Estratégia de Treinamento para Poda Consciente de Modelo de CNN Aplicado a Classificação de Imagens

Mateus A. S. de S. Goldbarg[1], Sérgio N. Silva[1], Lucileide M. D. da Silva[1,2], Marcelo A. C. Fernandes[1,3]

[1]InovaAI Lab, nPITI/IMD, Universidade Federal do Rio Grande do Norte (UFRN)
59.078-900 – Natal – RN – Brasil

[2]Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte (IFRN)
Natal – RN – Brasil

[3]Departamento de Engenharia da Computação e Automação (DCA)
Universidade Federal do Rio Grande do Norte (UFRN)
59.078-900 – Natal – RN – Brasil

mateus.goldbarg.081@ufrn.edu.br, sergionatan@dca.ufrn.br

lucileide.dantas@ifrn.edu.br, mfernandes@dca.ufrn.br

Abstract. *This article presents a new training strategy for compressing Convolutional Neural Network (CNN) models. The proposed strategy utilizes a conscious weight pruning scheme for the CNN, distinguishing it from conventional approaches. In this work, conscious pruning is applied continuously throughout the entire training process, in all mini-batches. The strategy was applied to a classification problem involving 10,000 images belonging to 10 different classes, using the CIFAR-10 dataset. The obtained results demonstrated that it was possible to remove approximately 82% of the CNN's parameters while maintaining high accuracy. These results highlight the effectiveness of the conscious weight pruning technique for this specific application.*

Resumo. *Este artigo apresenta uma nova estratégia de treinamento para compressão de modelos de redes neurais convolucionais (Convolutional Neural Networks - CNN). A estratégia proposta utiliza um esquema de poda consciente dos pesos da CNN, diferenciando-se das abordagens convencionais. Neste trabalho, a poda consciente é aplicada de forma contínua durante todo o processo de treinamento, em todos os mini-batches. A estratégia foi aplicada em um problema de classificação de 10 mil imagens, pertencentes a 10 classes diferentes, utilizando o dataset CIFAR-10. Os resultados obtidos demonstraram que foi possível remover aproximadamente 82% dos parâmetros da CNN, mantendo uma alta acurácia. Esses resultados evidenciam a eficácia da técnica de remoção de pesos por poda consciente para essa aplicação específica.*

1. Introdução

Otimizar o desempenho computacional de Redes Neurais Convolucionais (*Convolutional Neural Networks* - CNN) tem sido um foco importante no campo de aprendizado de máquina, e a pesquisa nessa área tem crescido exponencialmente. Existem duas principais categorias de estratégias utilizadas para acelerar a inferência de CNNs: aquelas baseadas em algoritmos e aquelas baseadas em sistemas.

As abordagens baseadas em algoritmos são voltadas para melhorar a eficiência do processamento dos modelos. Elas incluem técnicas como processamento paralelo, que exploram o paralelismo de dados e modelos para acelerar a computação. Além disso, a compressão de modelos é uma estratégia comum, que envolve a redução de peso e a quantização de valores de peso e ativação. A exploração da esparsidade, tanto em nível de valor quanto em nível de bit, é outra abordagem para acelerar a inferência. Além disso, a redução do modelo por meio de aproximação é uma técnica utilizada para simplificar o modelo, sacrificando um pouco da precisão [Blalock et al. 2020, Huang et al. 2018, Wang et al. 2019, Rakin et al. 2018, Jung et al. 2019, Tung and Mori 2020, Zhe et al. 2019, Kung et al. 2020, Fernandes and Kung 2021].

Por outro lado, as abordagens baseadas em sistema concentram-se em otimizar a infraestrutura de hardware e software para melhorar o desempenho geral. Isso inclui a utilização de sistemas de memória de alta largura de banda para minimizar o tempo de acesso aos dados, bem como aceleradores de hardware, como ASICs e FPGAs, projetados especificamente para acelerar o processamento de CNNs. Além disso, sistemas de computação distribuída têm sido explorados para distribuir a carga de trabalho e acelerar a inferência em escala [Imani et al. 2020, Guo et al. 2017, Coutinho et al. 2019]. Otimizações de compilador, como a eliminação de expressões comuns, também podem ser aplicadas para melhorar o desempenho [Kim et al. 2018].

Neste trabalho, o foco foi a estratégia baseada em algoritmos de compressão de modelos, especificamente a compressão dos pesos, como uma forma comum de reduzir o número de operações em uma Rede Neural Convolucional (CNN), mantendo um impacto mínimo na precisão da classificação. Essa abordagem, também conhecida como compressão consciente (*aware compression*), tem o potencial de melhorar o desempenho da CNN em situações que exigem classificação automática de modulação em tempo real.

A compressão consciente geralmente envolve técnicas como poda (pruning) [Blalock et al. 2020, Huang et al. 2018], que remove pesos menos relevantes do modelo, e quantização dos pesos [Wang et al. 2019, Rakin et al. 2018], que reduz a precisão dos valores dos pesos. Em alguns casos, a poda é realizada em etapas durante o treinamento em conjunto com a quantização [Han et al. 2015, Jin et al. 2019, Fernandes and Kung 2021].

Diferentemente da abordagem convencional, este trabalho propõe uma estratégia em que a poda consciente é realizada de forma contínua durante todo o processo de treinamento, ou seja, em todos os mini-batches. A Figura 2 detalha a estratégia proposta neste artigo, na qual não há chave seletora e a poda é executada em todos os mini-batches de cada época de aprendizagem.

No entanto, neste artigo, propomos um novo esquema de compressão de modelo que aborda a poda consciente a cada iteração do treinamento, ou seja, a cada *mini-batch*. Essa abordagem difere da metodologia convencional de poda consciente, na qual o processo de poda, ou remoção dos pesos, é sempre realizado no último *mini-batch* de cada época.

2. Proposta

2.1. Exploração de esparsidade

Uma matriz é dita esparsa quando a maior parte de seus elementos são zeros. A exploração da esparsidade em modelos de CNNs pode estar presente em multiplos níveis: bit, valor, canal, filtro, bloco entre outros. O valor da esparsidade, s_k, da k-ésima camada associado a um modelo da CNN pode ser expresso por

$$s_k = \frac{N_k}{G_k} \times 100 \tag{1}$$

onde N_k é o número de pesos da k-ésima camada cujo seu módulo é menor que um dado limiar chamado neste trabalho de β_k. Já a variável G_k é o número de pesos total da k-ésima camada. A esparsidade também pode ser dada por modelo, no qual pode ser expresso como

$$s = \frac{N^t}{G^t} \times 100 \tag{2}$$

onde N^t é o número total de pesos do modelo cujo o módulo é menor que β_k, que é expresso por

$$N^t = \sum_{k=1}^{P} G_k \tag{3}$$

sendo P a quantidade de camadas do modelo. A variável G^t representa o número total de pesos do modelo, que é expresso como

$$G^t = \sum_{k=1}^{P} N_k \tag{4}$$

.

A Tabela 1 apresenta um exemplo da esparsidade d obtida em uma rede neural artificial com $G^t = 65536$ pesos, para vários valores de β_k. É importante observar que o mesmo valor de β_k foi aplicado em todas as camadas.

Tabela 1. Exemplo de esparsidade para o modelo com diferêntes valores de β_k.

β_k	N^t	d
0,01	6438	9,82%
0,001	3220	4,91%
0,0001	1924	2,94%

2.2. Compressão de modelo por poda

A compressão por poda parte da hipótese de que as matrizes ou tensores de pesos possuem alta esparsidade, o que pode ser explorado para reduzir o tamanho do modelo da CNN e aumentar a velocidade de processamento.

A otimização utilizando a técnica de poda consiste na remoção de pesos insignificantes do modelo, a partir do limiar β. A definição de β expressa o quão agressiva será a poda no modelo, sendo que a remoção de muitos pesos pode afetar significativamente a

acurácia da rede [Sawant et al. 2022]. A remoção desses pesos faz com que a quantidade de operações realizadas durante a inferência da rede seja menor do que no modelo não comprimido.

Existem várias formas de calcular o limiar de poda, β. Em muitos trabalhos na literatura, parte-se da hipótese de que os pesos de uma dada k-ésima camada seguem uma distribuição Gaussiana com média zero. Assim, com base nessa hipótese, pode-se definir o limiar para a k-ésima camada como

$$\beta_k = \alpha \times \sigma_k \tag{5}$$

onde α define a agressividade da poda e σ_k é o desvio padrão dos pesos da k-ésima camada do modelo.

A remoção dos pesos do modelo pode ser feita de forma consciente, onde os pesos são removidos sistematicamente durante o processo de aprendizagem, chamado de "aware pruning"[Awan et al. 2022], ou também pode ser feita após o treinamento, chamado de "post-training pruning"(PTP), onde os pesos são removidos somente após o modelo ter sido treinado. Trabalhos na literatura mostram que o processo de poda consciente possui resultados melhores do que o PTP, dado que a poda consciente força o algoritmo de treinamento a encontrar uma nova solução para o valor dos pesos, diferente do PTP [He et al. 2022].

A Figura 1 ilustra o processo mais convencional de poda consciente de um modelo CNN no qual o processo de poda, ou remoção dos pesos, é sempre realizado no último *mini-batch* de cada época. Como pode ser observado na Figura 1, uma uma chave de seletora é utilizada para controle, realizando o a poda apenas quando acionada.

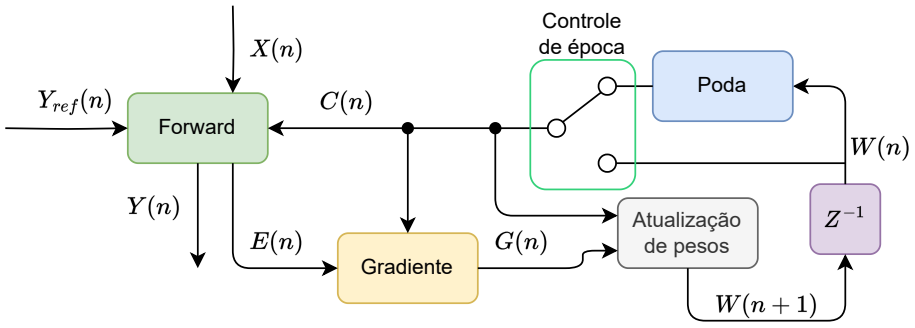


Figura 1. Representação da estratégia convencional do funcionamento da poda consciente.

No esquema, ilustrado na Figura 1, as diferentes etapas do processo são descritas: a iteração n, o atraso unitário z^{-1}, os pesos da CNN $\mathbf{W}(n)$, a entrada $\mathbf{X}(n)$, a saída desejada $\mathbf{Y}ref(n)$, a saída da CNN $\mathbf{Y}(n)$, a métrica de erro $\mathbf{E}(n)$ entre $\mathbf{Y}(n)$ e $\mathbf{Y}ref(n)$, e a métrica de gradiente $\mathbf{G}(n)$ de todas as camadas. $\mathbf{C}(n)$ representa os pesos da CNN após a poda na n-ésima iteração.

Diferentemente da abordagem convencional, este trabalho propõe uma estratégia em que a poda consciente é realizada de forma contínua durante todo o processo de treinamento, ou seja, em todos os mini-batches. A Figura 2 detalha a estratégia proposta neste

artigo, na qual não há chave seletora e a poda é executada em todos os mini-batches de cada época de aprendizagem.

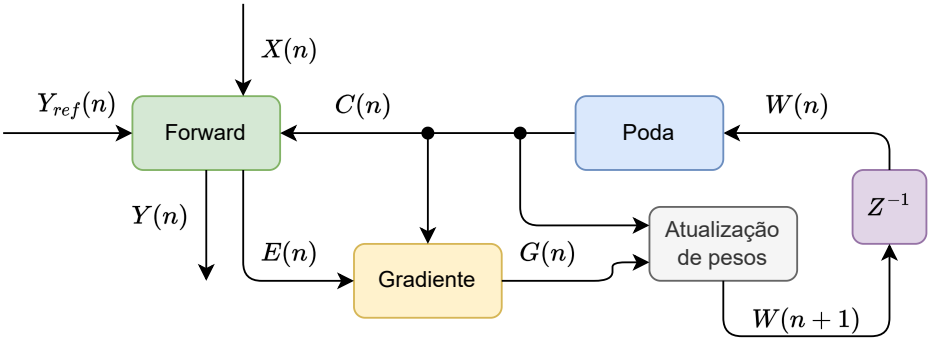
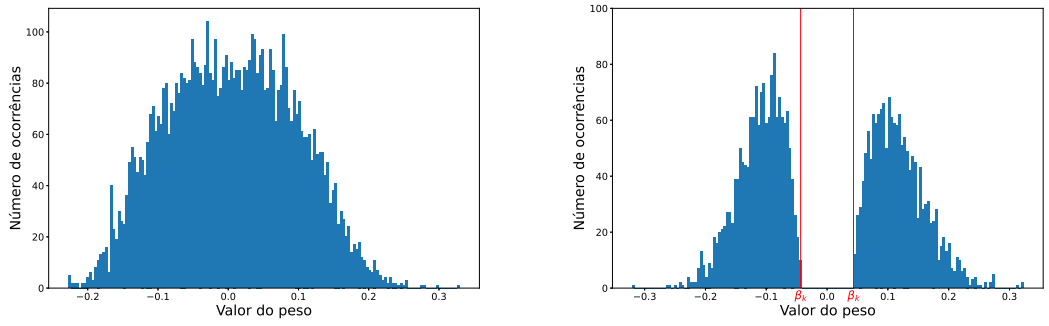


Figura 2. Esquema da estratégia proposta para poda consciente. O processo de poda é realizado em todos os mini-batches de cada época de aprendizagem.

Em uma k-ésima camada, na n-ésima iteração, a função que calcula a poda pode ser expressa como

$$\mathbf{C}_k(n) = P\left(\mathbf{W}_k(n), \beta_k\right) = \begin{cases} \mathbf{W}_k(n) & \text{se } |\mathbf{W}_k(n)| \geq \beta_k \\ 0 & \text{se } |\mathbf{W}_k(n)| < \beta_k \end{cases} \tag{6}$$

onde $P(\cdot,\cdot)$ é a função que avalia os pesos a serem removidos, $\mathbf{W}_k(n)$ são os pesos da k-ésima camada na n-ésima iteração do treinamento, e β_k é o limiar da k-ésima camada. A Figura 3 mostra o histograma dos pesos de uma camada associada a uma CNN antes da compressão e o resultado da aplicação da remoção de pesos por poda consciente com $\alpha = 0{,}5$.



(a) Pesos antes da compressão.

(b) Poda consciente com $\alpha = 0.5$.

Figura 3. Histograma dos valores dos pesos associado a um exemplo de compressão de pesos por poda consciente com $\alpha = 0.5$ ($\beta_k = 0.5 \times \sigma_k$) aplicada a uma camada de uma rede CNN.

3. Metodologia

3.1. Modelo e hiper-parâmetros

Com o objetivo de validar a estratégia proposta, foram realizados sete experimentos de treinamento, variando o valor de α para 0, 0.25, 0.5, 0.75, 1, 1.25 e 1.5. A função

utilizada para o cálculo das perdas foi a *Sparse Categorical Crossentropy*, e o otimizador utilizado foi o *Adam*. Em todos os experimentos, a estrutura da rede foi criada da seguinte forma:

1. Camada de entrada de $32 \times 32 \times 3$;
2. Camada convolucional com 32 filtros de tamanho 3×3, sem padding, stride igual a 1 e função de ativação ReLU;
3. Camada convolucional com 64 filtros de tamanho 3×3, sem padding, stride igual a 1 e função de ativação ReLU;
4. Camada de Max-pooling de tamanho 2×2, stride igual a 2;
5. Camada convolucional com 128 filtros de tamanho 3×3, sem padding, stride igual a 1 e função de ativação ReLU;
6. Camada convolucional com 128 filtros de tamanho 3×3, sem padding, stride igual a 1 e função de ativação ReLU;
7. Camada de Max-pooling de tamanho 2×2, stride igual a 2;
8. Camada totalmente conectada com 256 neurônios e função de ativação ReLU;
9. Camada totalmente conectada com 128 neurônios e função de ativação ReLU;
10. Camada de saída com 10 neurônios utilizando função de ativação softmax.

A Tabela 2 mostra a quantidade de parâmetros em cada camada, bem como o total de parâmetros do modelo.

Tabela 2. Quantidade de parâmetros da CNN utilizada para validar a proposta.

Tipo de Camada	Quantidade de Parâmetros
Convolucional 2D	896
Convolucional 2D	$18,496$
Max Pooling 2D	0
Convolucional 2D	$73,856$
Convolucional 2D	$147,584$
Max Pooling 2D	0
Flatten	0
Dense	$819,456$
Dense	$32,896$
Dense	$1,290$
Total de 1,094,474 parâmetros	

3.2. Dataset

O dataset escolhido para este trabalho foi o CIFAR-10 [Krizhevsky and Hinton 2009], que consiste em 60 mil imagens coloridas com 3 canais (RGB) de tamanho 32×32 pixels, divididas em 10 classes, com 6 mil imagens de cada classe. O dataset é dividido em 50 mil imagens para treinamento e 10 mil imagens para teste. As 10 classes são: avião, automóvel, pássaro, gato, veado, cachorro, sapo, cavalo, barco e caminhão. As classes são mutuamente exclusivas, ou seja, não há sobreposição entre automóveis e caminhões. A classe de automóveis inclui apenas sedans, SUVs e carros de pequeno porte. A classe de caminhões inclui apenas caminhões grandes, excluindo pickups. A Figura 4 apresenta algumas amostras do dataset CIFAR-10.

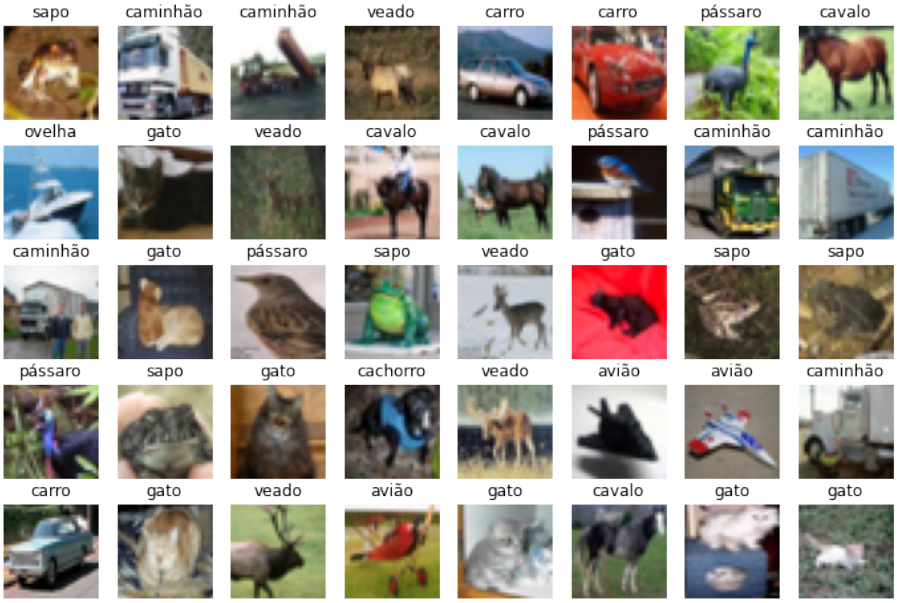


Figura 4. Exemplos de amostras do dataset CIFAR-10, utilizadas neste trabalho.

4. Resultados

Para analisar o comportamento dos pesos ao final do treinamento, foi montado um histograma contendo todos os pesos da segunda camada convolucional do modelo, conforme apresentado na Figura 5. É possível perceber que uma grande quantidade de pesos se concentra em valores próximos a zero, seguindo uma distribuição Gaussiana.

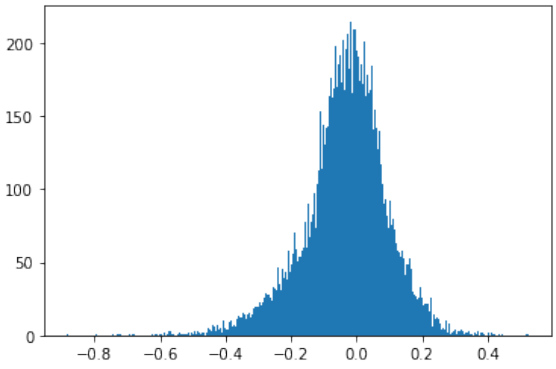


Figura 5. Histograma dos pesos da segunda camada convolucional do modelo sem poda ($\alpha = 0$).

No processo de validação, sem o uso de poda, obteve-se uma acurácia de $75,30\%$. Para visualização dos resultados, a Figura 6 mostra a matriz de confusão obtida.

A Figura 7 apresenta a curva de treinamento para os vários valores de α mencionados na metodologia. O treinamento foi realizado ao longo de 30 épocas. Já a Figura 8 mostra os resultados de acurácia da validação durante o treinamento para os diferentes valores de α. A variação da esparsidade ao longo das épocas para todos os valores de α pode ser visualizada na Figura 9.

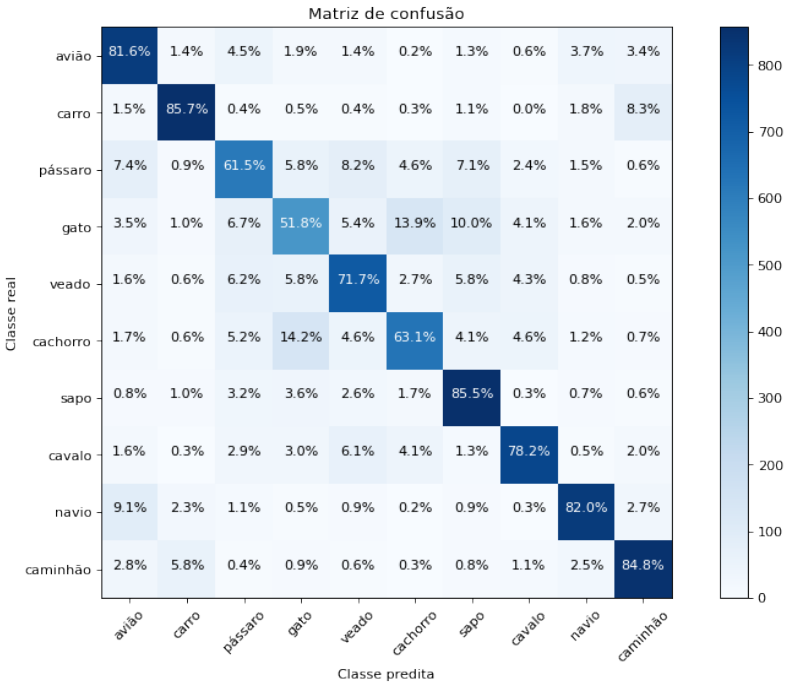


Figura 6. Matriz de confusão para a validação do modelo sem uso de poda ($\alpha = 0$).

As Figuras 7, 8 e 9 mostram os resultados esperados para a proposta, ou seja, quanto mais agressiva a poda (valores maiores de α), menor será a acurácia alcançada pelo modelo, porém maior será sua esparsidade. A Tabela 3 apresenta os resultados de esparsidade, acurácia de inferência e número total de parâmetros que não foram removidos para cada valor de α.

Tabela 3. Acurácia e esparsidade dos modelos

α	Esparsidade	Acurácia	Número de pesos
0,00	0,000 %	75,300%	$\approx$ 1,1M
0,25	52,728%	73,250%	$\approx$ 517k
0,50	68,251%	71,900%	$\approx$ 347k
0,75	82,301%	71,840%	$\approx$ 194k
1,00	92,465%	67,130%	$\approx$ 82k
1,25	96,006%	45,550%	$\approx$ 44k
1,50	97,827%	44,130%	$\approx$ 24k

Os dados da Tabela 3 são ilustrados na curva apresentada na Figura 10, que mostra o comportamento da acurácia em função da esparsidade obtida a partir dos experimentos com os diferentes valores de α. É possível observar que, a partir de uma certa esparsidade do modelo, a acurácia cai rapidamente devido ao limiar abranger os pesos mais significativos da rede. Para o modelo em questão, a acurácia se manteve acima de 70% até atingir uma esparsidade de 82,30%. A partir desse valor, a acurácia cai rapidamente, mostrando que a aplicação da técnica de poda de forma muito agressiva gera modelos com baixa acurácia.

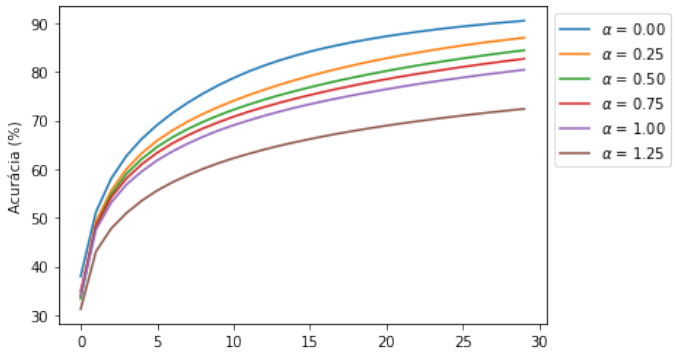


Figura 7. Acurácia associada ao treinamento para os diferentes valores de α.

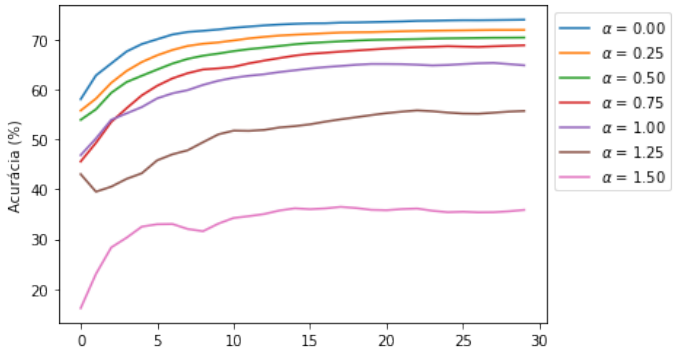


Figura 8. Acurácia associada à validação durante o processo de treinamento para os diferentes valores de α.

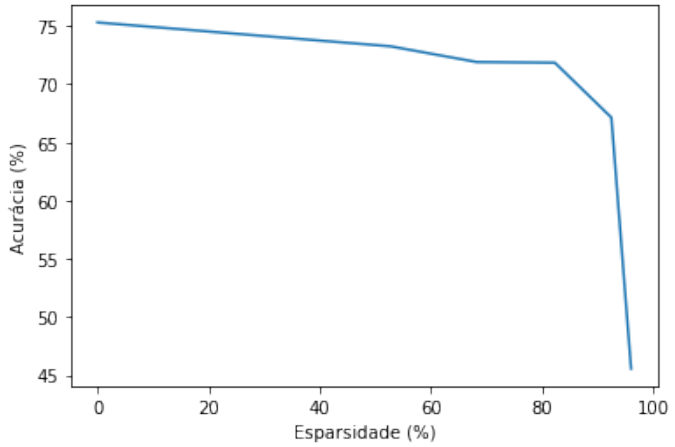


Figura 10. Curva de acurácia em função da esparsidade obtida a partir dos experimentos com os diferentes valores de α.

Na validação do experimento para $\alpha = 0,25$, obteve-se uma acurácia de 73,25%, o que significa que, em comparação com o modelo não comprimido, é possível remover 52,72% dos pesos e obter uma acurácia apenas 2,05% menor. É possível analisar o resultado a partir da matriz de confusão na Figura 11. Já para o caso com $\alpha = 0,50$, o resultado foi uma acurácia de 71,90%, mostrando que é possível remover aproximadamente 68,25% dos pesos do modelo e obter uma acurácia apenas 3,4% menor que a do modelo

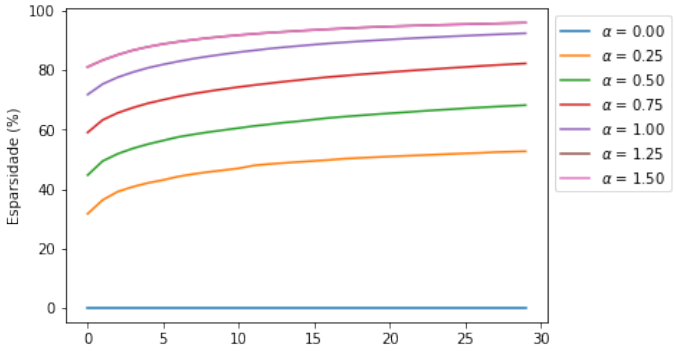


Figura 9. Evolução da esparsidade ao longo das épocas para todos os valores de α.

não comprimido. A matriz de confusão pode ser visualizada na Figura 12.

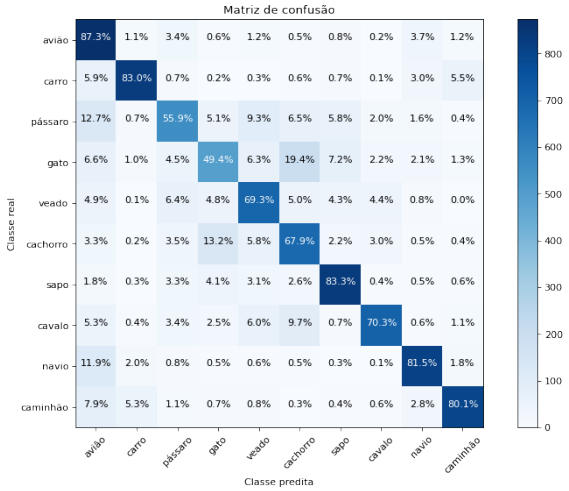


Figura 11. Matriz de confusão para $\alpha = 0,25$.

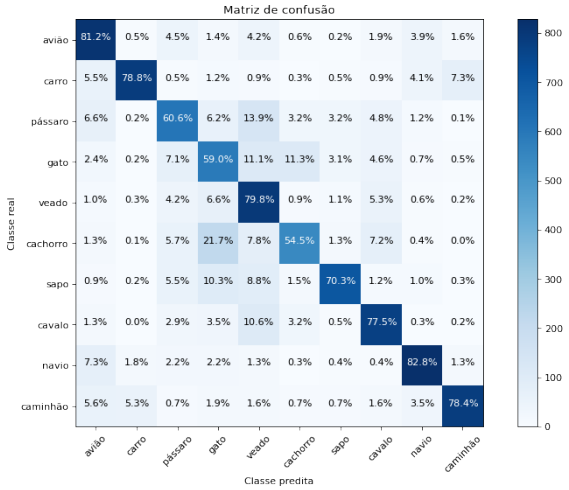


Figura 12. Matriz de confusão para $\alpha = 0,50$.

No caso de $\alpha = 0,75$, a acurácia foi de 71,84%, tornando possível a remoção de 82,30% dos pesos do modelo, com uma acurácia de validação apenas 3,46% menor que a da CNN não comprimida. A matriz de confusão pode ser visualizada na Figura 13. No experimento associado a $\alpha = 1,00$, a acurácia obtida foi de 67,13%, permitindo assim obter uma acurácia 7,17% menor que a do modelo não comprimido ao remover 92,44% de seus parâmetros. A Figura 14 apresenta a matriz de confusão do experimento com $\alpha = 1,00$.

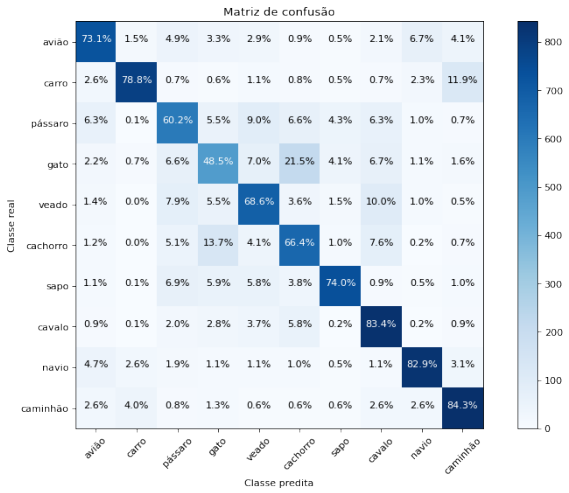


Figura 13. Matriz de confusão para $\alpha = 0,75$.

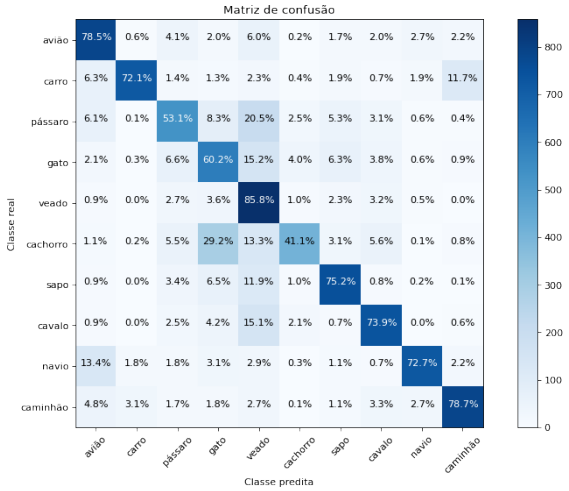


Figura 14. Matriz de confusão para $\alpha = 1,00$.

Nos experimentos com $\alpha = 1,25$ e $\alpha = 1,50$, os valores de acurácia foram bastante impactados, com a acurácia alcançando 45,55% e 44,13%, respectivamente. A esparsidade foi significativamente elevada, atingindo valores acima de 95%, porém essa alta esparsidade acabou comprometendo pesos importantes na classificação. As Figuras 15 e 16 apresentam as matrizes de confusão para os experimentos com $\alpha = 1,25$ e $\alpha = 1,50$, respectivamente.

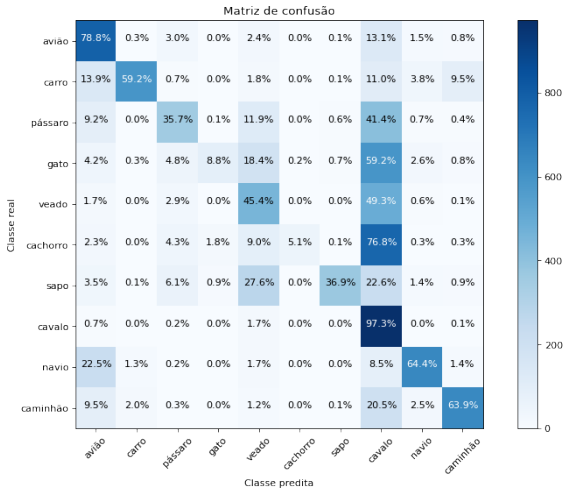


Figura 15. Matriz de confusão para $\alpha = 1,25$.

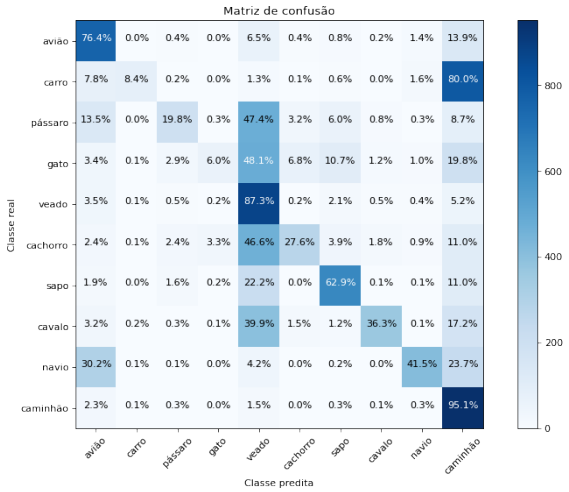


Figura 16. Matriz de confusão para $\alpha = 1,50$.

5. Conclusões

Conclui-se portanto que a utilização da técnica de compressão por poda a cada batch é capaz de gerar um modelo menor (com menos parâmetros) que a rede neural original, mantendo uma acurácia alta. A variação do valor do limiar afeta diretamente a esparsidade do modelo, ou seja, quantos parâmetros são removidos. Valores de limiar muito altos podem fazer com que a rede tenha uma acurácia baixa. Levando em conta que a diminuição de parâmetros do modelo implicam em menos operações matemáticas, trabalhos futuros podem ser realizados utilizando a rede neural comprimida em microcontroladores de baixo consumo energético analisando a energia necessária para a inferência comparando as redes neurais comprimidas e não comprimidas.

Referências

Awan, Z. W., Khalid, S., and Gul, S. (2022). A theoretical cnn compression framework for resource-restricted environments. In *2022 2nd International Conference on Digital Futures and Transformative Technologies (ICoDT2)*, pages 1–8. IEEE.

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

Coutinho, M. G. F., Torquato, M. F., and Fernandes, M. A. C. (2019). Deep neural network hardware implementation based on stacked sparse autoencoder. *IEEE Access*, 7:40674–40694.

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

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

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

He, H., Huang, L., Huang, Z., and Yang, T. (2022). The compression techniques applied on deep learning model. *Highlights in Science, Engineering and Technology*, 4:325–331.

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

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

Jin, S., Di, S., Liang, X., Tian, J., Tao, D., and Cappello, F. (2019). 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*, pages 159–170.

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

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

Krizhevsky, A. and Hinton, G. (2009). Cifar-10. Technical Report.

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

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

Sawant, S. S., Wiedmann, M., Göb, S., Holzer, N., Lang, E. W., and Götz, T. (2022). Compression of deep convolutional neural network using additional importance-weight-based filter pruning approach. *Applied Sciences*, 12(21):11184.

Tung, F. and Mori, G. (2020). Deep neural network compression by in-parallel pruning-quantization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(3):568–579.

Wang, K., Liu, Z., Lin, Y., Lin, J., and Han, S. (2019). Haq: Hardware-aware automated quantization with mixed precision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8612–8620.

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