanel, and S. Q. Zhang, "Term revealing: Furthering quantization at run time on quantized dnns," *arXiv preprint arXiv:2007.06389*, 2020.

[15] M. A. C. Fernandes and H. T. Kung, "A novel training strategy for deep learning model compression applied to viral classifications," in *2021 International Joint Conference on Neural Networks (IJCNN)*, 2021, pp. 1–9.

[16] M. Imani, M. Samragh Razlighi, Y. Kim, S. Gupta, F. Koushanfar, and T. Rosing, "Deep learning acceleration with neuron-to-memory transformation," in *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2020, pp. 1–14.

[17] K. Guo, S. Zeng, J. Yu, Y. Wang, and H. Yang, "A survey of fpga-based neural network accelerator," *arXiv preprint arXiv:1712.08934*, 2017.

[18] M. G. F. Coutinho, M. F. Torquato, and M. A. C. Fernandes, "Deep neural network hardware implementation based on stacked sparse autoencoder," *IEEE Access*, vol. 7, pp. 40 674–40 694, 2019.

[19] Y. Kim, Q. Tong, K. Choi, E. Lee, S. Jang, and B. Choi, "System level power reduction for yolo2 sub-modules for object detection of future autonomous vehicles," in *2018 International SoC Design Conference (ISOCC)*, 2018, pp. 151–155.

[20] 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.

[21] S. Jin, S. Di, X. Liang, J. Tian, D. Tao, and F. Cappello, "Deepsz: A novel framework to compress deep neural networks by using error-bounded lossy compression," in *Proceedings of the 28th International Symposium on High-Performance Parallel and Distributed Computing*, 2019, pp. 159–170.

[22] T. J. O'Shea, T. Roy, and T. C. Clancy, "Over-the-air deep learning based radio signal classification," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 168–179, 2018.

[23] DEEPSIG. (2022) Rf datasets for machine learning. [Online]. Available: https://www.deepsig.ai/datasets