

Compressão Consciente de Modelos de Redes Neurais Profundas Baseada em Poda Seguida de Quantização

Mateus Arnaud Santos de Sousa Goldberg¹

Prof. Dr. Marcelo Augusto Costa Fernandes²

Prof. Dr. Sérgio Natan Silva³

¹mateus.goldbarg@dca.ufrn.br

^{1,2,3}Programa de Pós-graduação em Engenharia Elétrica e de Computação
(PPGgEEC)

Universidade Federal do Rio Grande do Norte

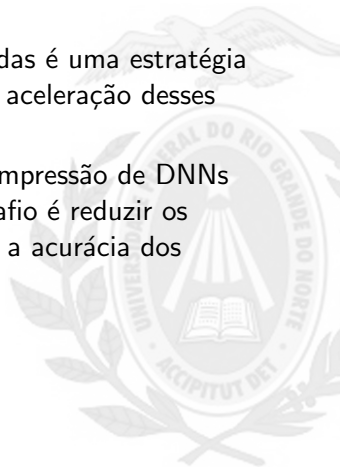
20 de Fevereiro de 2024

Contextualização e Motivação

- A Inteligência Artificial (IA), já está presente em uma gama de atividades como publicidade, finanças, jogos eletrônicos, visão computacional e diagnósticos médicos;
- Técnicas de aprendizado profundo (deep learning) têm sido usados com sucesso na solução de muitos problemas;
- A grande quantidade de operações numéricas realizadas em algoritmos de aprendizado profundo podem ser um gargalo quando se é necessário o processamento de um conjunto de dados em um pequeno intervalo de tempo ou quando os recursos de processamento são limitados;
- Devido a complexidade das redes neurais profundas, é necessário um elevado espaço em memória para armazená-las.

Contextualização e Motivação

- A compressão das Redes Neurais Profundas é uma estratégia viável para a redução da complexidade e aceleração desses algoritmos;
- As abordagens mais convencionais de compressão de DNNs são as de poda e de quantização. O desafio é reduzir os modelos de forma a afetar minimamente a acurácia dos modelos.



Objetivos

- Desenvolver técnicas de compressão consciente de DNNs utilizando as estratégias de poda, quantização e poda seguida de quantização.
- Validar a compressão do modelo de DNNs através das métricas de esparsidade e do tamanho do modelo comprimido em relação ao não comprimido.
- Validar a estratégia de compressão consciente de modelos de DNNs aplicando-os à ambientes de microserviços e avaliar seu consumo de memória, processamento, tempo de inferência e sua escalabilidade.

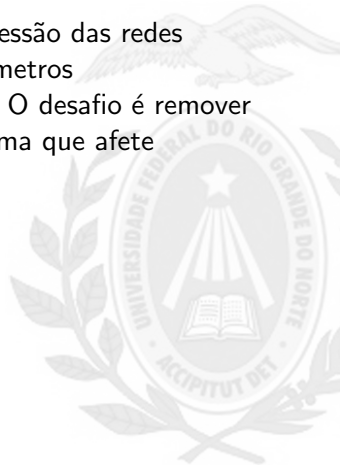
Estratégias de Compressão

- Compressão pós treino: A compressão é realizada apenas após o treinamento do modelo.
 - Poda pós treino (Post-training pruning - PTP);
 - Quantização pós treino (Post-Training Quantization - PTQ).
- Compressão consciente: A compressão é realizada durante o loop de treinamento.
 - Poda consciente (Aware Prune);
 - Quantização consciente (Aware Quantization);
 - Poda seguida de quantização (Prune Followed by Quantization).

Os algoritmos mais convencionais de compressão consciente realizam a compressão apenas uma vez por época, enquanto o algoritmo proposto realiza a compressão a cada mini-batch de cada época do treinamento.

Compressão consciente por poda

Uma das abordagens convencionais de compressão das redes neurais é a poda (pruning), que remove parâmetros sistematicamente de um modelo já existente. O desafio é remover uma grande quantidade de parâmetros de forma que afete minimamente a acurácia do modelo.



Compressão consciente por poda

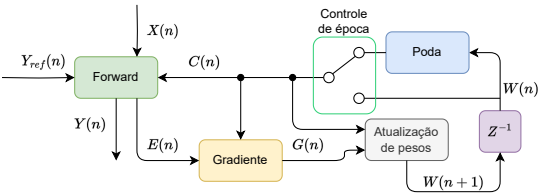


Figura 1: Diagrama do loop de aprendizado utilizando poda com controle de época

As estratégias mais convencionais de poda realizam a remoção apenas uma vez por época. Geralmente no último mini-batch de cada época.

Compressão consciente por poda

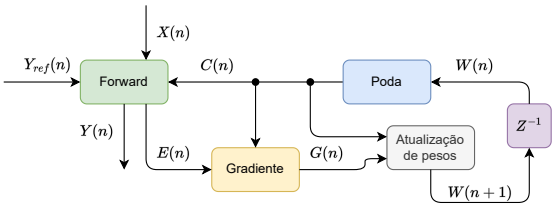


Figura 2: Diagrama do loop de aprendizado utilizando poda consciente a cada mini-batch

Compressão consciente por poda

A estratégia utilizada para escolher quais pesos devem ser removidos é dada por

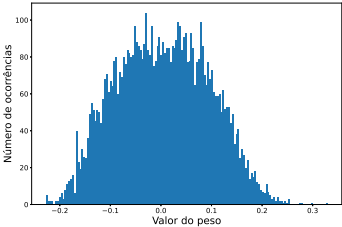
$$C_k(n) = P(W_k(n), \beta_k) = \begin{cases} w_k(n) & \text{if } |w_k(n)| \geq \beta_k \\ 0 & \text{if } |w_k(n)| < \beta_k \end{cases}, \quad (1)$$

onde β_k é a janela de corte da k -ésima camada definida por

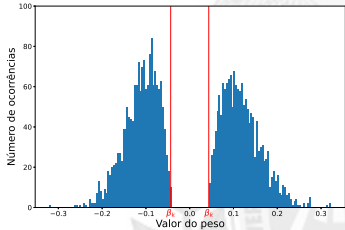
$$\beta_k = \alpha \times \sigma_k, \quad (2)$$

sendo σ_k o desvio padrão da k -ésima camada e α o valor da agressividade da poda.

Compressão consciente por poda



(a) Pesos não comprimidos



(b) Poda consciente

Figura 3: Histograma do 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 à uma camada de uma rede CNN.

Compressão consciente por Quantização

A quantização de DNNs é uma estratégia que visa reduzir o consumo de recursos computacionais nas operações matemáticas reduzindo a precisão dos parâmetros da rede diminuindo a representação, em bits, dos seus valores.



Compressão consciente por Quantização

A estratégia utilizada para quantização de pesos é definida por

$$C_k(n) = Q(W_k(n), q_k) = \left\lceil \frac{W_k(n)}{q_k} \right\rceil \times q_k, \quad (3)$$

sendo q_k definido como o fator de quantização, ou de escala, da k -ésima camada

$$q_k = \frac{\max \{|W_k(n)|\}}{2^{b-1} - 1}. \quad (4)$$

onde b é o parâmetro que define a quantidade de bits para quantização.



Compressão consciente por Quantização

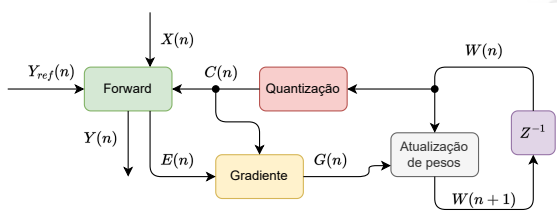
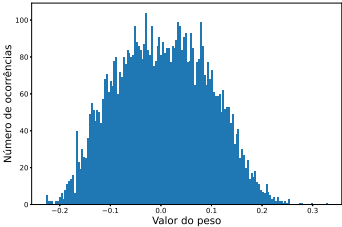
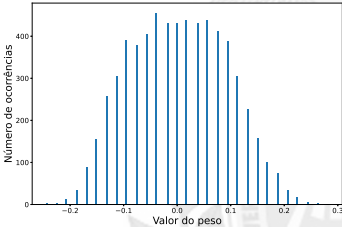


Figura 4: Diagrama do loop de aprendizado com quantização

Compressão consciente por Quantização



(a) Pesos não comprimidos

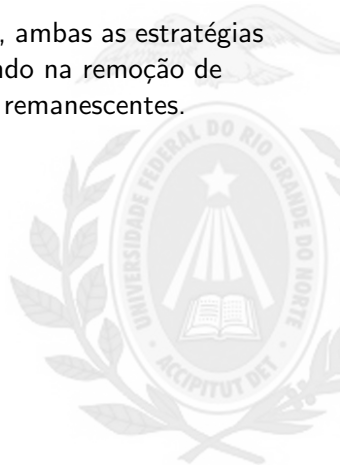


(b) Quantização consciente

Figura 5: 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.

Compressão consciente por Poda seguida de Quantização

Na compressão por poda seguida quantização, ambas as estratégias são utilizadas durante o treinamento, resultando na remoção de pesos insignificantes e quantização dos pesos remanescentes.



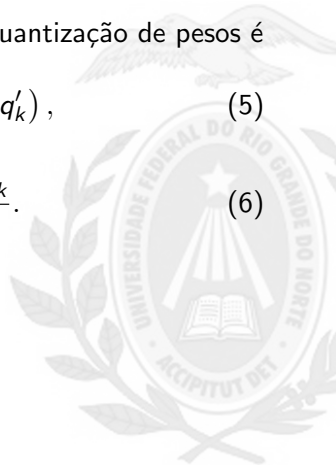
Compressão consciente por Poda seguida de Quantização

A estratégia utilizada para poda seguida de quantização de pesos é definida por

$$C_k(n) = Q(P(W_k(n), \beta_k), q'_k), \quad (5)$$

onde

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



Compressão consciente por Poda seguida de Quantização

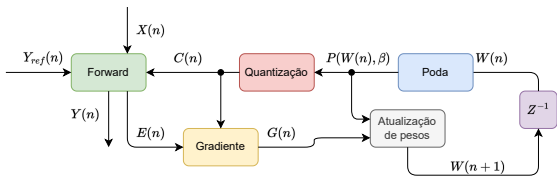
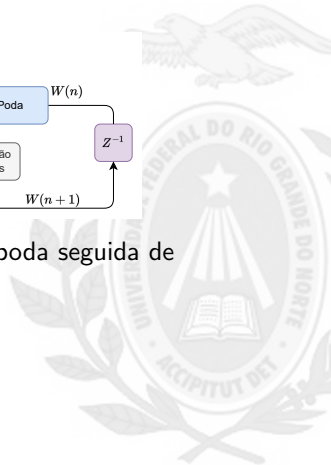
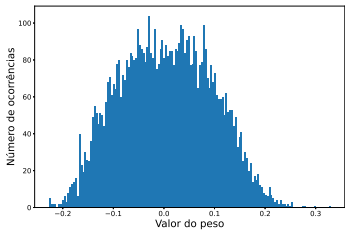


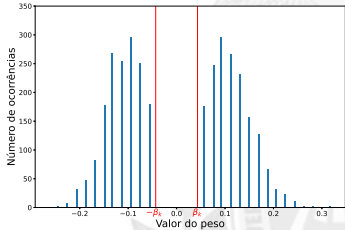
Figura 6: Diagrama do loop de aprendizado com poda seguida de quantização



Compressão consciente por Poda seguida de Quantização



(a) Pesos não comprimidos



(b) Poda seguida de Quantização

Figura 7: Histograma do valores dos pesos associado a um exemplo de compressão consciente dos pesos por poda seguida de quantização a cada iteração com $\alpha = 0,5$ e 5 bits aplicada à uma camada de uma rede CNN

Geração de modelos comprimidos

Após utilização da técnica, o arquivo do modelo gerado precisa ser comprimido. A Para lidar com os modelos que sofreram a compressão por poda, é possível escolher duas estratégias:

- Gerar modelo no formato esparsos:
 - Modelo gerado salvo no formato esparsos CSR ou CSC;
 - Necessidade de hardwares específicos para operações esparsas;
 - Modelos com baixa esparsidade podem aumentar o consumo de memória, tempo de inferência e processamento.
- Manter formato denso e comprimir a partir de Deflate:
 - O arquivo do modelo gerado é comprimido e menor que o original;
 - Combina as técnicas de Huffman e Lempel-Ziv-Storer-Szymanski para compressão do modelo;
 - A inferência é feita com o modelo no formato denso (pesos supostamente removidos).

Para a estratégia de quantização, os parâmetros do modelos são transformados para uma representação em bits, sendo calibrados a partir do fator de quantização q_k .



Classificação Automática de modulações

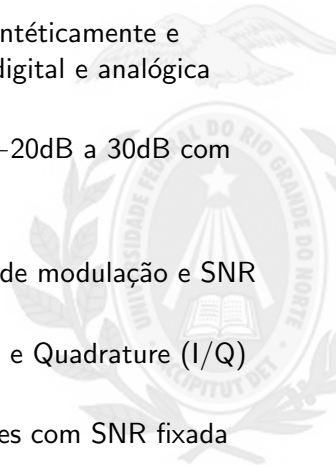
- A Modulação, nas transmissões de dados, refere-se ao processo de modificar uma ou mais características de um sinal chamado de portadora para representar informações ou dados.
- A modulação é necessária para transmitir dados por meio de um meio de comunicação, como cabos de cobre, fibras ópticas ou ondas de rádio.
- 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 facilita a detecção de falhas, monitoramento de interferência de espectro e acesso dinâmico ao espectro.

Dataset

O *dataset* inclui efeitos de canal simulados sinteticamente e gravações pelo ar de 24 tipos de modulação digital e analógica com as seguintes características:

- 26 níveis de relação sinal ruído (SNR) (-20dB a 30dB com passo de 2dB);
- 2 milhões de sinais;
- 4096 realizações (frames) para cada par de modulação e SNR (2.555.904 frames no total);
- cada frame com 1024 amostras In Phase e Quadrature (I/Q) (2×1024).

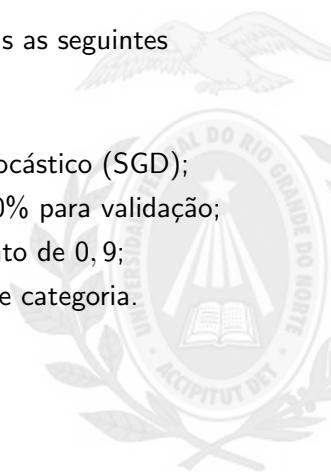
Foram utilizados apenas 7 tipos de modulações com SNR fixada em -20dB .



Treinamento da DNN

Para o treinamento do modelo foram definidas as seguintes características:

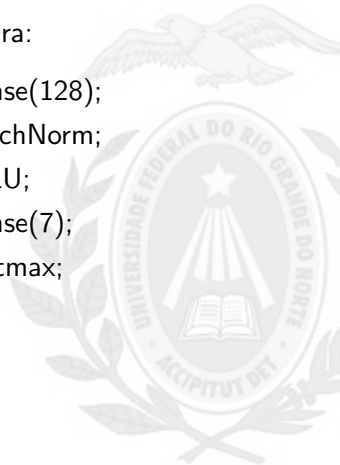
- Tamanho de cada batch de 64;
- Otimizador de Descida de Gradiente Estocástico (SGD);
- 70% das amostras para treinamento e 30% para validação;
- Taxa de aprendizagem de 0,05 e momento de 0,9;
- Critério de loss como entropia cruzada de categoria.



Arquitetura do modelo

O modelo foi criado com a seguinte arquitetura:

- Entrada (1024×2);
- Bloco repetido 6 vezes:
 - Conv1D (40×4);
 - BatchNorm;
 - ReLU;
 - MaxPool (2);
- Flatten;
- Dense(128);
- BatchNorm;
- ReLU;
- Dense(128);
- BatchNorm;
- ReLU;
- Dense(7);
- Softmax;



Resultados

Tabela 1: Acurácias obtidas a partir do modelo comprimido para vários valores de tamanho de bits e α .

| <div>bits</div> <div>α</div> | 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% |

Resultados

Tabela 2: Valores de esparsidade obtidos a partir do modelo comprimido para vários valores de tamanho de bits e α .

| $\alpha \backslash$ 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% |

Resultados

Tabela 3: Tamanho do modelo comprimido para vários valores de tamanho de bits e α .

| α \ 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 |

Resultados

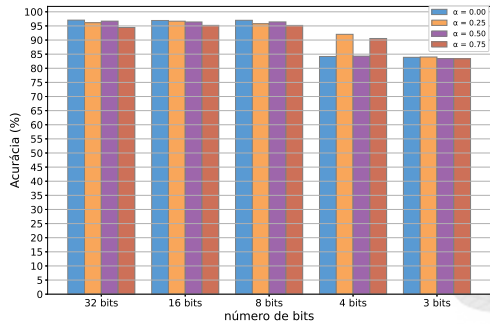


Figura 8: Acurácias obtidas para diversas configuração de α e número de bits

Resultados

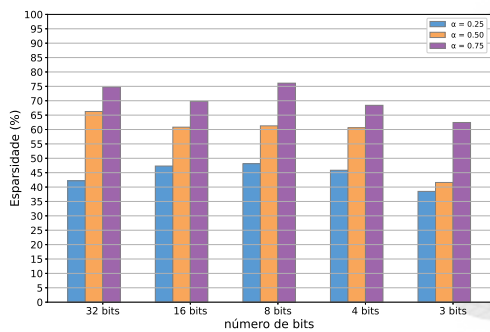


Figura 9: Esparsidades obtidas para diversas configuração de α e número de bits

Resultados

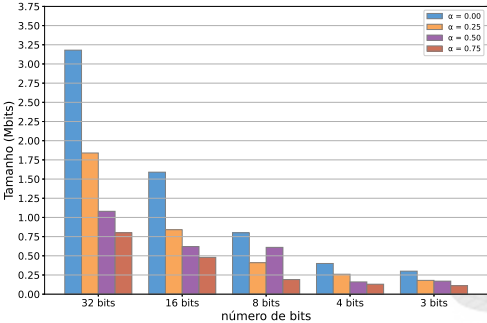
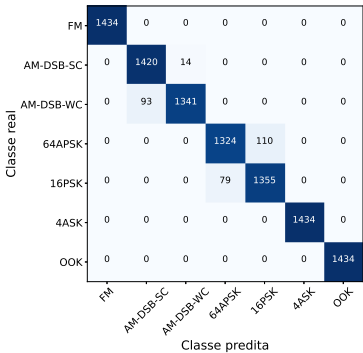
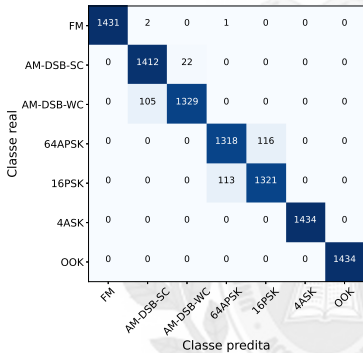


Figura 10: Tamanhos dos modelos obtidos para diversas configuração de α e número de bits

Resultados



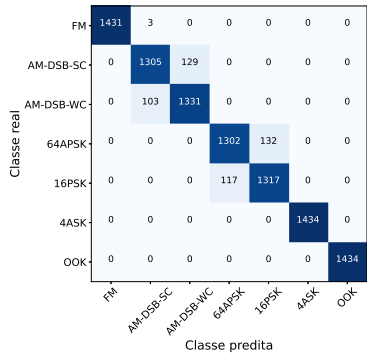
(a) Modelo não comprimido



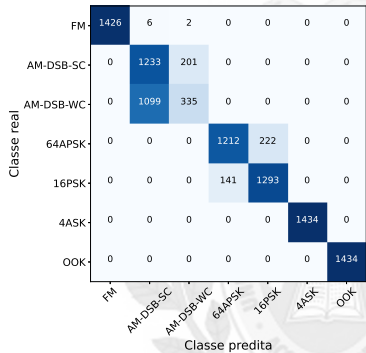
(b) 8 bits e $\alpha = 0,50$

Figura 11: Matrizes de confusão de alguns modelos obtidos apos o treinamento

Resultados



(a) 8 bits e $\alpha = 0,75$

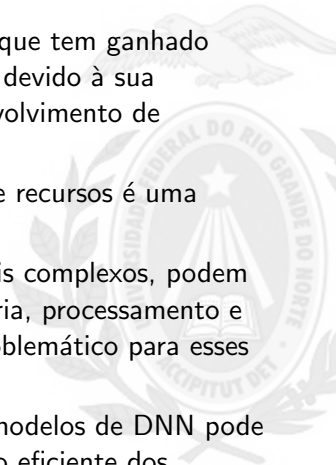


(b) 3 bits e $\alpha = 0,75$

Figura 12: Matrizes de confusão de alguns modelos obtidos apos o treinamento

Compressão de modelos para microserviços

- Microserviços são um estilo arquitetural que tem ganhado crescente popularidade nos últimos anos devido à sua abordagem ágil e escalável para o desenvolvimento de aplicações;
- Em ambientes de microserviços, o uso de recursos é uma preocupação importante;
- Modelos de DNNs, especialmente os mais complexos, podem requisitar um grande consumo de memória, processamento e tempo de inferência. O que pode ser problemático para esses ambientes.
- O uso de estratégias de compressão de modelos de DNN pode afetar positivamente o desempenho e uso eficiente dos recursos computacionais de microserviços;



Dataset

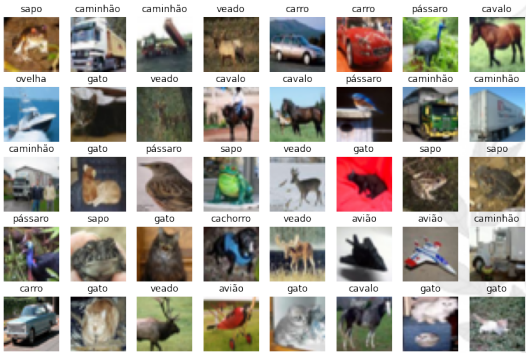
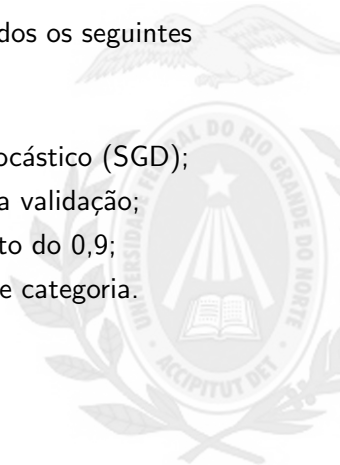


Figura 13: Amostra do dataset cifar10

Treinamento da DNN

Para o treinamento do modelo, foram escolhidos os seguintes parâmetros:

- Tamanho de cada lote de 64;
- Otimizador de Descida de Gradiente Estocástico (SGD);
- 60 mil amostras para treino e 10 mil para validação;
- Taxa de aprendizagem de 0,05 e momento de 0,9;
- Critério de loss como entropia cruzada de categoria.



Arquitetura do modelo

O modelo foi definido com a arquitetura VGG16 com duas camadas densas:

- Entrada($32 \times 32 \times 3$);
- Conv2D($64 \times 3 \times 3$) e ReLU;
- Conv2D($64 \times 3 \times 3$) e ReLU;
- MaxPool(2×2);
- Conv2D($128 \times 3 \times 3$) e ReLU;
- Conv2D($128 \times 3 \times 3$) e ReLU;
- MaxPool(2×2);
- Conv2D($256 \times 3 \times 3$) e ReLU;
- Conv2D($256 \times 3 \times 3$) e ReLU;
- Conv2D($256 \times 3 \times 3$) e ReLU;
- MaxPool(2×2);
- Conv2D($512 \times 3 \times 3$) e ReLU;
- Conv2D($512 \times 3 \times 3$) e ReLU;
- Conv2D($512 \times 3 \times 3$) e ReLU;
- MaxPool(2×2);
- Dense(4096) e ReLU;
- Dense(4096) e ReLU;
- Dense(10) e Softmax.

Resultados do treinamento

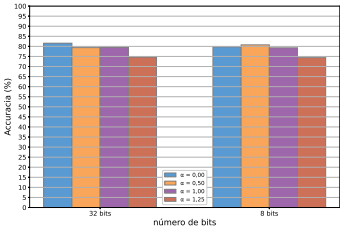
Tabela 4: Acurácias obtidas na classificação de imagens a partir do modelo comprimido para vários valores de quantidade de bits e α .

| <div><div>α</div><div>bits</div></div> | 0,00 | 0,50 | 1,50 | 1,25 |
|--|--------|--------|--------|--------|
| 32 bits | 81,58% | 79,39% | 79,53% | 74,64% |
| 8 bits | 79,76% | 80,77% | 79,31% | 74,42% |

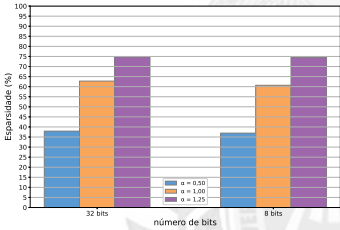
Tabela 5: Esparsidades obtidas na classificação de imagens a partir do modelo comprimido para vários valores de tamanho de bits e α .

| <div><div>α</div><div>bits</div></div> | 0,50 | 1,50 | 1,25 |
|--|--------|--------|--------|
| 32 bits | 37,92% | 62,76% | 74,76% |
| 8 bits | 36,94% | 60,68% | 74,49% |

Resultados do treinamento



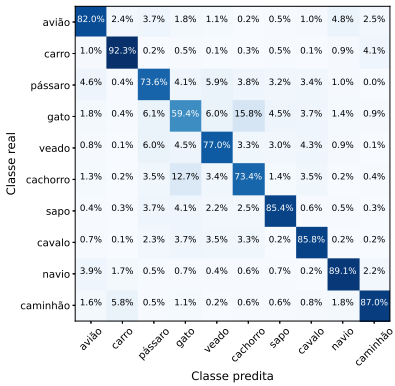
(a) Acurácia dos modelos



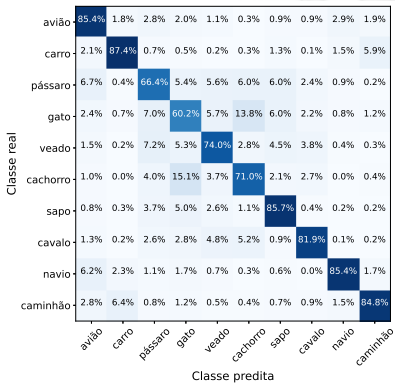
(b) Esparsidade dos modelos

Figura 14: Acurácias e esparsidades obtidas na classificação de imagens a partir do modelo comprimido para vários valores de tamanho de bits e α .

Resultados do treinamento



(a) Modelo não comprimido



(b) 8 bits e $\alpha = 1,25$

Figura 15: Matrizes de confusão do modelo sem compressão e do mais comprimido.

Resultados do treinamento

Tabela 6: Tamanho do modelo comprimido para vários valores de quantidade de bits e α em MegaBytes após compressão por Deflate

| <div><div>α</div><div>bits</div></div> | 0,00 | 0,50 | 1,50 | 1,25 |
|--|----------|---------|---------|---------|
| 32 bits | 123,51MB | 87,11MB | 55,80MB | 39,39MB |
| 8 bits | 32,50MB | 32,12MB | 22,09MB | 16,20MB |

Resultados do treinamento

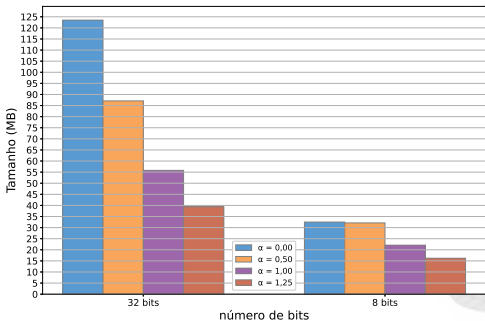


Figura 16: Tamanho do modelo comprimido para vários valores de quantidade de bits e α em MegaBytes após compressão por Deflate

Infraestrutura

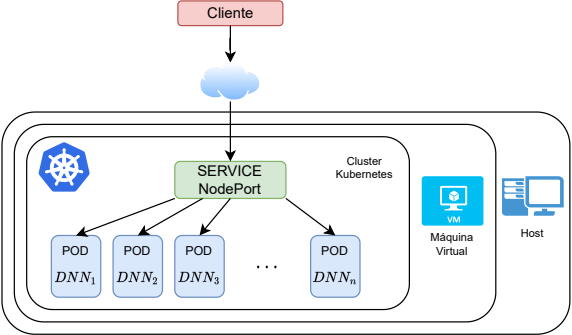
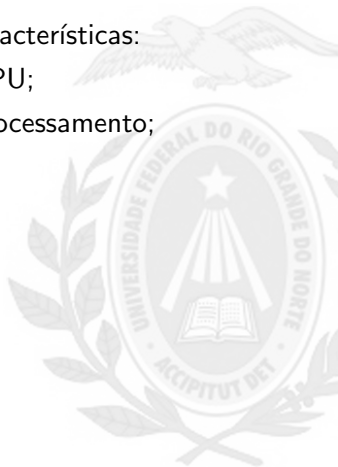


Figura 17: Infraestrutura geral desenvolvida

Configurações dos pods

Cada pod foi construído com as seguintes características:

- Máximo de processamento em 100miliCPU;
- Nova réplica com 20% do máximo de processamento;
- Máximo de 10 réplicas.



Perfil de estresse

Foram criados perfis de estresse utilizando o *Open Model Thread Group* da ferramenta Apache JMeter, para variar a quantidade de requisições do sistema com o passar do tempo.

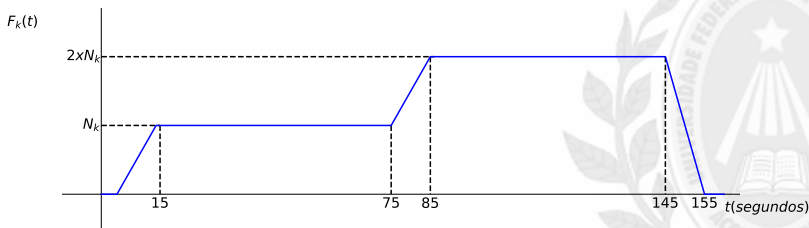


Figura 18: Perfil de estresse com n variando em 75, 125 e 250

Resultados

Tabela 7: Latência de resposta das requisições, Consumo de memória e processamento de cada microserviço do primeiro perfil de estresse $N_1 = 75$.

| bits | α | Latência | Memória | CPU |
|------|----------|---------------------|---------|------------|
| 32 | 0,00 | 16,23ms \pm 2,988 | 701MiB | 135miliCPU |
| 32 | 0,50 | 15,97ms \pm 3,467 | 707MiB | 136miliCPU |
| 32 | 1,00 | 16,02ms \pm 2,659 | 701MiB | 143miliCPU |
| 32 | 1,25 | 15,44ms \pm 2,597 | 703MiB | 132miliCPU |
| 8 | 0,00 | 9,39ms \pm 2,026 | 412MiB | 95miliCPU |
| 8 | 0,50 | 9,27ms \pm 3,011 | 414MiB | 94miliCPU |
| 8 | 1,00 | 9,57ms \pm 2,144 | 411MiB | 96miliCPU |
| 8 | 1,25 | 9,44ms \pm 1,729 | 411MiB | 94miliCPU |

Resultados

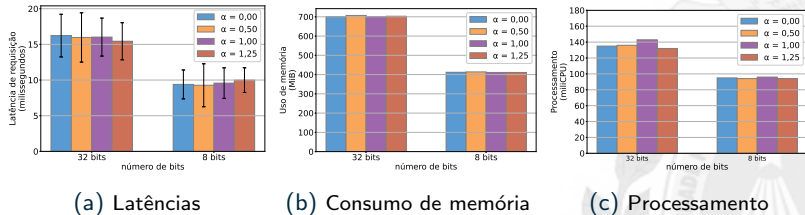


Figura 19: Latência de resposta em milissegundos, consumo de memória e uso de CPU de cada microserviço para o primeiro perfil de estresse $N_1 = 75$.

Resultados

Tabela 8: Latência de resposta das requisições, Consumo de memória e processamento de cada microserviço do segundo perfil de estresse $N_2 = 125$.

| bits | α | Latência | Memória | CPU |
|------|----------|---------------------|---------|------------|
| 32 | 0,00 | 16,01ms \pm 2,755 | 701MiB | 136miliCPU |
| 32 | 0,50 | 16,44ms \pm 3,134 | 704MiB | 137miliCPU |
| 32 | 1,00 | 15,51ms \pm 2,299 | 703MiB | 139miliCPU |
| 32 | 1,25 | 15,23ms \pm 3,101 | 703MiB | 136miliCPU |
| 8 | 0,00 | 9,22ms \pm 2,577 | 411MiB | 94miliCPU |
| 8 | 0,50 | 8,68ms \pm 3,822 | 412MiB | 94miliCPU |
| 8 | 1,00 | 8,67ms \pm 3,715 | 411MiB | 95miliCPU |
| 8 | 1,25 | 9,58ms \pm 3,122 | 411MiB | 94miliCPU |

Resultados

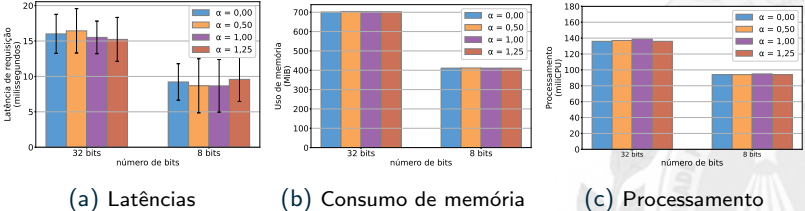


Figura 20: Latência de resposta em milissegundos, consumo de memória e uso de CPU de cada microserviço para o segundo perfil de estresse $N_2 = 125$.

Resultados

Tabela 9: Latência de resposta das requisições, Consumo de memória e processamento de cada microserviço do terceiro perfil de estresse $N_3 = 250$.

| bits | α | Latência | Memória | CPU |
|------|----------|---------------------|---------|------------|
| 32 | 0,00 | 16,23ms \pm 3,122 | 702MiB | 135miliCPU |
| 32 | 0,50 | 16,78ms \pm 3,214 | 706MiB | 136miliCPU |
| 32 | 1,00 | 16,24ms \pm 3,333 | 704MiB | 139miliCPU |
| 32 | 1,25 | 15,34ms \pm 2,767 | 701MiB | 134miliCPU |
| 8 | 0,00 | 9,30ms \pm 2,666 | 412MiB | 94miliCPU |
| 8 | 0,50 | 9,25ms \pm 2,714 | 414MiB | 95miliCPU |
| 8 | 1,00 | 9,25ms \pm 2,555 | 410MiB | 96miliCPU |
| 8 | 1,25 | 8,71ms \pm 2,991 | 411MiB | 94miliCPU |

Resultados

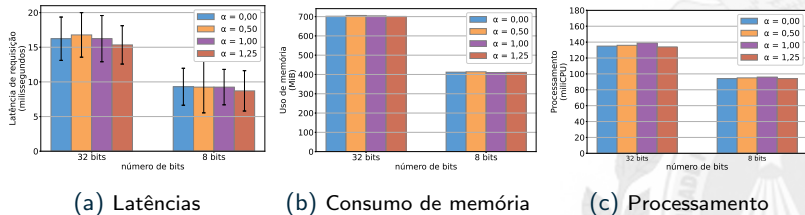


Figura 21: Latência de resposta em milissegundos, consumo de memória e uso de CPU de cada microserviço para o terceiro perfil de estresse $N_3 = 250$.

Resultados

Tabela 10: Tempo médio para criação de uma nova réplica.

| <div><div>α</div><div>bits</div></div> | 0,00 | 0,50 | 1,50 | 1,25 |
|--|----------------|----------------|----------------|----------------|
| 32 bits | 2,71s ± 0,0220 | 2,72s ± 0,0267 | 2,57s ± 0,0306 | 2,41s ± 0,0294 |
| 8 bits | 2,30s ± 0,0199 | 2,25s ± 0,0444 | 2,22s ± 0,0212 | 2,14s ± 0,0409 |

Resultados

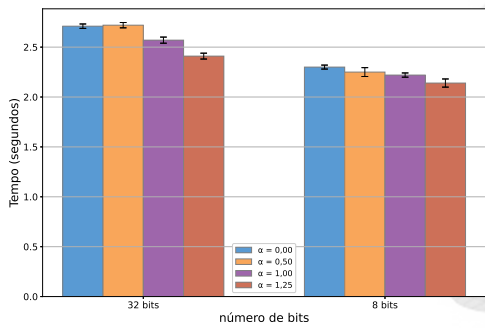


Figura 22: Tempo médio para criação de uma nova réplica.

Resultados

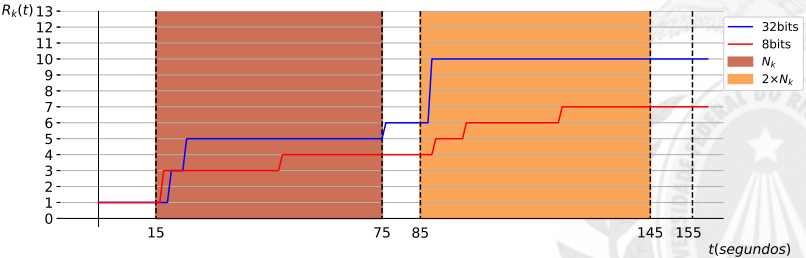


Figura 23: Comportamento da criação de novas réplicas com o passar do tempo do modelo não comprimido e quantizado em 8 bits o primeiro perfil de estresse $N_1 = 75$.

Resultados

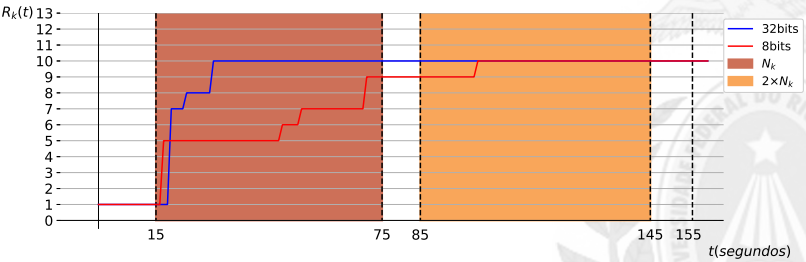


Figura 24: Comportamento da criação de novas réplicas com o passar do tempo do modelo não comprimido e quantizado em 8 bits o segundo perfil de estresse $N_2 = 125$.

Resultados

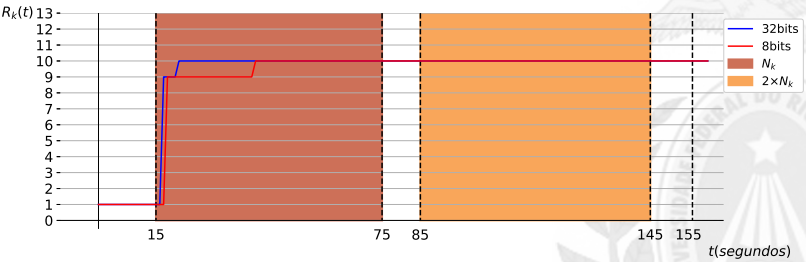
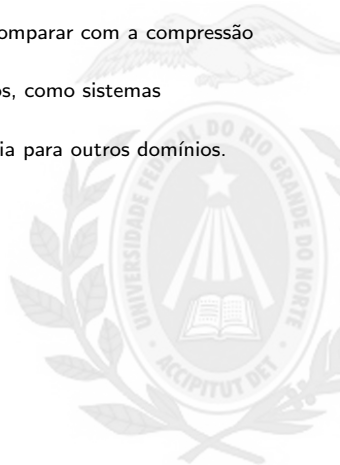



Figura 25: Comportamento da criação de novas réplicas com o passar do tempo do modelo não comprimido e quantizado em 8 bits o terceiro perfil de estresse $N_3 = 250$.


Trabalhos futuros


- Utilizar outra estratégia para geração de modelo e comparar com a compressão de Deflate, como o formato esparsos;
- Aplicar a compressão de modelos em outros domínios, como sistemas embarcados;
- Inclusão de novas métricas, como consumo de energia para outros domínios.





Referências


 BLALOCK, D. et al. *What is the state of neural network pruning?* 2020. Acesso em: 8 jul. 2021. Disponível em: <https://arxiv.org/abs/2003.03033>.


 FERNANDES, M. A. C.; KUNG, H. T. A novel training strategy for deep learning model compression applied to viral classifications. In: IEEE INTERNATIONAL JOINT CONFERENCE ON NEURAL NETWORKS (IJCNN). [S.l.], 2021.

 NVIDIA. *GPU-Based Deep Learning Inference: A Performance and Power Analysis*. 2015. Acesso em: 10 jul. 2021. Disponível em: https://www.nvidia.com/content/tegra/embedded-systems/pdf/jetson_tx1_whitepaper.pdf.

 RAKIN, A. S. et al. Defend deep neural networks against adversarial examples via fixed and dynamic quantized activation functions. *arXiv preprint arXiv:1807.06714*, 2018.

 TUNG, F.; MORI, G. Deep neural network compression by in-parallel pruning-quantization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 42, n. 3, p. 568–579, 2020.

 WANG, K. et al. Haq: Hardware-aware automated quantization with mixed precision. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2019. p. 8612–8620.

 Zhe, W. et al. Optimizing the bit allocation for compression of weights and activations of deep neural networks. In: *2019 IEEE International Conference on Image Processing (ICIP)*. [S.l.: s.n.], 2019. p. 3826–3830.