MATEUS ARNAUD SANTOS DE SOUSA GOLDBARG

ANÁLISE DE TÉCNICAS DE COMPRESSÃO EM REDES NEURAIS PROFUNDAS POR PODA EM DATASET DE IMAGENS

Natal – RN

Setembro de 2021

MATEUS ARNAUD SANTOS DE SOUSA GOLDBARG

ANÁLISE DE TÉCNICAS DE COMPRESSÃO EM REDES NEURAIS PROFUNDAS POR PODA EM DATASET DE IMAGENS

Trabalho de Conclusão de Curso de Engenharia de Computação da Universidade Federal do Rio Grande do Norte, apresentado como requisito parcial para a obtenção do grau de Bacharel em Engenharia de Computação

Orientador: Marcelo Augusto Costa Fernandes

Universidade Federal do Rio Grande do Norte – UFRN

Departamento de Engenharia de Computação e Automação – DCA

Curso de Engenharia de Computação

Natal – RN

Setembro de 2021

MATEUS ARNAUD SANTOS DE SOUSA GOLDBARG

ANÁLISE DE TÉCNICAS DE COMPRESSÃO EM REDES NEURAIS PROFUNDAS POR PODA EM DATASET DE IMAGENS

Trabalho de Conclusão de Curso de Engenharia de Computação da Universidade Federal do Rio Grande do Norte, apresentado como requisito parcial para a obtenção do grau de Bacharel em Engenharia de Computação

Orientador: Marcelo Augusto Costa Fernandes

Trabalho aprovado. Natal – RN, 10 de Setembro de 2021:

Prof. Dr. Marcelo Augusto Costa Fernandes - Orientador
UFRN

Dr. Sérgio Natan Silva - Convidado
UFRN

Dr. José Cláudio Vieira e Silva Júnior
UFRN

Natal – RN
Setembro de 2021

AGRADECIMENTOS

A Débora, pelo apoio e por seu incomparável consolo e amor. Por sempre me ajudar a escolher o melhor, por entender minha ausência enquanto eu me dedicava a realização deste trabalho e por sua dedicação em fazer com que tudo ficasse bem.

Aos meus pais, Gilmar e Fabiola, por me apoiarem no momentos difíceis, que me aconselharam nos momentos de dúvidas e incertezas, e que sempre estiveram prontos para me levantar a qualquer sinal de fraqueza.

A meu irmão, Rafael, que sempre está me divertindo e fezendo com que momentos de dificuldades se tornem leves com momentos diviertidos. E também por sua ajuda nos últimos momentos do desenvolvimento deste trabalho.

A meu orientador, professor Marcelo Augusto Costa Fernandes, pela sua dedicação, compreensão e pelo compartilhar de seu conhecimento.

Aos meus professores, pelos ensinamentos e correções que me ajudaram a desenvolver novas habilidades.

RESUMO

Técnicas de redes neurais profundas tem sido utilizadas com sucesso para classificação de imagens a partir da utilização de redes neurais convolucionais. Porém, algoritmos de deep learning realizam uma grande quantidade de operações matemáticas. Essas operações podem ser um gargalo no processo de grandes quantidades de imagens em um curto período de tempo. Em microcontroladores de baixo custo, essas operações podem resultar em um aumento significativo do consumo energético, mostrando assim a necessidade da aplicação de técnicas de compressão dessas redes. Atualmente, a maioria das redes profundas utilizadas para classificação de imagens não são otimizadas. A proposta deste trabalho é otmizar uma rede neural convolucional a partir da técnica de compressão de dados por poda. Durante o treino, a técnica é remover os pesos a cada *batch* ao invés da remoção dos pesos apenas no primeiro *batch* de cada época. Essa estratégia foi aplicada para classificação de 10 mil imagens de 10 classes diferentes. Foi possível remover aproximadamente 82% dos parâmetros da rede neural profunda mantendo uma alta acurácia. Esses resultados mostram que a técnica de remoção de pesos por *batch* se mostrou eficaz para essa aplicação.

Palavras-chaves: Classificação de imagens. Redes Neurais Profundas. Compressão de Modelo. Poda. Treinamento de Modelo.

ABSTRACT

Deep neural network techniques have been successfully used for image classification using convolutional neural networks. However, deep learning algorithms perform a lot of mathematical operations. These operations can be a bottleneck in the process of large amounts of images. In low-cost microcontrollers, these operations can result in a significant increase in energy consumption, showing the need to apply compression techniques for these networks. Currently, most of the deep networks used for image classification are not optimized. The purpose of this work is to optimize a convolutional neural network using the technique of data compression by pruning. During training, the technique is to remove the weights at each batch, instead of removing weights only in the first batch of each epoch. This strategy was applied to classify 10,000 images from 10 different classes. It was possible to remove approximately 82% of the parameters from the deep neural network while maintaining high accuracy. These results shows that the batch weight removal technique proved to be effective for this application.

Keywords: Image classification. Deep Neural Networks. Model Compression. Pruning. Model Training.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ANN	*Rede Neural Artificial*
CNN	*Rede Neural Convolucional*
DNN	*Rede Neural Profunda*
IA	*Inteligência Artificial*
IoT	*Internet das Coisas*
ML	*Aprendizado de Máquina*
RGB	*Vermelho, Verde e Azul (3 canais)*

LISTA DE SÍMBOLOS

α	Letra grega Alpha
β	Letra grega Beta
σ	Letra grega Sigma

SUMÁRIO

1 INTRODUÇÃO

A Inteligência Artificial é um campo que vem oferecendo, e que ainda oferecerá, muitas oportunidades para mercados emergentes e serviços revolucionando assim quase todos os segmentos da sociedade atual. Com as técnicas de aprendizado de máquina (machine learning), é possível fornecer uma ferramenta de alta precisão para resolver problemas de classificação e previsão, como reconhecimento de padrões ou síntese de fala. A Inteligência Artificial já está presente em uma gama de atividades do dia-a-dia como, por exemplo, publicidade, finanças, multimídia, visão computacional, jogos eletrônicos e outras. No entanto, ela requer processamento intensivo e hardwares de alto desempenho, e isso pode ser um problema para aplicações de IoT cujo alvo são hardwares de baixo consumo energético.

Atualmente, a maioria das técnicas mais convencionais de otimização por poda (pruning) realiza a remoção de pesos apenas uma vez por época (normalmente apenas no primeiro *batch*). Neste trabalho é proposto um esquema de compressão de modelo por poda, com remoção dos pesos em todos os *batchs* de todas as épocas, durante o treinamento, em dataset de imagens.

1.1 Contextualização e Motivação

Técnicas de aprendizado profundo (deep learning), como redes neurais profundas, têm sido usados com sucesso na resolução de muitos problemas. Porém, algoritmos desse tipo requerem muitas operações numéricas e esses cálculos podem ser um gargalo para o processamento de muitos dados em um curto período de tempo. Assim, o processamento de redes neurais convolucionais incorre em alto consumo de energia devido sua alta complexidade computacional (NVIDIA, 2015).

O estudo sobre as formas de acelerar a computação no aprendizado profundo tem sido um foco no aprendizado de máquina, e sua literatura tem crescido rapidamente. Uma das abordagens convencionais de compressão é a poda (pruning), que remove parâmetros sistematicamente de uma rede já existente (BLALOCK et al., 2020). O desafio é remover esses parâmetros e diminuir o tamanho da rede de forma que afete minimamente a sua acurácia. A remoção desses parâmetros resulta na diminuição da quantidade de operações matemáticas necessárias.

1.2 Resumo bibliográfico

Em (BLALOCK et al., 2020) os autores propõem a utilização da técnica de poda e avaliação dos resultados com modelos de redes neurais e diferentes datasets baseados no tamanho do modelo, velocidade e acurácia.

No artigo (HAN; MAO; DALLY, 2015) os autores sugerem a redução do tamanho do modelo e da energia requerida para inferencia de uma rede neural, utilizando técnicas de compressão, para ser utilizada em dispositivos móveis.

Em (FERNANDES; KUNG, 2021) os autores propõem a compressão de dados utilizando as técnicas de poda e quantização aplicadas a classificação viral.

Em (HUANG et al., 2018) é proposto a utilização da técnica de compressão de poda de filtros nas camadas convolucionais, em redes neurais convolucionais, e introduz o algoritmo de try-and-learn no aprendizado de máquina.

No artigo (YANG; CHEN; VIVIENNE, 2017) os autores trazem a análise de consumo de energia em redes neurais convolucionais antes e depois da utilização da técnica de compressão por poda em todas as camadas dos modelos utilizados, reduzindo o número de parâmetros, seu tamanho e a complexidade do algorítmo.

No artigo (REED, 1993) o autor propõe a utilização do algoritmo de poda em uma rede neural já treinada e analíse da eficiência do modelo após sua aplicação.

Em (NVIDIA, 2015) os autores propõem um estudo sobre o consumo energético durante a inferência de redes neurais convolucionais.

Diante do apresentado, é notável que a técnica de compressão de dados por poda está sendo implementada e analisada em diferentes contextos, mas com o objetivo de redução do tamanho e da complexidade das redes neurais já treinadas tanto para forma de análise quanto para aplicações em sistemas embarcados para diminuição do consumo energético. Porém, a utilização da remoção de pesos em todos os *batchs* foi muito pouco explorado, sendo utilizado apenas uma vez para classificação viral.

1.3 Objetivo

O objetivo deste trabalho é desenvolver a técnica de compressão de redes neurais profundas por poda, utilizando um dataset de classificação de imagens, aplicando a técnica de remoção de pesos por poda a cada *batch* e analisar a quantidade de pesos que podem ser removidos sem que a acurácia do modelo seja muito prejudicada.

1.4 Estrutura do Trabalho

Este trabalho apresenta uma introdução sobre o tema, mostrando os fatores que motivam a implantação da ideia, além da motivação e dos objetivos. Em sequência, o Capítulo 2 aborda os principais conceitos teóricos e técnicos utilizados no trabalho. O Capítulo 3, por sua vez, descreve os detalhes de organização e implementação de todas as partes da ténica proposta, bem como os procedimentos adotados para os testes e validação, enquanto o Capítulo 4 trata de mostrar os resultados obtidos experimentalmente e apresenta discussões a respeito destes resultados. Por fim, o Capítulo 5 traz as principais conclusões e contribuições deste trabalho.

2 REFERENCIAL TEÓRICO

Neste capítulo, são apresentados os principais conceitos utilizados neste trabalho, sendo eles: Redes Neurais Convolucionais com os seus hiperparâmetros e técnicas de compressão de dados com ênfase na no método da poda (pruning).

2.1 Redes Neurais Convolucionais (CNN)

A rede neural convolucional é uma das classes de redes neurais utilizada para processamento e análise de imagens. Ela se diferencia dos outros tipos de redes neurais artificiais por possuir camadas convolucionais antes das camadas *fully conected* e está presente em uma gama de aplicações como busca de imagens, recomendações de produtos, sistema de busca, reconhecimento facial entre outros.

Assim como as ANNs, as redes neurais convolucionais possuem pesos que são alterados durante o processo de aprendizagem, a diferença é que nas camadas convolucionais, os pesos são organizados em *kernels*, também chamados de filtros. Além das camadas convolucionais, é bastante comum a presença de camadas de *pooling*, também conhecidas como camadas de agrupamento.

2.1.1 Camada convolucional

A camada convolucional da CNN consiste em um conjunto de filtros que serão operados por toda a imagem de entrada. Como o próprio nome diz, a operação realizada entre os filtros e a imagem de entrada é a chamada convolução, onde cada pixel da imagem de saída é o resultado da operação entre os elementos do filtro e regiões da imagem de entrada. Os hiperparâmetros desta camada incluem o tamanho do filtro, a quantidade de filtros, o *stride* (passo) e se haverá, ou não, *padding* (preenchimento). Se houver preenchimento, o tamanho da imagem de saída é igual ao de entrada. Porém, se não houver preenchimento, a imagem de saída será reduzida a depender do tamanho do *kernel* e do *stride* escolhido. Assim, o tamanho da imagem é dado por:

$$L_s = L_e + k_s - 1e \tag{2.1}$$

$$A_s = A_e + k_s - 1, \tag{2.2}$$

Sendo L_e e A_e a lagura e altura da imagem de entrada, L_s e A_s a lagura e altura da imagem de saída e sendo k_s o tamanho do *kernel*. A Figura 1 mostra uma representação de como acontece o processo de convolução quando não há padding.

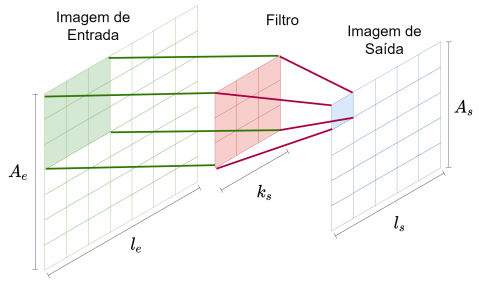


Figura 1 – Operação de convolução

Fonte – Elaborada pela autor.

2.1.2 Camada de Agrupamento (*Pooling*)

Na camada de agrupamento é realizado o processamento da imagem de entrada para redução da quantidade de parâmetros. Isso se faz selecionando, dentre um certo valor de parâmetros dentro da janela de *pooling*, aquele que se adeque com o desejado. Normalmente, o agrupamento é feito a partir dos valores mais altos do grupo, através do *max-polling*. Porém, o agrupamento pode ser realizado a partir do mínimo, do valor médio, entre outros.

Assim como a camada convolucional, é necessário escolher alguns hiperparâmetros para essa camada, como o tamanho da janela do agrupamento e o seu *stride*. A Figura 2 apresenta a operação de *max-pooling* para uma janela de agrupamento de tamanho 2 e *stride* de 2, onde cada pixel da imagem de saída é o maior elemento dentro da janela de 4 elementos.

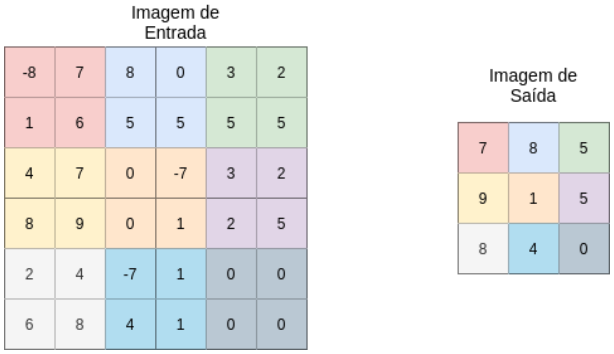


Figura 2 – Operação de *max-pooling*

Fonte – Elaborada pela autor.

2.1.3 Camada de *flattening*

A Figura 3 ilustra o processo de flattening, que é responsável por realiza uma transformação na matriz da imagem, alterando seu formato para um vetor. Por exemplo, se a entrada da camada de *flattening* for uma matriz de 32x32x3, a saída será um vetor de 3072 posições.

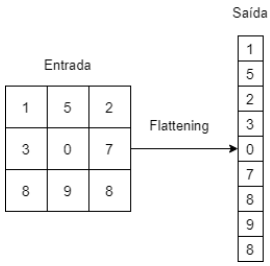


Figura 3 – Operação de *flattening*

Fonte – Elaborada pela autor.

2.1.4 Camada *fully-connected* (totalmente conectada)

A camada totalmente conectada é a camada mais utilizada em aplicações de redes neurais. Ela se forma a partir da conexão de todos os neurônios de entrada e saída por pesos. Como hiperparâmetro, pode-se definir a quantidade de parâmetros a serem utilizados. A Figura 4 apresenta um modelo convencional de camadas totalmente conectadas.

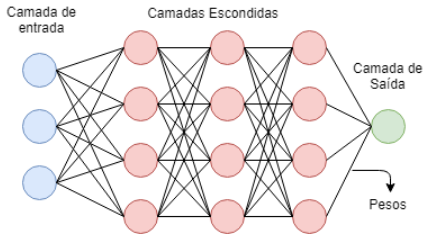


Figura 4 – Estrutura e operações da camada *fully-connected*

Fonte – Elaborada pela autor.

A Figura 5 ilustra as operações realizadas entre entre os neurônios e os pesos, onde X_n é o valor armazenado no n-ésimo neurônio e w_n é o peso que conecta o neurônio da camada anterior com o neurônio da camada atual.

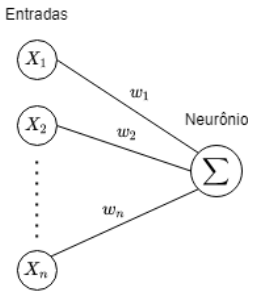


Figura 5 – Operações realizadas na camada *fully-connected*

Fonte – Elaborada pela autor.

2.1.5 Funções de ativação

A aplicação da função de ativação pode ser utilizada em qualquer camada da rede. Ela é a transformação não linear que é feita ao longo da entrada. A saída desta função é enviada para a próxima camada. Uma rede neural sem função de ativação torna-se apenas um modelo linear, sendo que a maioria dos problemas complexos resolvidos por redes neurais são não-lineares.

Existem diversas funções de ativação diferentes. Para este trabalho foram utilizadas apenas duas: a ReLU e a softmax. A função ReLU é definida na Equação 2.3, onde y é a saída da função que retorna o valor máximo entre 0 e o valor de x, que é o parâmetro sendo analisado. A Figura 6 mostra o gráfico dessa função.

$$y = max(0, x) \tag{2.3}$$

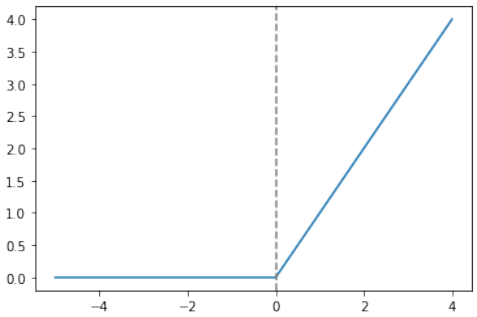


Figura 6 – Função de ativação ReLU

Fonte – Elaborada pela autor.

A função de ativação softmax, também conhecida como softargmax ou função exponencial normalizada, é uma função que recebe um vetor com z números reais e o normaliza em uma distribuição probabilistica para os z valores. Essa função é normalmente utilizada na camada de saída para que seja representada a probabilidade da imagem em

análise pertencer a cada uma das classes. Essa função é definida na Equação 2.4, onde z_i é o i-ésimo termo do vetor z e z_j é o j-ésimo termo do mesmo vetor. Essa função é representada graficamente na Figura 7.

$$S(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}} \tag{2.4}$$

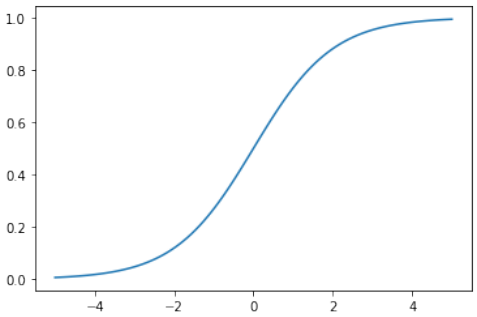


Figura 7 – Função de ativação softmax

Fonte – Elaborada pela autor.

2.2 Técnicas de compressão de modelo

As estratégias para aceleração de inferência podem ser classificadas em baseadas em algoritmos e baseada em sistemas. Abordagens de aceleração baseadas em algorítmos incluem: processamento paralelo, compressão de modelo, exploração de esparsidade e redução de modelo por aproximação. Já as abordagens de aceleração baseadas em sistemas abrangem sistemas de memória de alta largura de banda para minimizar o tempo de acesso à memória, aceleradores de hardware, sistemas de computação distribuída e otimizações de compilador, como eliminação de expressão comum. A estratégia baseada em algoritmos usando compresão, foco deste trabalho, é uma forma comum na redução do número de operações nas redes neurais profundas.

2.2.1 Exploração de esparsidade

Uma matriz é dita esparsa quando a maior parte de seus elementos são zeros. A exploração da esparsidade pode estar presente em multiplos níveis: *bit*, valor, canal, filtro, bloco entre outros. O valor da esparsidade de certa camada, ou modelo, é encontrado a partir da Equação 2.5, onde n_{zeros} representa a quantidade de zeros e n_{pesos} representa a quantidade de pesos.

$$esparsidade = \frac{n_{zeros}}{n_{pesos}} * 100 \tag{2.5}$$

A esparsidade também pode ser computada a partir da contagem de elementos com magnitude muito próximas a zero, ou não significativos. A Tabela 1 apresenta um exemplo da esparsidade de uma rede neural com 65536 parâmetos, onde W_k representa o limite que define a magnitude dos pesos que são considerados insignificantes para o modelo a coluna de Quantidade de pesos mostra quantos pesos são removidos para cada limite de W_k e a teceira coluna mostra as esparsidades do modelo.

Mag. do peso	Qtd. de pesos	Esparsidade
$W_k < 0.01$	6438	9,82%
$W_k < 0.001$	3220	4,91%
$W_k < 0.0001$	1924	2,94%
Rede neural com 65536 pesos		

Tabela 1 – Esparsidade para pesos com diferentes magnitudes

2.2.2 Compressão de modelo por poda

Atualmente, diferentes técnicas de compressão de modelo são utilizados, como quantização da rede, poda e ajuste fino de modelos pré-treinados. A compressão de modelo também pode ser realizada a partir da quantização seguida da poda ou da poda seguida da quantização.

A otimização utilizando a técnica de poda, objeto deste trabalho, consiste na remoção de pesos insignificantes do modelo a partir de um certo *threshold* (limiar). A definição deste limiar expressa o quão agressiva será a poda no modelo, sendo que a remoção de muitos pesos pode afeter significativamente a acurácia da rede. A remoção desses pesos faz com que a quantidade de operações realizadas durante a inferêmcia da rede seja menor que a do modelo não comprimido. A figura 8 mostra como acontece a remoção dos pesos do modelo.

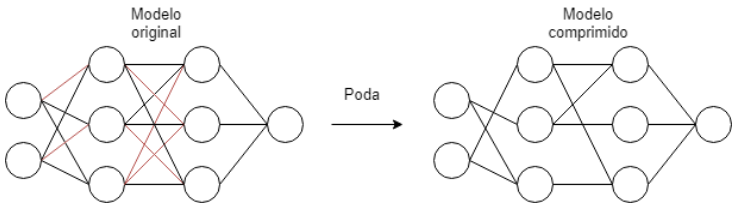


Figura 8 – Representação da técnica de poda

Fonte – Elaborada pela autor.

A remoção dos pesos ocorre durante o loop de aprendizagem do modelo. As técnicas mais convencionais realizam a remoção dos pesos apenas no primeiro *batch* de cada época, outras realizam o mesmo processo apenas no final de cada época. O *loop* de aprendizado de *prune-aware* pode ser visualizado na Figura 9. Onde $Y(n)$ é a classe predita do modelo

para a entrada X_n, $Yref(n)$ é a classe real da entrada, $E(n)$ é o erro obtido pela diferença entre a saída predita e a saída real, $G(n)$ é a aplicação do gradiente de decida levando em consideração tanto o erro quanto os pesos removidos. $W(n)$ é o n-ésimo peso, Z^{-1} é um atraso e $C(n)$ é o resulado do peso que sofreu a poda. Para o controle do momento que deverá haver a poda, existe uma chave de controle de época que fecha ou abre o caminho com a remoção do peso quando desejado.

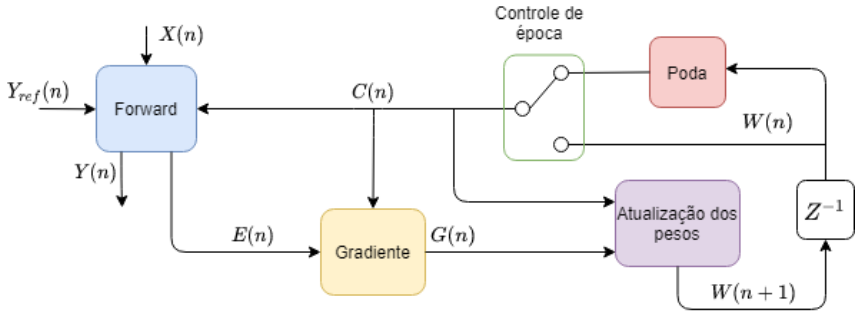


Figura 9 – loop de aprendizado de *prune-aware* convencional

Fonte – Elaborada pela autor.

3 METODOLOGIA

Nesta seção, será apresentada de forma técnica e descritiva a metodologia utilizada para o desenvolvimento da ténica de compressão de redes neurais profundas por poda, análise dos resultados obtidos de acurácia e esparsidade para modelos sem utilização de poda e para podas mais agressivas.

A metodologia adotada para realizar os primeiros experimentos foi, inicialmente, a seleção dos dados a serem utilizados para classificação, definição do modelo de rede e definição das bibliotecas e o ambiente de desenvolvimento a ser utilizado. Após isso, foram definido os hiperparametrose as métricas a serem analisadas para os resultados obtidos. E então, o processo de aprendizagem foi realizado.

3.1 Proposta

Uma das características das redes neurais é a grande quantidade de operações numéricas realizadas durante os processos de treinamento e inferência. Nos hardwares, em geral, muitas operações matemáticas aumentam significativamente o seu consumo energético. Como resultado, aparelhos alimentados por bateria ainda não conseguem executar redes neurais convolucionais de ultima geração devido o seu limite energético (YANG; CHEN; VIVIENNE, 2017).

As técnicas de compressão mais convencionais realizam a poda somente no primeiro batch de cada época, ou ao final de cada época. A proposta é a realização da poda em todos os batches. A aplicação desta técnica já foi utilizada para classificação viral (FERNANDES; KUNG, 2021), a proposta é utilizar a mesma ténica para classificação de imagens.

3.1.1 Treinamento

A estratégia escolhida para redução da quantidade de operações númericas foi a de compressão por poda. Assim, durante o treinamento, foi aplicada a poda em todas as camadas do modelo baseados em um limiar (β_k) definido pelo desvio padrão dos seus pesos multiplicados por uma constante α_k determinada antes do processo de treinamento. Esta variável é que indica o quão agressiva será a compressão do modelo. Todos os pesos com magnitude menor que esse valor são removidos.

O processo de aprendizado e compressão por poda pode ser vizualizado na Figura 10. Onde $Y(n)$ é a classe predita pelo modelo para a entrada $X(n)$. O erro $E(n)$ é calculado a partir da diferença entre a saída predita e $Y_{ref}(n)$ que é a classe real do dado de entrada.

A partir do erro, é claculado o gradiente de descida $G(n)$ e então os pesos são atualizados. $C(n)$ são os pesos do modelo após o processo da poda. Os pesos que foram zerados também passam pelo cálculo do gradiente.

$$\beta_k = \alpha * \sigma_k \tag{3.1}$$

$$C_k(n) = P(W_k(n), \beta_k) = \begin{cases} w_k(n) & \text{if } |w_k(n)| \geq \beta_k \\ 0 & \text{if } |w_k(n)| < \beta_k \end{cases} \tag{3.2}$$

A Equação 3.1 mostra como é feito o cálculo do limiar, sendo β_k o limiar da k-ésima camada, α a constante que estabelece a agressividade da poda e σ_k o desvio padrão da k-ésima camada.

Na Equação 3.2 temos P(.,.), que representa a função de poda, é feita a análise se o n-ésimo peso da k-ésima camada é menor que o limiar calculado. Se for, o valor do peso passa a ser zero. As Figuras 12 e 13 apresentam um exemplo da remoção dos pesos para α = 0.75, removendo um total de 15638 parâmetros dessa camada.

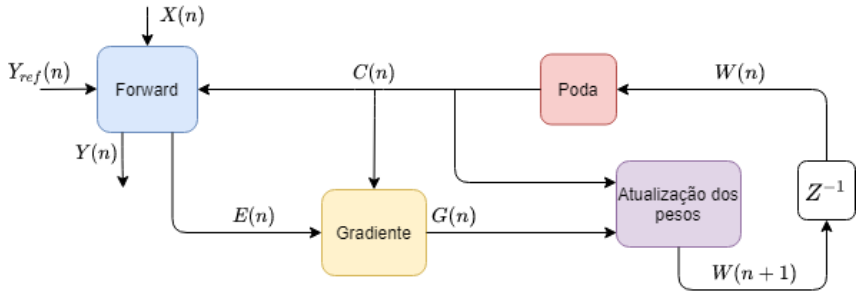


Figura 10 – Diagrama de loop de aprendizado do *pruning-aware*

Fonte – Elaborada pela autor.

A cada iteração do loop de aprendizado, foi contabilizada a quantidade de pesos que são removidos e, após isso, foi calculada a esparsidade do modelo, ou seja, a porcentagem de pesos insignificantes da rede. A análise da eficácia da poda é medida a partir da diferença entre as acurácias de de inferência entre o modelo sem compressão e o modelo com compressão.

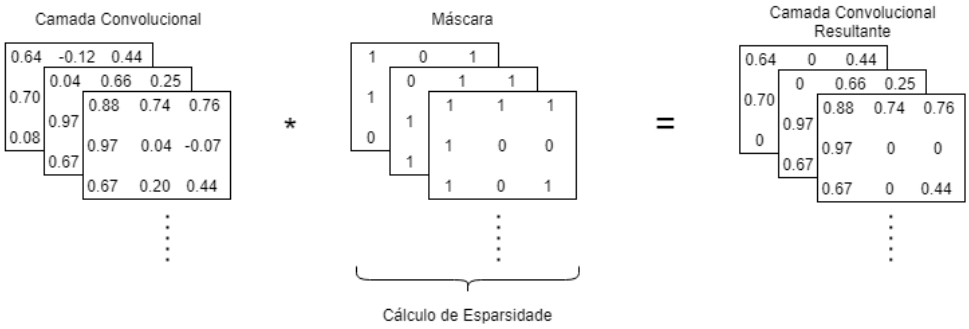


Figura 11 – Processo de remoção de pesos na camada convolucional

Fonte – Elaborada pela autor.

A remoção dos pesos foi realizada a partir da criação de uma máscara, com o mesmo formato da camada do modelo em análise. Cada índice da máscara é preenchido com zero ou um a depender da magnitude do peso associado na camada. Caso seu peso seja menor que o limiar definido, seu índice será zero, caso contrário será 1. A partir da multiplicação entre a camada e a máscara, é obtido como resultado a camada com os pesos a serem removidos e o valor da esparsidade a partir da contagem de pesos zerados. A figura 10 ilustra como funciona o processo de remoção de pesos. As Figuras 12 e 13 mostram o resultado da remoção dos pesos na segunda camada convolucional do modelo. Como pode ser visto na Tabela 2, a segunda camada convolucional possui 18.496 parâmetros, portanto, nesse exemplo, apenas 2.858 parâmetros possuem pesos significativos.

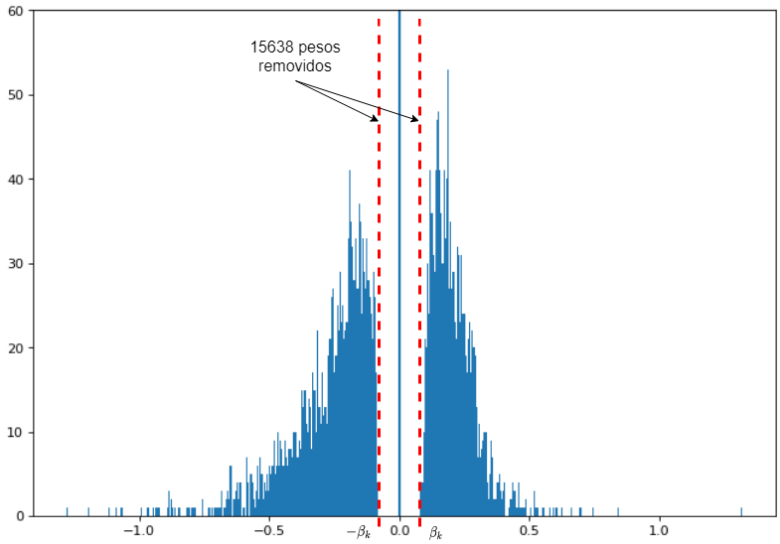


Figura 12 – Histograma de pesos da segunda camada convolucional antes da poda

Fonte – Elaborada pela autor.

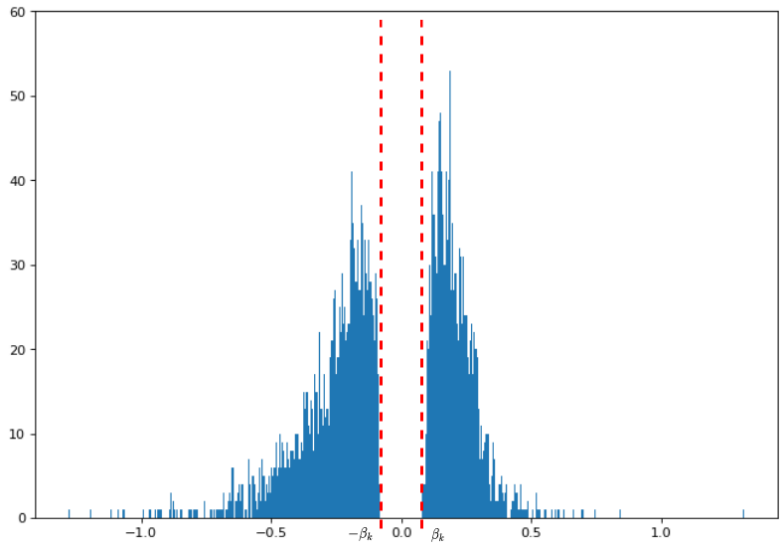


Figura 13 – Histograma de pesos da segunda camada convolucional depois da poda

Fonte – Elaborada pela autor.

3.1.2 Modelo e hiper-parâmetros

O processo de treinamento foi realizado sete vezes, variando em $\alpha = 0{,}00$, $\alpha = 0{,}25$, $\alpha = 0{,}50$, $\alpha = 0{,}75$, $\alpha = 1{,}00$, $\alpha = 1{,}25$ e $\alpha = 1{,}50$. A função utilizada para cáculo das perdas foi a Sparse Categorical Corossentropy e a função do otimizador utilizada foi Adam. Em todos os experimentos, a estrutura da rede foi criada da seguinte forma:

1. Camada de entrada de 32x32x3;
2. Camada convolucional com 32 kernels de 3x3, sem padding, stride igual a 1 e função de ativição relu;
3. Camada convolucional com 64 kernels de 3x3, sem padding, stride igual a 1 e função de ativição relu;
4. Camada de Max-pooling de 2x2, stride igual a 2;
5. Camada convolucional com 128 kernels de 3x3, sem padding, stride igual a 1 e função de ativição relu;
6. Camada convolucional com 128 kernels de 3x3, sem padding, stride igual a 1 e função de ativição relu;
7. Camada de Max-pooling de 2x2, 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 para 10 classes com função de ativação softmax.

A tabela 2 mostra a quantidade de parâmetros de cada camada e também de todo o modelo.

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	

Tabela 2 – Quantidade de parâmetros da rede

3.2 Dataset

O dataset escolhido para esse trabalho chama-se CIFAR-10, que consiste em 60 mil imagens coloridas com 3 canais (RGB) de 32x32 pixels divididas em 10 classes, com 6 mil imagens de cada classe. O dataset é dividido em 50 mil imagens para treino e 10 mil para inferência.

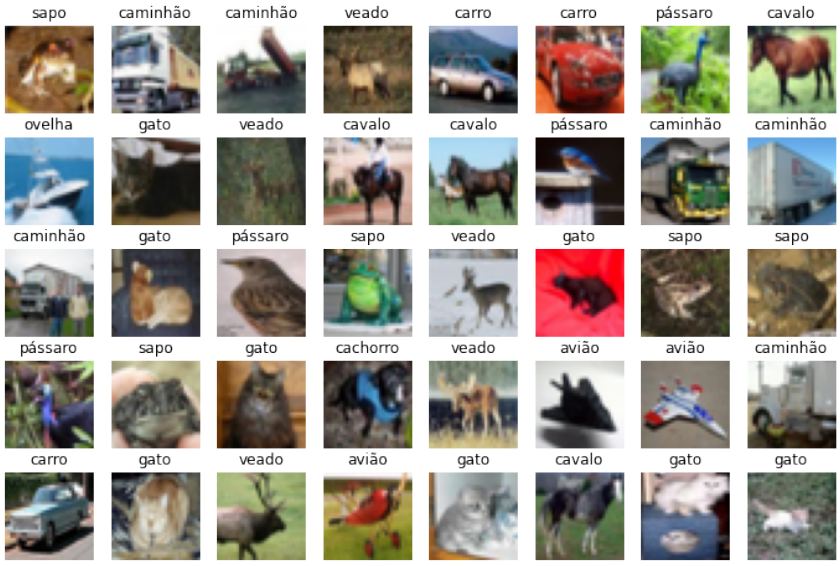


Figura 14 – Amostras do dataset CIFAR-10

Fonte – Elaborada pela autor.

As 10 classes são: avião, automóvel, pássaro, gato, veado, cachorro, sapo, cavalo, barco e caminhão. As classes são exclusivas entre si, ou seja, não há sobreposição entre automóveis e caminhões. A classe de automóveis inclui apenas sedans, SUV's e carros de pequeno porte. A classe de caminhão inclue apenas caminhões grandes, excluindo pickups. A Figura 14 apresenta algumas amostras do dataset.

4 RESULTADOS

Neste capitulo serão apresentados os resultados obtidos e serão feitas algumas observações a respeito desses resultados a partir da modificação do limiar α em sete valores diferentes. Primeiramente, serão apresentados os dados obtidos para cada limiar e após isso será feita a comparação entre os resultados.

4.1 Resultados obtidos para diferentes valores do Limiar

Os resultados apresentados nesta seção foram obtidos a partir da modificação do limiar antes do processo de aprendizagem. Essa modificação é realizada a partir da alteração da consetante α como ja apresentado nos capítulos anteriores deste trabalho.

4.1.1 $\alpha = 0$

O primeiro experimento realizado foi utilizando o $\alpha = 0$, ou seja, não é realizada nenhuma poda no modelo. O primeiro experimento foi executado sem poda para futuras comparações entre o modelo otimizado e o não otimizado. Dessa forma, a esparsidade final é igual a 0%. A partir da Figura 15, é possível ver a variação dos valores de acurácia de treinamento e de validação durante as 30 épocas.

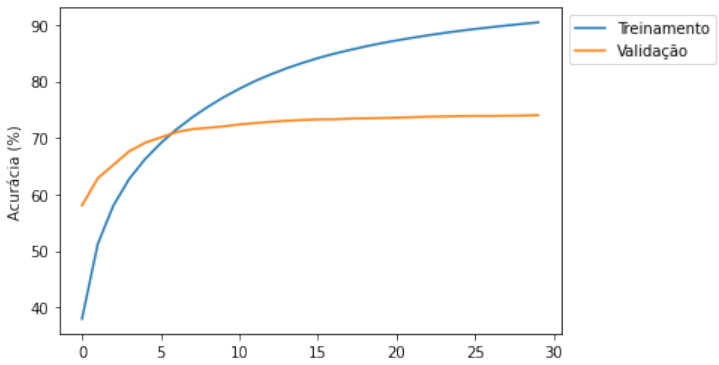


Figura 15 – Acurácia de treinamento e validação para $\alpha = 0$

Fonte – Elaborada pela autor.

Para analisar o comportamento dos pesos ao final do treinamento, foi montado um histograma contendo todos os pesos da segunda camada convolucional do modelo, como visto na Figura 16. É possível perceber que uma grande quantidade de pesos se concentram em valores próximos a zero.

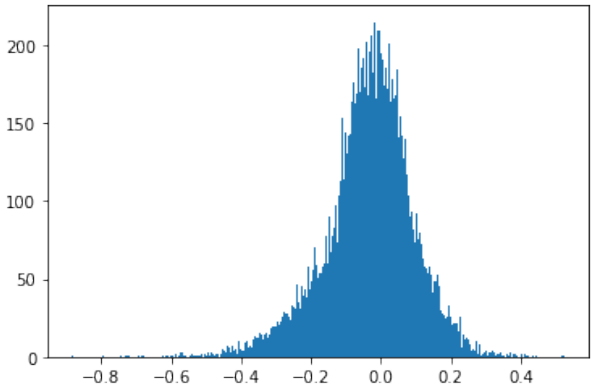


Figura 16 – Histograma de pesos da segunda camada convolucional do modelo para $\alpha = 0$

Fonte – Elaborada pela autor.

No processo de inferência, obteve-se uma acurácia de 75,30%. Para visualização dos resultados, a Figura 17 mostra a matriz de confusão obtida.

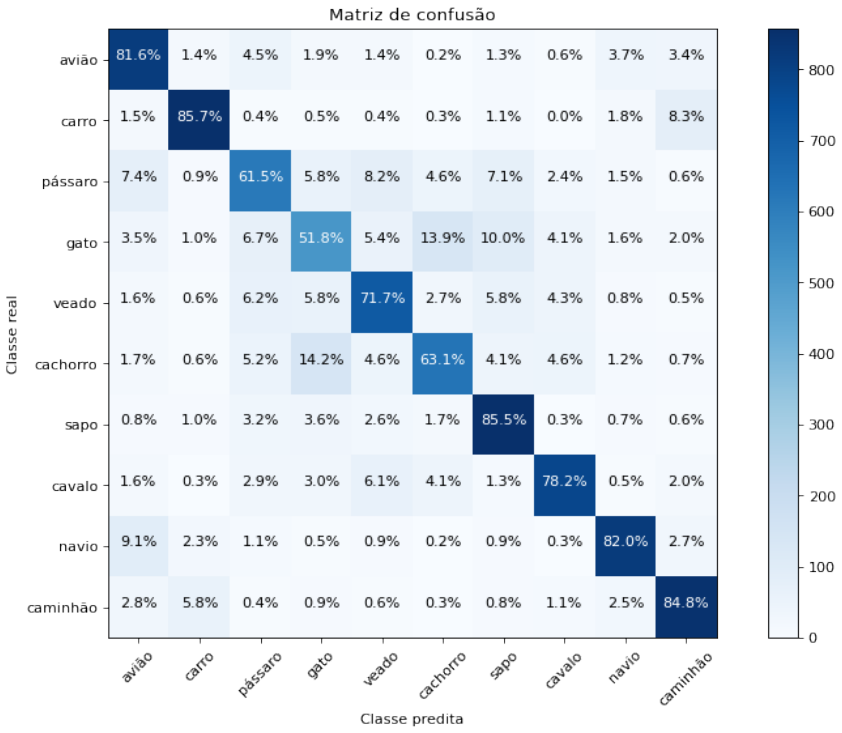


Figura 17 – Matriz de confusão para $\alpha = 0$

Fonte – Elaborada pela autor.

4.1.2 $\alpha = 0{,}25$

Após o experimento realizado sem compressão, foi realizado o mesmo processo de treinamento, porém o α foi alterado para 0,25, sendo esse o valor menos agressivo de poda realizado durante todos os experimentos. Dessa vez, o modelo apresentará esparsidade

maior que 0%, uma vez que haverá poda. A Figura 18 mostra a acurácia de e treinamento e validação para o novo valor do limiar. Já a variação da esparsidade durante as épocas pode ser vizualizada na Figura 19. A esparsidade final do modelo chegou a 52,72%.

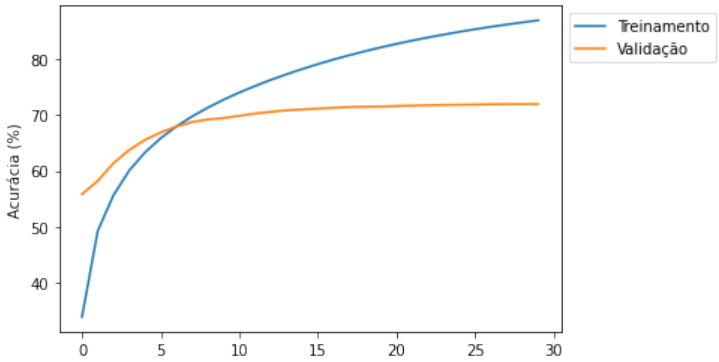


Figura 18 – Acurácia de treinamento e validação para $\alpha = 0{,}25$

Fonte – Elaborada pela autor.

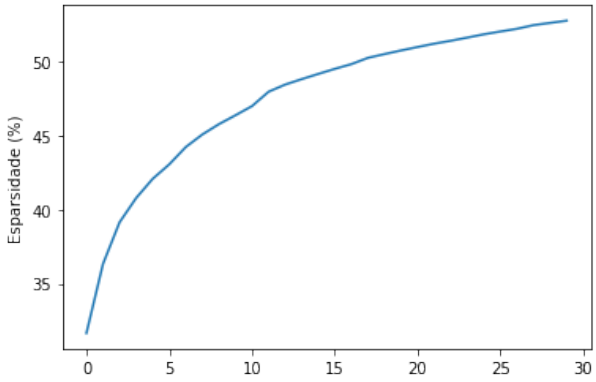


Figura 19 – Variação da esparsidade durante as épocas para $\alpha = 0{,}25$

Fonte – Elaborada pela autor.

A Figura 20 contém o histograma dos pesos da segunda camada convolucional. Devido o processo de compressão há uma grande concentração de pesos com valores muito próximos de zero, ou seja, os pesos a serem removidos por não terem grande influência no processo de classificação. Por esse motivo, foi necessário ampliar o histograma para verificar os pesos maiores que o limiar, como apresentado na Figura 21.

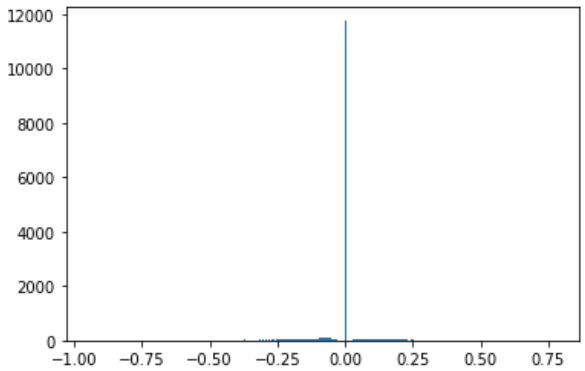


Figura 20 – Histograma de pesos da segunda camada convolucional para $\alpha = 0{,}25$

Fonte – Elaborada pela autor.

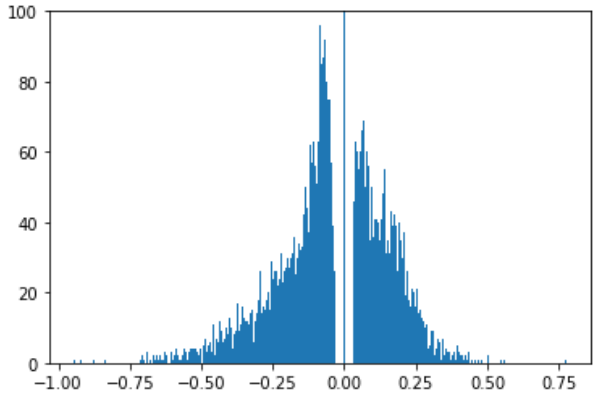


Figura 21 – Histograma de pesos da segunda camada convolucional para $\alpha = 0{,}25$

Fonte – Elaborada pela autor.

No processo de inferência, foi obtida uma acurácia de 73,25%, ou seja, quando comparado ao 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 22.

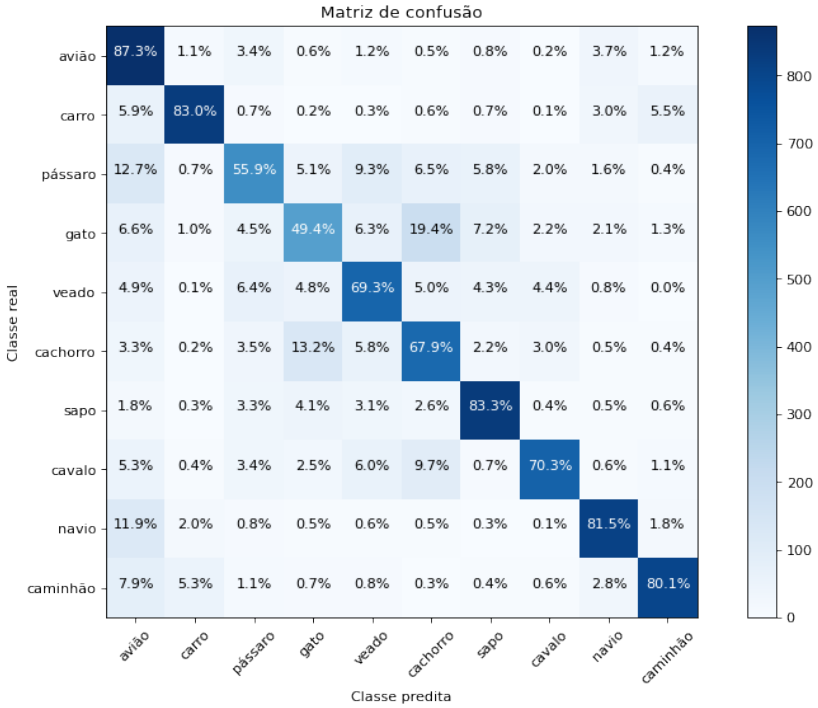


Figura 22 – Matriz de confusão para $\alpha = 0{,}25$

Fonte – Elaborada pela autor.

4.1.3 $\alpha = 0{,}50$

O terceiro experimento consistiu em manter o mesmo processo de treinamento e inferência, mas dessa vez alterando o valor de α para 0,50. A Figura 23 apresenta o valor das acurácias de treino e validação para o modelo nessas configurações e a Figura 24 mostra a variação da esparsidade.A esparsidade do modelo chegou a aproximadamente 68,25%.

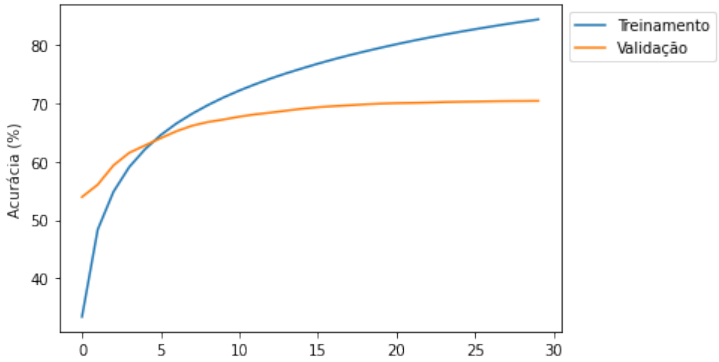


Figura 23 – Acurácia de treinamento e validação para $\alpha = 0{,}50$

Fonte – Elaborada pela autor.

Assim como no experimento anterior, foi visualizado o histograma de pesos da

segunda camada convolucional. Como a quantidade de pesos próximos a zero foi muito alta, foi impossível analisar os pesos com valores mais altos, em módulo, que o limiar. Portanto, a Figura 25 apresenta o histograma de pesos já ampliado.

Com o fim do treinamento, foi feita a inferência no modelo. O resultado foi uma acurácia de 71,90%, mostrando que é possível remover aproximadamente 68,25% dos parâmetros do modelo e obter uma acurácia apenas 3,4% menor que a do modelo não comprimido. A matriz de confusão pode ser visualizada na Figura 26.

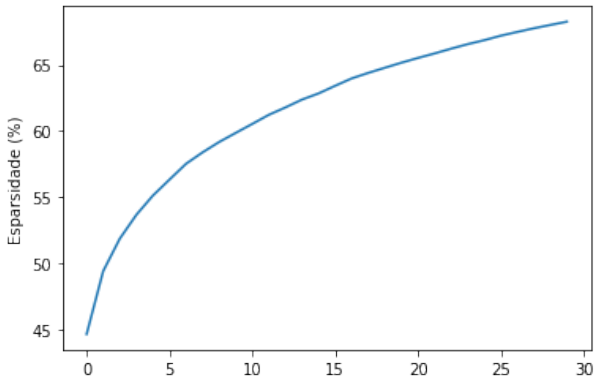


Figura 24 – Variação da esparsidade durante as épocas para $\alpha = 0{,}50$

Fonte – Elaborada pela autor.

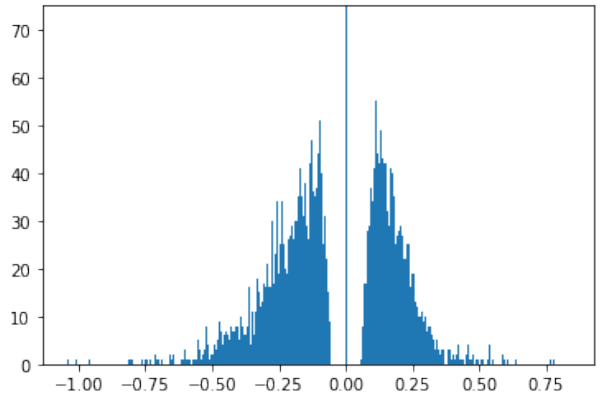


Figura 25 – Histograma de pesos da segunda camada convolucional para $\alpha = 0{,}50$

Fonte – Elaborada pela autor.

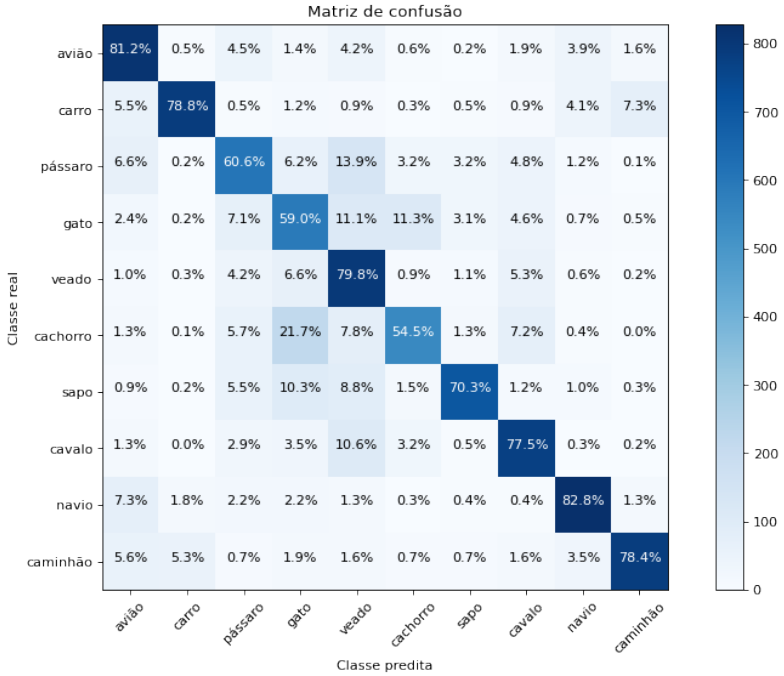


Figura 26 – Matriz de confusão para $\alpha = 0{,}50$

Fonte – Elaborada pela autor.

4.1.4 $\alpha = 0{,}75$

Os próximos resultados são obtidos a partir do mesmo procedimento dos experimentos anteriores, porém com $\alpha = 0{,}75$. As acurácias de treinamento e validação podem ser observadas na Figura 27 e a variação da esparsidade durante o treinamento pode ser visualizada na Figura 28. A esparsidade final do modelo foi de 82,30%. O histograma de pesos pode ser visto na Figura 29.

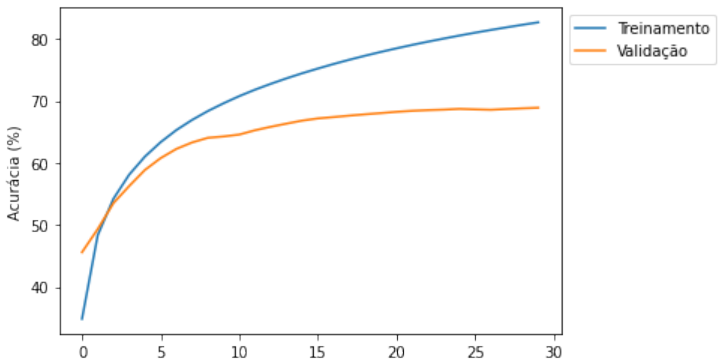


Figura 27 – Acurácia de treinamento e validação para $\alpha = 0{,}75$

Fonte – Elaborada pela autor.

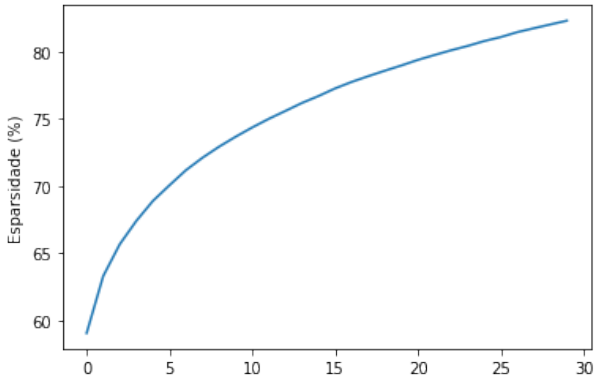


Figura 28 – Variação da esparsidade durante as épocas para $\alpha = 0{,}75$

Fonte – Elaborada pela autor.

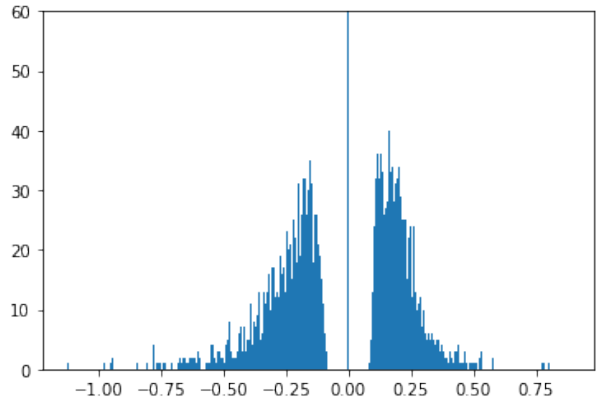


Figura 29 – Histograma de pesos da segunda camada convolucional para $\alpha = 0{,}75$

Fonte – Elaborada pela autor.

Dessa vez, a acurácia de inferência foi de 71,84%, sendo possível então a remoção de 82,30% dos parâmetros do modelo para obteção de uma acurácia de inferência apenas 3,46% menor que a da rede neural não comprimida. A matriz de confusão pode ser visualizada na Figura 30.

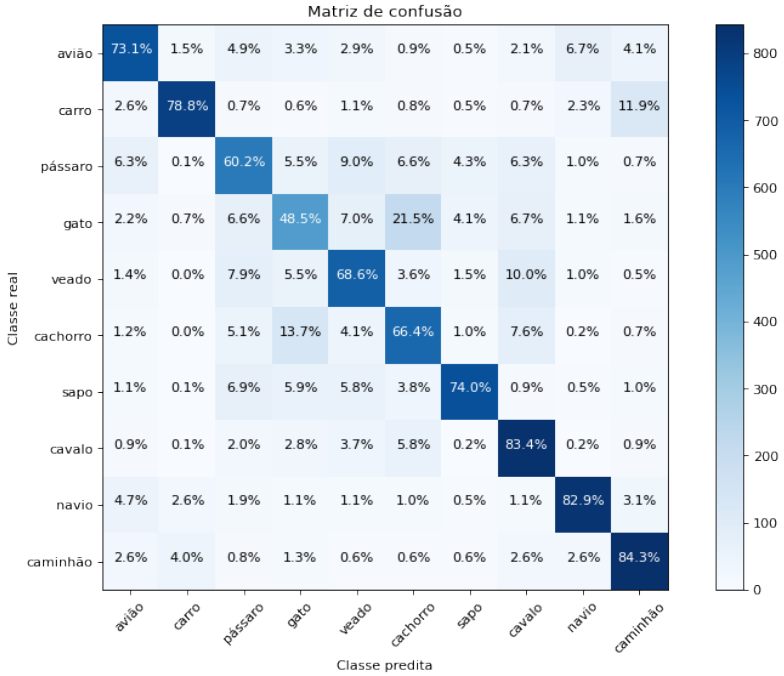


Figura 30 – Matriz de confusão para $\alpha = 0{,}75$

Fonte – Elaborada pela autor.

4.1.5 $\alpha = 1{,}00$

O próximo experimento foi realizado com o α igual a 1,00. Desta vez o limiar será igual ao desvio padrão da camada. É possível vizualizar as acurácias de treinamento e validação na Figura 31 e a variação da esparsidade na Figura 32. Ao final do processo de aprendizagem, a esparsidade foi de 92,44%.

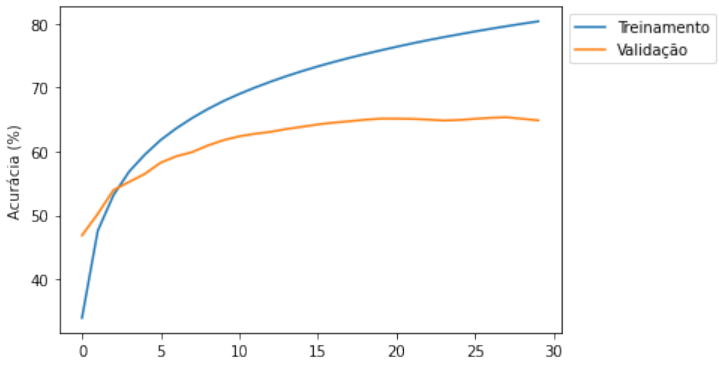


Figura 31 – Acurácia de treinamento e validação para $\alpha = 1{,}00$

Fonte – Elaborada pela autor.

A Figura 33 aprensenta o histograma de pesos após o processo de compressão por poda da segunda camada convolucional. A Figura 34 mostra a matriz de confusão da inferência do modelo.

Para este experimento, a acurácia de inferência obtida foi de 67,13%, sendo então possível obter uma acurácia 7,17% menor que a do modelo não comprimido ao se remover 92,44% dos seus parâmetros.

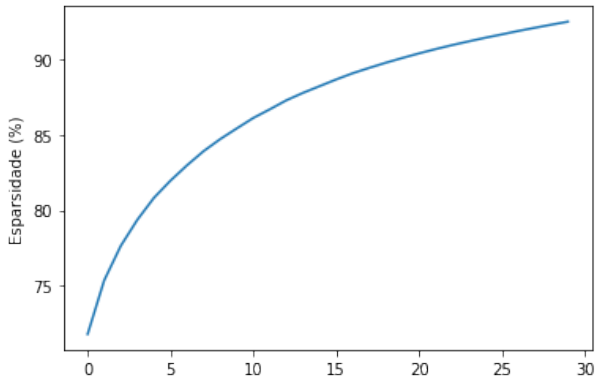


Figura 32 – Variação da esparsidade durante as épocas para $\alpha = 1{,}00$

Fonte – Elaborada pela autor.

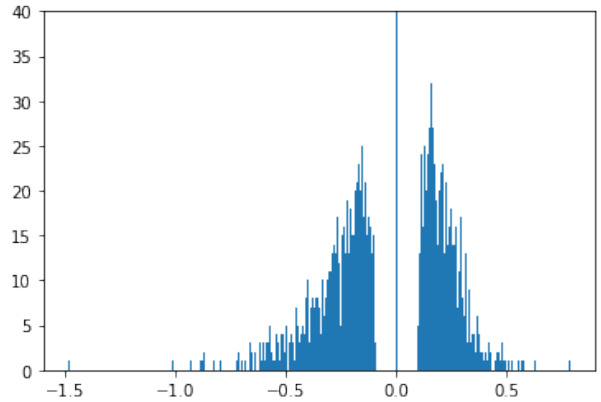


Figura 33 – Histograma de pesos da segunda camada convolucional para $\alpha = 1{,}00$

Fonte – Elaborada pela autor.

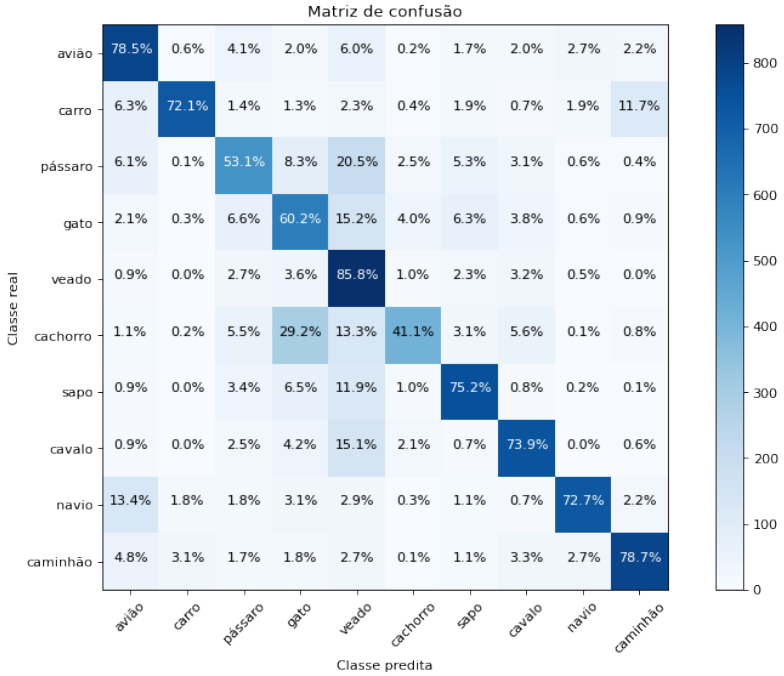


Figura 34 – Matriz de confusão para $\alpha = 1{,}00$

Fonte – Elaborada pela autor.

4.1.6 $\alpha = 1{,}25$

Neste experimento, o valor de α foi modificado para 1,25 resultando em uma maior agressividade na remoção dos pesos. As acurácias de treinamento e validação podem ser visualizadas na Figura 35, enquanto o comportamento da esparsidade no decorrer do treinamento pode ser visto na Figura 36. A esparsidade chegou a 96,01%

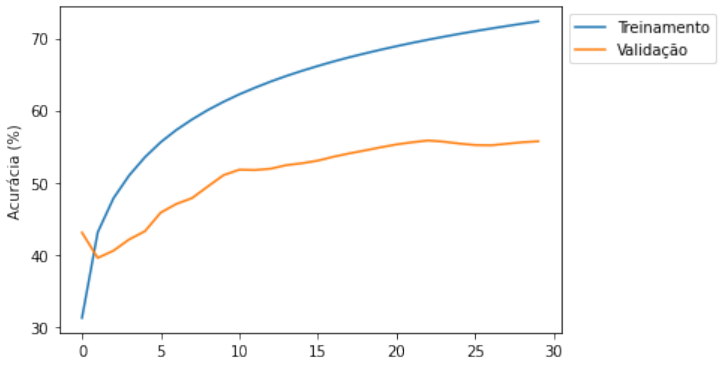


Figura 35 – Acurácia de treinamento e validação para $\alpha = 1{,}25$

Fonte – Elaborada pela autor.

Assim como analisado nos experimentos anteriores, foi analisado o histograma de pesos da segunda camada convolucional da rede neural resultante. Assim como já

mencionado, foi necessário fazer a ampliação do gráfico de forma que seja possível a visualização dos pesos que não sofreram a poda Este histograma se encontra na Figura 37.

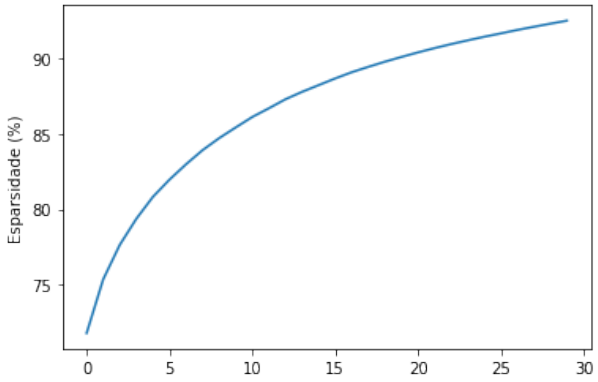


Figura 36 – Variação da esparsidade durante as épocas para $\alpha = 1{,}25$

Fonte – Elaborada pela autor.

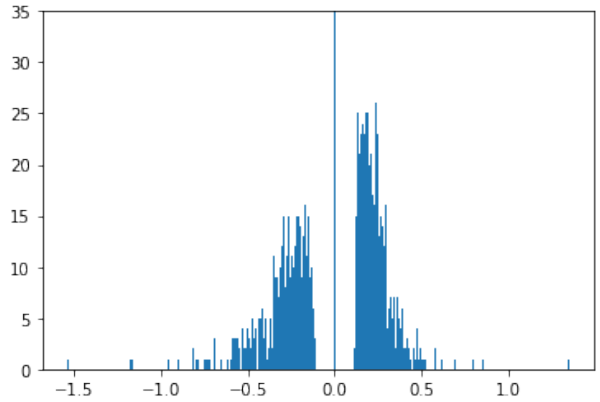


Figura 37 – Histograma de pesos da segunda camada convolucional para $\alpha = 1{,}25$

Fonte – Elaborada pela autor.

A acurácia de validação do modelo foi de 49,55%, mostrando que ao se remover paroximadamente 96,01% dos parâmetros da rede neural em questão, há uma grande interferência na acurácia do modelo, fazendo com que a acurácia de inferência seja 29,75% menor que a do modelo não comprimido.

Para visualização dos resultados obtidos na inferência, a Figura 38 mostra a matriz de confusão para o modelo. É possível perceber que desta vez a acurácia do modelo foi muito afetada quando o processo de poda foi mais agressivo.

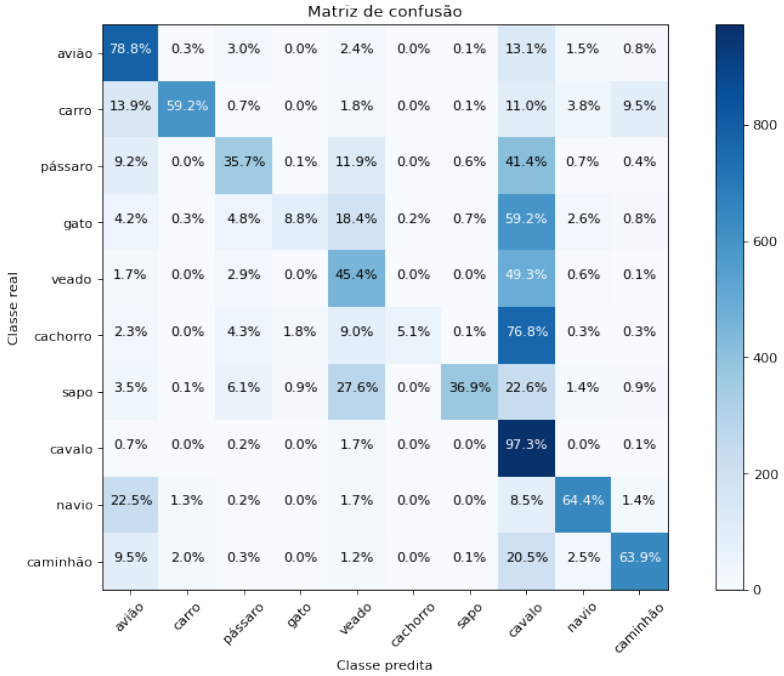


Figura 38 – Matriz de confusão para $\alpha = 1{,}25$

Fonte – Elaborada pela autor.

4.1.7 $\alpha = 1{,}50$

Como último experimento, o valor de α definido como 1,50. Como resultado, obteve-se o modelo com a maior esparsidade dentre os experimentos anteriores. Pode-se vizualizar as curvas das acurácias de treino e validação na Figura 39 e o comportamento esparsidade durante o treinamento na Figura 40. Ao final do processo de aprendizagem, a esparsidade do modelo chegou a 97,82%.

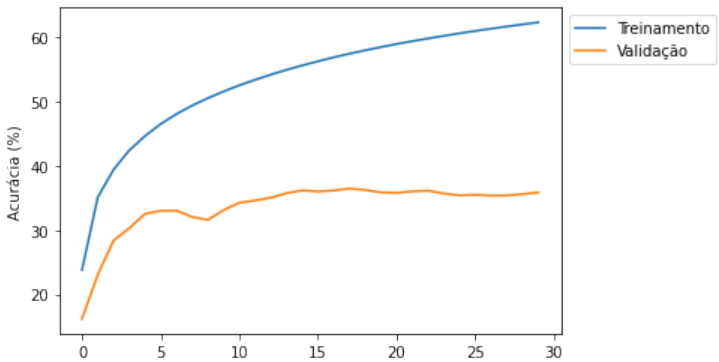


Figura 39 – Acurácia de treinamento e validação para $\alpha = 1{,}50$

Fonte – Elaborada pela autor.

O histograma com os pesos da segunda camada convolucional para este experimento pode ser visualizado na Figura 41.

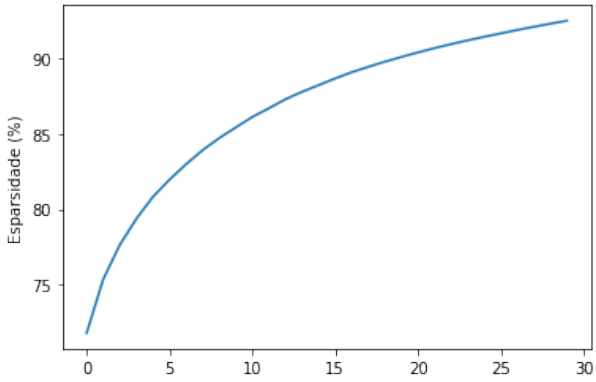


Figura 40 – Variação da esparsidade durante as épocas para $\alpha = 1{,}50$

Fonte – Elaborada pela autor.

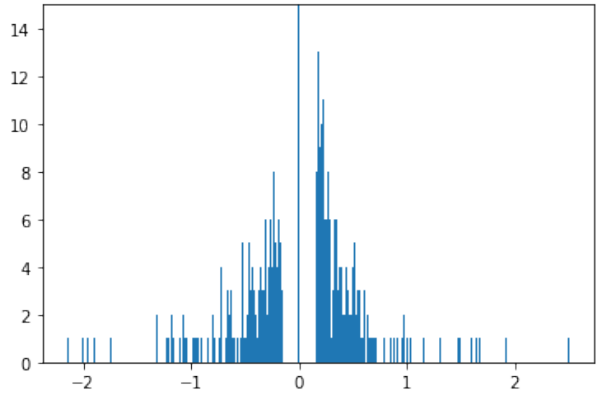


Figura 41 – Histograma de pesos da segunda camada convolucional para $\alpha = 1{,}50$

Fonte – Elaborada pela autor.

Para este experimento, a técnica de compressão por poda afetou muito a acurácia da rede. Com a remoção de 97,82% dos pesos da CNN, a acurácia de validação foi de apenas 44,13%. A Figura 42 mostra a matriz de confusão deste experimento.

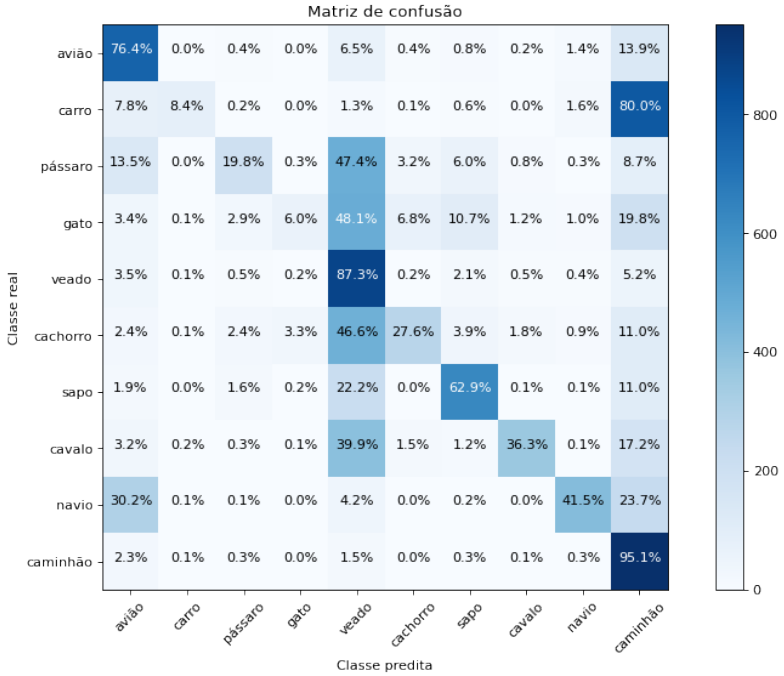


Figura 42 – Matriz de confusão para $\alpha = 1{,}50$

Fonte – Elaborada pela autor.

4.2 Comparações entre resultados

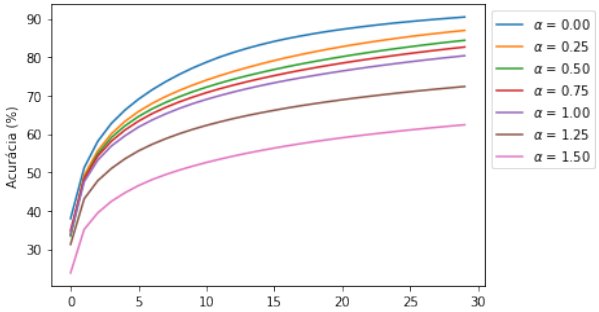
Os resultados de acurácia de treinamento e de validação, para os diferentes valores de α, podem ser visualizados na Figura 43 (a) e (b), respectivamente. Já o comparativo da variação da esparsidade das redes neurais pode ser visualizado na Figura 43 (c). A Tabela 3 mostra os resultados da esparsidade, acurácia de inferência e número total de parâmetros que não foram removidos para cada valor de α.

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

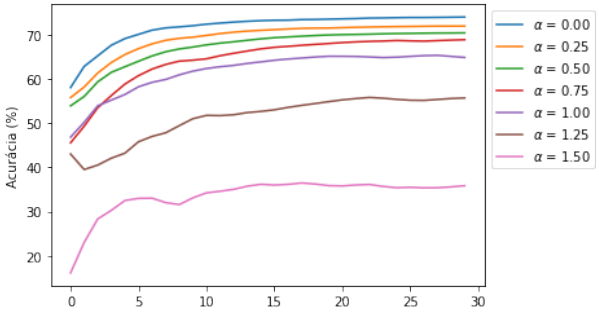
Tabela 3 – Acurácia e esparsidade dos modelos

A Figura 44 mostra o comportamento da acurácia de acordo com a esparsidade do modelo. Pode-se observar que a partir de uma certa esparsidade do modelo, a acurácia cai rapidamente devido o limiar englobar os pesos mais significativos da rede.

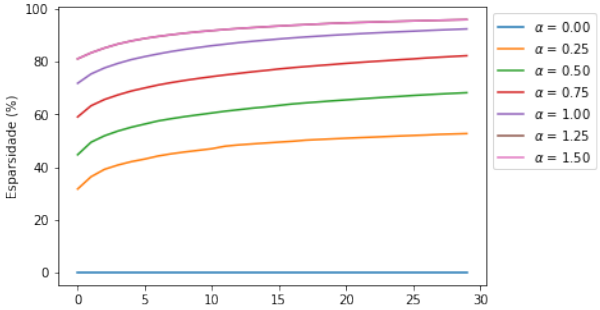
Para o modelo em questão, acurácia se manteve acima dos 70% até a esparsidade de 82,30%. A partir desse valor a acurácia caí rapidamente, mostrando que a aplicação da técnica de poda de forma muito agressiva gera modelos com baixa acurácia.



(a) Acurácias de treinamento



(b) Acurácias de validação



(c) Esparsidade

Figura 43 – Comparação entre resultados dos experimêntos

Fonte – Elaborada pela autor.

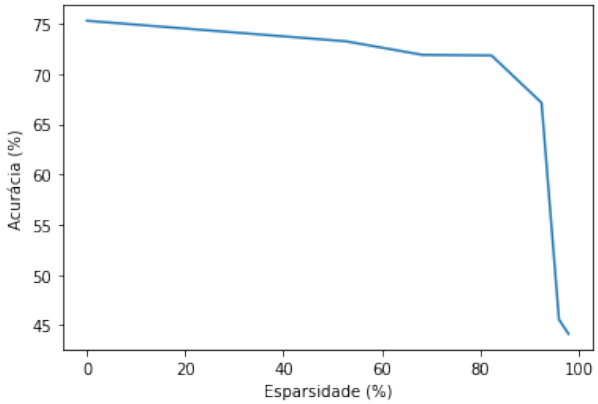


Figura 44 – Comparativo das esparsidades

Fonte – Elaborada pela autor.

5 CONCLUSÃO

Conforme exposto no capítulo 4, podemos notar que:

- O loop de aprendizado, realizando a poda por batch, foi capaz de executar o treinamento e validação como esperado;
- A estratégia de remover os pesos a partir da criação de uma máscara, com o mesmo formato da camada sendo preenchida com zero ou um a depender do limiar estabelecido, multiplicada pela camada em análise demonstrou-se eficaz;
- A partir da modificação da variável α altetar o limiar desejado definindo a agressividade da poda;
- É possível obter uma acurácia com a rede neural comprimida muito próxima da rede não comprimida utilizando a técnica de poda;
- A remoção dos pesos resultou no modelo esparso, sendo que quanto mais pesos são removidos, maior a esparsidade do modelo ao final do treinamento;
- A partir de uma certa esparsidade, a acurácia da rede cai rapidamente, mostrando que existe um certo limite para que o efeito da poda afete a acurácia do modelo minimamente.

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

ANWAR, A. *What is a Convolutional Neural Network?* 2020. Acesso em: 10 abr. 2021. Disponível em: <https://towardsdatascience.com/a-visualization-of-the-basic-elements-of-a-convolutional-neural-network-75fea30cd78d>.

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>.

BROWNLEE, J. *How to Develop a CNN From Scratch for CIFAR-10 Photo Classification*. 2020. Acesso em: 14 jul. 2021. Disponível em: <https://machinelearningmastery.com/how-to-develop-a-cnn-from-scratch-for-cifar-10-photo-classification/>.

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.

GUAN, L. et al. *CNNPruner: Pruning Convolutional Neural Networks with Visual Analytics*. 2020. Acesso em: 13 jun. 2021. Disponível em: <https://arxiv.org/abs/2009-.09940>.

HAN, S.; MAO, H.; DALLY, W. J. *Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding*. 2015. Acesso em: 10 jul. 2021. Disponível em: <https://arxiv.org/abs/1510.00149>.

HUANG, Q. et al. Learning to prune filters in convolutional neural networks. In: IEEE WINTER CONFERENCE ON APPLICATIONS OF COMPUTER VISION (WACV). [S.l.]: IEEE, 2018. p. 709–718.

JIN, S. et al. Deepsz: A novel framework to compress deep neural networks by using error-bounded lossy compression. *Proceedings of the 28th International Symposium on High-Performance Parallel and Distributed Computing*, Association for Computing Machinery, p. 159–170, 2019.

KRIZHEVSKY, A.; NAIR, V.; HINTON, G. *The CIFAR-10 dataset*. 2009. Acesso em: 10 mai. 2021. Disponível em: <https://www.cs.toronto.edu/~kriz/cifar.html>.

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>.

O'SHEA, K.; RYAN, N. *An Introduction to Convolutional Neural Networks*. 2015. Acesso em: 13 abr. 2021. Disponível em: <https://arxiv.org/abs/1511.08458>.

REED, R. Pruning algorithms - a survey. *IEEE Transactions on Neural Networks*, IEEE, v. 4, n. 5, p. 740–747, 1993.

SAHA, S. *A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way*. 2018. Acesso em: 11 abr. 2021. Disponível em: <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>.

STUART, J. R.; NORVING, P. Artificial intelligence. In: _____. *Artificial Intelligence: A Modern Approach*. 1. ed. New Jersey: Alan Apt, 2013. cap. 1, p. 4–16.

THE GOOGLE BRAIN TEAM. *Writing a training loop from scratch*. 2021. Acesso em: 12 jun. 2021. Disponível em: <https://www.tensorflow.org/guide/keras/writing_a_training_loop_from_scratch>.

VERSLOOT, C. *TensorFlow pruning schedules: ConstantSparsity and PolynomialDecay*. 2020. Acesso em: 15 jul. 2021. Disponível em: <https://www.machinecurve.com/index.php/2020/09/29/tensorflow-pruning-schedules-constantsparsity-and-polynomialdecay/>.

YANG, T.; CHEN, Y.; VIVIENNE, S. *Designing Energy-Efficient Convolutional Neural Networks using Energy-Aware Pruning*. 2017. Acesso em: 9 jul. 2021. Disponível em: <https://arxiv.org/abs/1611.05128>.