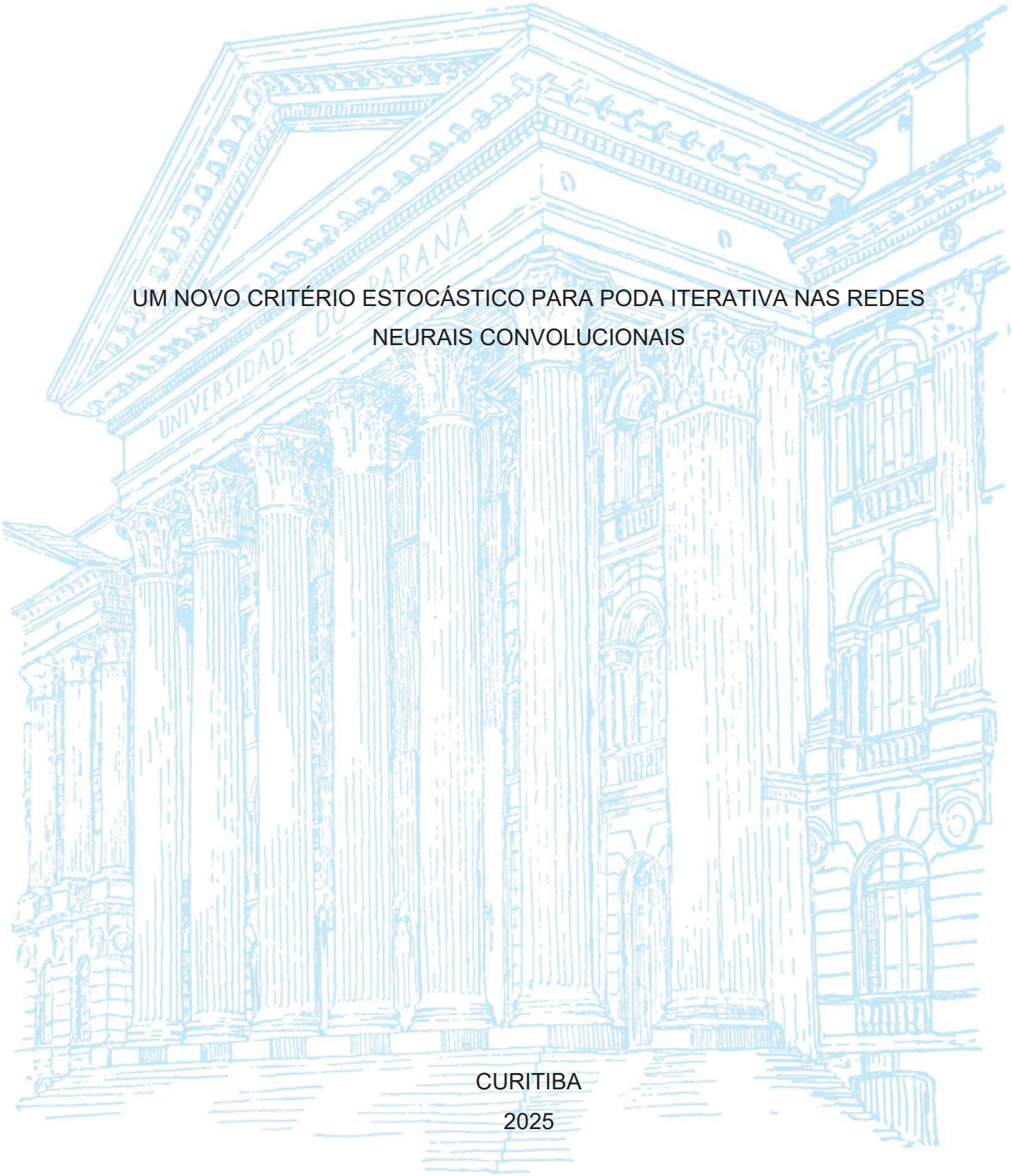
UNIVERSIDADE FEDERAL DO PARANÁ

LUCAS LAMY



UM NOVO CRITÉRIO ESTOCÁSTICO PARA PODA ITERATIVA NAS REDES NEURAIS CONVOLUCIONAIS

CURITIBA

2025

LUCAS LAMY

UM NOVO CRITÉRIO ESTOCÁSTICO PARA PODA ITERATIVA NAS REDES NEURAIS CONVOLUCIONAIS

Tese apresentada ao Programa de Pós-Graduação em Métodos Numéricos em Engenharia da área de concentração de Programação Matemática, dos Setores de Ciências Exatas e Tecnologia da Universidade Federal do Paraná, como um dos requisitos parciais para obtenção do título de Doutor em Métodos Numéricos em Engenharia.

Orientador: Prof. Dr. Paulo Henrique Siqueira.

CURITIBA

2025

DADOS INTERNACIONAIS DE CATALOGAÇÃO NA PUBLICAÇÃO (CIP)
UNIVERSIDADE FEDERAL DO PARANÁ
SISTEMA DE BIBLIOTECAS – BIBLIOTECA DE CIÊNCIA E TECNOLOGIA

Lamy, Lucas
Um novo critério estocástico para poda iterativa nas redes neurais convolucionais / Lucas Lamy. – Curitiba, 2025.
1 recurso on-line : PDF.

Tese (Doutorado) - Universidade Federal do Paraná, Setor de Ciências Exatas, Programa de Pós-Graduação em Métodos Numéricos em Engenharia.

Orientador: Paulo Henrique Siqueira

1. Redes neurais (Computação). 2. Convoluções (Matemática). 3. Imagem – Classificação. 4. Processamento de imagens. 5. Visão computacional. I. Universidade Federal do Paraná. II. Programa de Pós-Graduação em Métodos Numéricos em Engenharia. III. Siqueira, Paulo Henrique. IV. Título.

Bibliotecário: Elias Barbosa da Silva CRB-9/1894

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação MÉTODOS NUMÉRICOS EM ENGENHARIA da Universidade Federal do Paraná foram convocados para realizar a arguição da tese de Doutorado de **LUCAS LAMY**, intitulada: **UM NOVO CRITÉRIO ESTOCÁSTICO PARA PODA ITERATIVA NAS REDES NEURAIS CONVOLUCIONAIS**, sob orientação do Prof. Dr. PAULO HENRIQUE SIQUEIRA, que após terem inquirido o aluno e realizada a avaliação do trabalho, são de parecer pela sua APROVAÇÃO no rito de defesa.

A outorga do título de doutor está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

Curitiba, 25 de Abril de 2025.

Assinatura Eletrônica
28/04/2025 12:51:56.0
PAULO HENRIQUE SIQUEIRA
Presidente da Banca Examinadora

Assinatura Eletrônica
28/04/2025 12:22:07.0
MYRIAM REGATTIERI DE BIASE DA SILVA DELGADO
Avaliador Externo (UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ)

Assinatura Eletrônica
28/04/2025 13:24:43.0
LEANDRO MAGATÃO
Avaliador Externo (UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ)

Assinatura Eletrônica
28/04/2025 12:11:32.0
LUCAS GARCIA PEDROSO
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

Documento assinado eletronicamente de acordo com o disposto na legislação federal Decreto 8539 de 08 de outubro de 2015.
Gerado e autenticado pelo SIGA-UFPR, com a seguinte identificação única: 445849
Para autenticar este documento/assinatura, acesse https://siga.ufpr.br/siga/visitante/autenticacaoassinaturas.jsp e insira o codigo 445849

“Em tempos, os homens entregavam o pensamento às máquinas, na esperança de que isso os libertasse. Mas só permitiu que outros homens com máquinas os escravizassem.” (HERBERT, Frank. Duna,1965.)

RESUMO

As técnicas de poda aplicadas nas redes neurais artificiais são divergentes em relação à tendência dos últimos anos. Enquanto as arquiteturas se tornam cada vez maiores, complexas, com mais parâmetros e camadas, os métodos de poda buscam reduzir a quantidade de parâmetros e a complexidade das redes. A hipótese do bilhete premiado é um algoritmo de poda iterativa para achar redes super esparsas. Esta técnica mostrou que com os mesmos pesos é possível chegar a uma rede de desempenho igual ou superior à rede não podada. Dessa forma, a pergunta natural é: existem outros critérios de poda para se achar o bilhete premiado? Neste trabalho é apresentado um novo critério de poda, denominado de Poda Estocástica com Camada Nula. Utilizando quatro conjuntos de dados e sete arquiteturas da literatura foi analisado o comportamento do método proposto. A análise foi feita por meio do desempenho, níveis de esparsidade e contagem das operações de pontos flutuantes ao longo das rodadas de poda. O algoritmo de Poda Estocástica com Camada Nula conseguiu melhorar a acurácia das redes testadas e apresentou o melhor desempenho quando estavam em seus maiores níveis de esparsidade.

Palavras-chave: Poda iterativa; Redes Neurais Convolucionais; Classificação de imagens; Visão computacional.

ABSTRACT

Pruning techniques applied to artificial neural networks are divergent from the trend observed in recent years. While architectures are becoming larger, more complex, with more parameters and layers, pruning methods reduce the number of parameters and the complexity of networks. The lottery ticket hypothesis is an iterative pruning algorithm to find highly sparse networks. It demonstrated that it is possible to achieve a network with equal or superior performance to the unpruned network with the same initial weights. Thus, the natural question is: are there other pruning criteria to find the winning ticket? In this work, a new pruning criterion is presented, called Stochastic Pruning with Null Layer. The proposed method was analyzed using four datasets and seven architectures from the literature. The analysis was conducted through performance, sparsity levels, and Floating-Point Operations throughout the pruning rounds. The Stochastic Pruning with Null Layer algorithm improved the accuracy of the tested networks and achieved the best performance when it was at its highest levels of sparsity.

Keywords: Iterative pruning; Convolutional neural network; Image Classification; Computational Vision.

LISTA DE FIGURAS

LISTA DE GRÁFICOS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ABREVIATURAS OU SIGLAS

AM	- Aprendizagem de Máquina
AP	- Aprendizagem Profunda
CIFAR-10	- *Canadian Institute for Advanced Research*
CN	- Camada Nula
CNN	- *Convolution Neural Network*
CPU	- *Central Process Unit*
FA	- Função Ativação
FLOP	- *Floating Points Operation*
GPU	- *Graphics Processing Unit*
IA	- Inteligência Artificial
ILSVRC	- *ImageNet Large Scale Visual Recognition Challenge*
LLM	- *Large Language Model*
Lt	- *Lottery Ticket*
Lth	- *Lottery Ticket Hypothesis*
MNIST	- *Modified National Institute of Standards and Technology*
PE	- Poda Estocástica
PECN	- Poda Estocástica com Camada Nula
RNA	- Rede Neural Artificial
RNN	- *Recurrent Neural Network*
RNP	- Rede Neural Profunda
SGD	- *Stochastic Gradient Descent*
Tanh	- Tangente Hiperbólica
TC	- Totalmente Conectada
TPU	- *Tensor Processing Unit*
VC	- Visão Computacional

LISTA DE SÍMBOLOS

$*$ - Operação de convolução

C_p - Critério de poda

$g_a(\cdot)$ - Função ativação

β_1 - Momento

β_2 - Momento

θ_0 - Pesos iniciais da rede de convolução

θ_{aux} - Pesos da rede de convolução amortecidos

θ_n - Pesos da rede de convolução gerados pela distribuição normal

σ^2 - Variância

$\mathcal{B}$ - Batelada

f - Tamanho do filtro

$\mathcal{R}$ - *Receptive field*

B - Viés

C - Número de classes

D - Conjunto de dados

DF_c - Dimensão final da operação de convolução

DF_p - Dimensão final da operação de *pooling*

E - Tamanho da imagem de entrada

F - Filtro

G - Resultado da função ativação

I - Imagem de entrada

L - Função Perda

N - Distribuição Normal

P - Quantidade de z*ero-padding*

S - Tamanho do passo

U - Distribuição uniforme

Y - Mapas de característica

Z - Resultado da camada totalmente conectada

l - Número de camadas

m - Máscara binária

mc - Elemento do mapa de característica

n - Número de elementos no conjunto de dados

p - Taxa de poda

x - Imagem de entrada de um conjunto de dados

y - Resposta de uma imagem de um conjunto de dados

$\mathcal{D}$ - Distribuição

$\mathcal{N}$ - Rede neural

$\mathcal{N}'$ - Bilhete Premiado

α - Taxa de amortecimento

γ - Taxa de aprendizagem

η - Limiar de poda

θ - Pesos da rede

μ - Média

$\sigma(\cdot)_c$ - *Score* da função *Softmax*

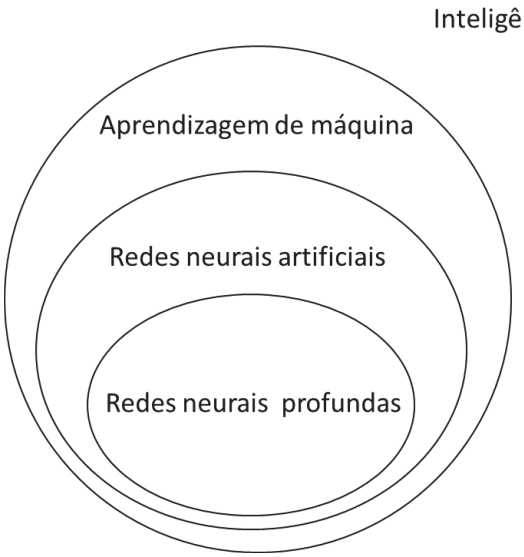
ϵ - Constante próxima de zero

SUMÁRIO

1 INTRODUÇÃO

A partir dos anos 1950, quando os computadores começaram a ter mais poder computacional, técnicas complexas - que iam além de simples cálculos aritméticos ou instruções diretas - começaram a ser desenvolvidas. Dentre elas, surgiram os algoritmos de inteligência artificial (IA). A IA surgiu na mistura de várias tarefas, tais como: provas de teoremas matemáticos, planejamento para jogos e programas que poderiam aprender por exemplos (Kelleher, 2019). Nesse contexto, Poole e Mackworth (2010) definem a IA como o campo que estuda agentes computacionais que operam de forma inteligente. Com o tempo, a IA passou a dividir a sua atuação de forma específica, como ilustrado na FIGURA 1.

FIGURA 1 – TAXONOMIA DA INTELIGENCIA ARTIFICIAL



FONTE: O autor (2025).

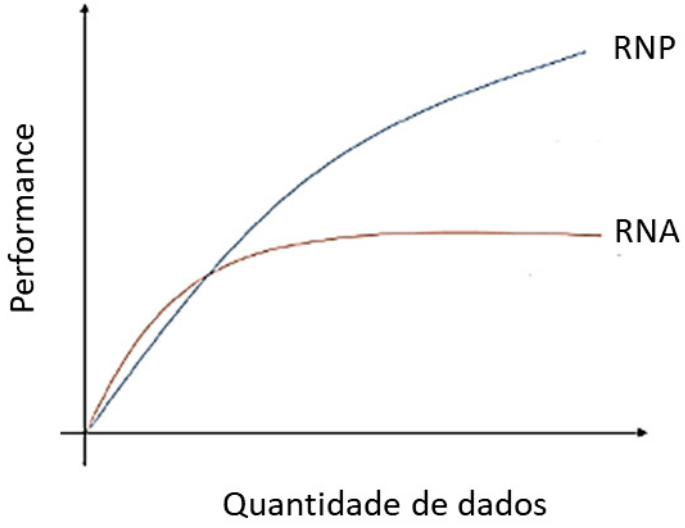
Um dos subcampos da IA é denominado aprendizagem de máquina (AM). Esse subcampo está relacionado com algoritmos que extraem informações de conjuntos de dados e aprendem com eles. Esta técnica tem revolucionado vários campos científicos pois possui uma ampla aplicabilidade, tais como: identificação de objetos (Assis *et al.*, 2024), acessibilidade (ZainEldin *et al*., 2024), análise de crédito (Chang *et al*., 2024) entre outras.

Dentro da AM existe a abordagem inspirada no funcionamento dos neurônios humanos, as Redes Neurais Artificiais (RNAs). As RNAs são compostas por camadas

e neurônios (chamados também de pesos ou parâmetros) de forma hierárquica, isto é, as informações são passadas sequencialmente entre as camadas. Nesse processo, onde cada neurônio recebe e processa os sinais dos neurônios anteriores, é que a RNA aprende com os exemplos. Dessa forma, as RNAs possuem o aprendizado baseado nos dados de entrada com regras genéricas de aprendizagem que não são acondicionadas manualmente por humanos (Lecun *et al.*, 2015). A arquitetura, ou seja, a quantidade de camadas e neurônios pode variar de acordo com o problema. Porém, enquanto nas RNAs normalmente se usam arquiteturas com poucas camadas, existem outras redes mais complexas e com muitas camadas que se encontram no campo da aprendizagem profunda (AP). Essas redes também são conhecidas como redes neurais profundas (RNPs).

As RNPs ganharam destaque nos últimos anos (Shetty *et al*., 2020). Um dos motivos é o bom desempenho que proporcionam em comparação com as suas antecessoras, as RNAs, como apresentado na FIGURA 2.

FIGURA 2 – DESEMPENHO DA RNP VS. RNA



FONTE: Adaptado de Alom *et al*. (2018).

Dessa forma, é vantajoso o uso das RNAs quando a quantidade de dados é menor, pois, à medida que a quantidade de dados aumenta, o desempenho fica estagnado. De forma contrária, nas RNP o desempenho é proporcional à quantidade de dados que a rede possui (Najafabadi *et al*., 2015; Xue-Wen; Xiaotong, 2014; Zhou

et al., 2014). Do ponto de vista qualitativo, a abordagem da RNP aproveita a época em que vivemos, caracterizada pela utilização da *big data.*

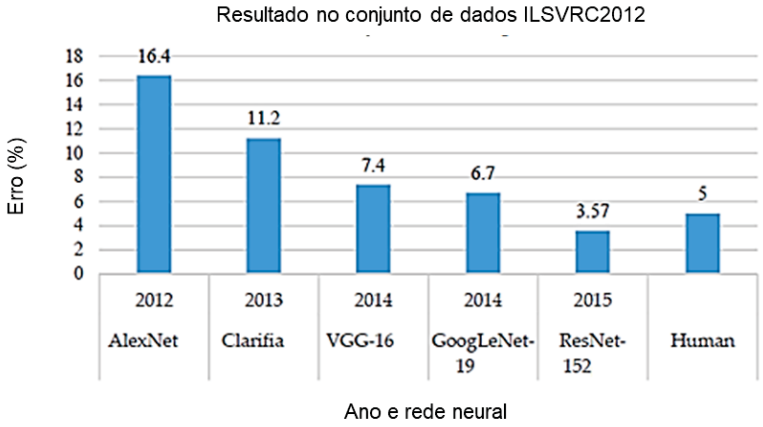
Existem diferentes tipos de redes neurais dentro das RNPs, como: as redes neurais convolucionais (CNN, do inglês, *Convolutional Neural Network*), as Redes Neurais Recorrentes (RNN, do inglês, *Recurrent Neural Network*) e as *Large Language Models* (LLMs).

Em especial, para a visão computacional (VC), a rede neural mais utilizada é a CNN, onde é aplicada em diversas tarefas, tais como: localização e detecção de objetos, segmentação de objetos e classificação de imagens. A tarefa fundamental para a VC é a classificação de imagens, pois essa é a base para todas as outras. Dessa forma, Li (2020) define classificação de imagem como a tarefa de categorizar imagens em uma das classes pré-definidas.

Um marco para as RNPs e para a VC ocorreu em 2012, na competição de algoritmos que avalia o reconhecimento e classificação de objetos em imagens de alta resolução e em larga escala, o *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC). Krizhevsky *et al.* (2012) conseguiram uma taxa de erro abaixo de 17% na tarefa de classificação de imagem no conjunto de dados ImageNet (Jia Deng *et al.*, 2009), o qual possui mais de 15 milhões de imagens em alta resolução com 22 mil classes diferentes.

Desde então, as RNPs têm evoluído anualmente, superando até os próprios humanos na tarefa de reconhecimento de imagens, como mostra a FIGURA 3.

FIGURA 3 – PORCENTAGEM DE ERRO NO ILSVRC-2012



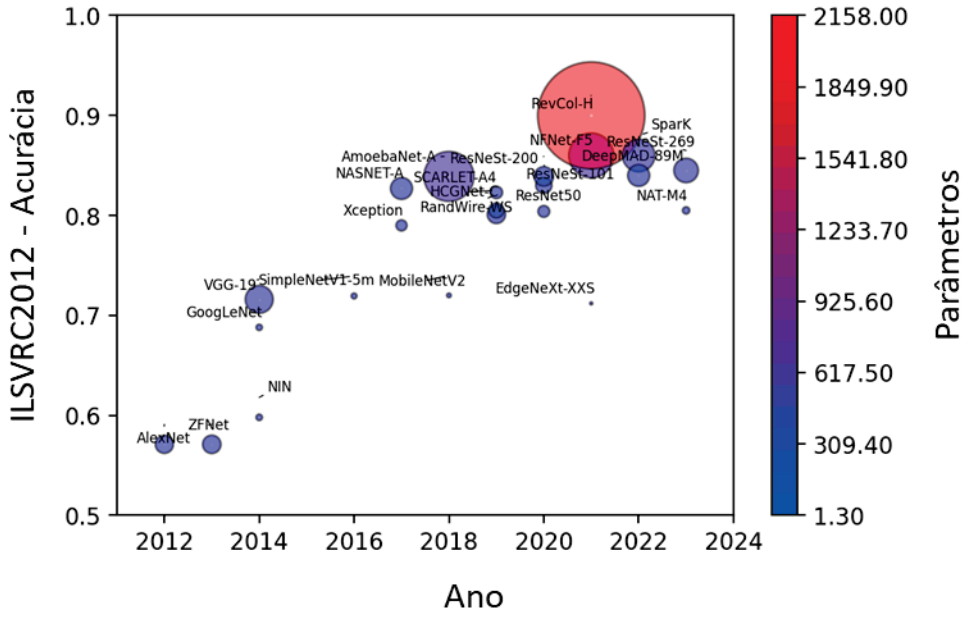
FONTE: Adaptado de Alom *et al*. (2019).

LEGENDA: AlexNet (Krizhevsky *et al.,* 2012), Clarifia (Zeiler; Fergus, 2013), VGG-16 (Simonyan; Zisserman, 2014), GoogleNet-19 (Szegedy *et al*., 2015), ResNet-152 (He *et al*., 2015a).

Além de classificação de imagem, as RNPs têm apresentado bons resultados em outros campos de conhecimento, como: agricultura (Huynh; Nguyen, 2024), procedimentos médicos (Sharma et al., 2024), detecção de atividades maliciosas em computação nas nuvens (Bhingarkar et al., 2023), *chatbot* (Du et al., 2024) e outras.

Com o passar dos anos, essas arquiteturas ficaram cada vez maiores (FIGURA 4); consequentemente, passaram a possuir muitos parâmetros, necessitando de um grande esforço computacional para a sua execução (Almeida *et al.*, 2019).

FIGURA 4 – EVOLUÇÃO DA QUANTIDADE DE PARÂMETROS NAS RNPS



FONTE: O autor (2025).

LEGENDA: AlexNet (Krizhevsky *et al.*, 2012), NIN (Lin *et al.*, 2014), ZFNet (Zeiler; Fergus, 2013), VGG19 (Simonyan; Zisserman, 2014), GoogLeNet (Szegedy *et al.*, 2015), Xception (Chollet; Inc. Google, 2017), NASNET-A (Zoph *et al.*, 2018), MobileNetV2 (Sandler *et al.*, 2018), AmoebaNet-A (Real *et al.*, 2018), SCARLET-A4 (Chu *et al.*, 2019), RandWire-WS (Xie *et al.*, 2019), ResNeSt-101 (Zhang *et al.*, 2020), ResNeSt-200 (Zhang *et al.*, 2020), ResNet-50 (Wightman *et al.*, 2021), NFNet-F5 (Brock *et al.*, 2021), RevCol-H (Cai *et al.*, 2022), EdgeNeXt-XXS (Maaz *et al.*, 2022), SparK (Tian *et al.*, 2023), DeepMAD-89M (Shen *et al.*, 2023), ResNeSt-269 (Zhang *et al.*, 2020), NAT-M4 (Lu *et al.*, 2020), HCGNet-C (Yang *et al.*, 2019), SimpleNetV1-5m (Hasanpour *et al.*, 2016).

Logo, de forma natural, surge a demanda de tornar as RNPs mais eficientes; pois, por mais que os hardwares tenham avançado ao longo dos anos, tais avanços

não têm sido suficientes para suprir a crescente demanda computacional imposta por modelos cada vez mais complexos.

Existem vários métodos para otimizá-las, como: variar os tipos de camadas, fazer uma busca pelos melhores hiperparâmetros, alterar o algoritmo de aprendizado ou até desenvolver e utilizar hardware de alto desempenho. No entanto, no contexto de uma rede profunda que possui muitas camadas, existe a abordagem de escolher quais parâmetros são úteis. Ao eliminar parâmetros que não são úteis para a rede é possível acelerar o treinamento e sua velocidade de inferência. Esse tipo de técnica é chamado de poda, que faz parte das técnicas de compressão de rede.

As técnicas de poda, como otimizadores das RNPs, buscam alcançar maior ou igual desempenho de uma rede não podada, mas utilizando apenas uma porção reduzida dos pesos existentes. O desenvolvimento dessas técnicas é motivado pela grande quantidade de parâmetros treináveis das RNPs que nem sempre são úteis, ou seja, pelo problema da super parametrização (*overparametrization)*.

Além disso, a poda dos parâmetros reduz as operações de ponto flutuante (FLOPs, do inglês, *Floating-point Operations*), o que diminui o custo computacional melhorando a eficiência, reduzindo a memória necessária, o tempo de execução, a latência de comunicação e detecção em dispositivos embarcados/móveis. Essas melhorias tornam a técnica de poda essencial quando existe hardware limitados, como em sistemas embarcados, onde os recursos computacionais e energéticos são restritos. Ainda, ao diminuir a complexidade dos modelos, a técnica contribui para a redução do uso de infraestrutura de grandes *data centers*, reduzindo o consumo de energia elétrica e favorecendo operações mais sustentáveis.

Nesse contexto, diversas estratégias de poda têm sido propostas na literatura, entre as quais se destaca o algoritmo iterativo proposto por Frankle e Carbin (2019), chamado de hipótese do bilhete premiado (Lt, do inglês, *Lottery Ticket Hypothesis*). Onde, por meio de repetidos processos de poda e retreinamento é possível achar dentro da rede original uma sub-rede esparsa que alcança, no mínimo, a mesma acurácia que a rede original com os mesmos pesos iniciais.

Com base nesse trabalho, esta tese apresenta um novo algoritmo de poda iterativa, ou seja, um novo critério de seleção dos pesos para a geração de redes de alta esparsidade.

1.1 JUSTIFICATIVA

Como mencionado anteriormente, as RNPs possuem muitas camadas e milhões de parâmetros. Essa quantidade excessiva de parâmetros não reflete necessariamente na performance da rede, mas implica em complexidade, esforço e maior demanda computacional para o processamento. Dessa forma, a busca por uma rede otimizada, neste caso, redes esparsas que mantêm a performance, é essencial para o aumento de desempenho da rede neural. Essas redes esparsas reduzem o número de operações necessárias e possibilitam as redes neurais em hardwares limitados.

1.2 OBJETIVOS

1.2.1 Objetivo geral

Desenvolver um algoritmo de poda iterativa para as redes neurais profundas, competitivo quando comparado aos algoritmos existentes da literatura.

1.2.2 Objetivos específicos

Os objetivos específicos são:

- Compreender e aprimorar as técnicas de poda, utilizando como base a hipótese do bilhete premiado;
- Classificar as técnicas pelos seus mecanismos de funcionamento;
- Desenvolver um critério de seleção dos pesos para a aplicação da técnica de poda iterativa, de forma a obter uma RNP esparsa que permita atingir uma taxa de erro igual ou inferior à da rede não podada;
- Comparar o critério desenvolvido com os existentes da literatura.

1.3 LIMITAÇÕES DO TRABALHO

A principal limitação do trabalho é a viabilidade de experimentação da técnica desenvolvida, ou seja, a sua aplicabilidade às RNPs já estabelecidas na literatura. Como o objeto de estudo são as RNPs (algumas com mais de 100 camadas) e a

técnica de poda iterativa exige sucessivas rodadas de treinamentos, a utilização de um computador doméstico é limitada devido ao tempo exigido e à necessidade de memória computacional tanto para a execução e armazenamento dos parâmetros das redes quanto para os conjuntos de dados.

1.4 INOVAÇÃO PROPOSTA

A inovação desta tese reside no desenvolvimento de um novo critério estocástico de poda para as CNNs, o critério de Poda Estocástica com Camada Nula, fundamentado na comparação entre valores gerados pela distribuição normal e os pesos da rede. Com esse novo critério de identificação de parâmetros não relevantes para a rede, são proporcionadas melhorias na acurácia, esparsidade e operação de pontos flutuantes ao estado da arte.

1.5 ESTRUTURA DA TESE

Além desta Introdução, esta tese apresenta uma fundamentação teórica no Capítulo 2, composta pela literatura sobre as técnicas de poda e pela descrição da arquitetura das CNNs. No Capítulo 3, denominado de Material e Métodos, são descritos os tópicos necessários para replicabilidade do método proposto, como: as arquiteturas utilizadas, os conjuntos de dados e algoritmos de aprendizagem. A explicação do critério de poda desenvolvido é apresentada no Capítulo 4. A análise dos resultados e a sua discussão é feita no Capítulo 5. No último capítulo, a denominado Conclusões, é apresentado o fechamento da pesquisa de tese.

2 REVISÃO DE LITERATURA

Este capítulo tem por finalidade apresentar os principais fundamentos teóricos que sustentam a proposta desta tese. Inicialmente, são discutidos a evolução histórica e os aspectos estruturais das CNNs, com a descrição das funções e características das camadas que as compõem. Em seguida, abordam-se os métodos de inicialização dos pesos. Nesse capítulo, também são descritas as técnicas de poda, com uma classificação baseada em suas propriedades. Por fim, são apesentados os algoritmos de treinamento e as funções perda, finalizando-se com considerações que integram os tópicos abordados.

2.1 REDES NEURAIS CONVOLUCIONAIS

2.1.1 Evolução histórica das redes neurais convolucionais

Para o desenvolvimento das primeiras CNNs, as ideias elementares vieram dos trabalhos de Hubel e Wiesel (1959, 1962, 1968) que pesquisavam a estrutura do córtex visual. Esses trabalhos mostraram que o córtex visual é formado por células simples, complexas e muito complexas. Essas células são especializadas para a tarefa de reconhecimento e se arranjam em camadas de forma hierárquica. Os neurônios das camadas iniciais respondem a padrões simples como luzes orientadas (barras e linhas), porém, ignoram estruturas complexas; de forma contrária, os neurônios de camadas mais profundas respondem às estruturas mais complexas e não respondem às formas mais simples das camadas iniciais (Rawat; Wang, 2017).

Foi somente em 1979 a primeira CNN, chamada Neocognitron (Fukushima, 1979, 1980, 1988), foi publicada. Apesar de não existirem computadores com capacidade de processamento capazes de utilizá-la na época, já era possível o reconhecimento de padrões de imagens como algarismos e letras.

Após 10 anos, LeCun (1989) propôs a primeira CNN multicamada supervisionada, chamada de ConvNet, que conseguiu com sucesso estabelecer base para a construção das futuras redes utilizadas para reconhecimento de imagem. A sua sucessora, LeNet-5 (Lecun *et al*., 1998), mostrou a capacidade das CNNs, ela foi utilizada para reconhecimento de dígitos e posteriormente utilizada para ler milhares de cheques por dia.

Um dos grandes avanços para as CNNs foi a utilização das placas de vídeo (GPU, do inglês, G*raphics Processing Units*) para o processamento das redes. Enquanto as CPUs (do inglês, *Central Processing Unit*) usam de forma sequencial seus núcleos de processamento, as GPUs possuem centenas de pequenos núcleos especializados e operam em paralelo. Dessa forma, as GPUs tornaram o treinamento de uma rede muito mais veloz, viabilizando a utilização das RNPs.

Posteriormente, a rede chamada AlexNet proposta por Krizhevsky *et al*. (2012) colocou as redes de convolução em outro patamar. Pela primeira vez, uma rede neural conseguiu alcançar uma taxa de erro abaixo de 17% em um conjunto de dados de alta resolução. Para alcançar tal sucesso, a AlexNet trouxe algumas inovações, como a implementação da Função Ativação (FA) ReLU (*Rectified Linear Unit*), o uso de duas GPUs em paralelo, as técnicas de modificação de imagem (*data augmentation*) e o desenvolvimento da técnica *Dropout*.

Após a rede AlexNet, diversas inovações foram realizadas nas CNNs. As principais redes foram resumidas na linha do tempo da FIGURA 5.

FIGURA 5 – EVOLUÇÃO HISTÓRICA DAS REDES DE CONVOLUÇÃO

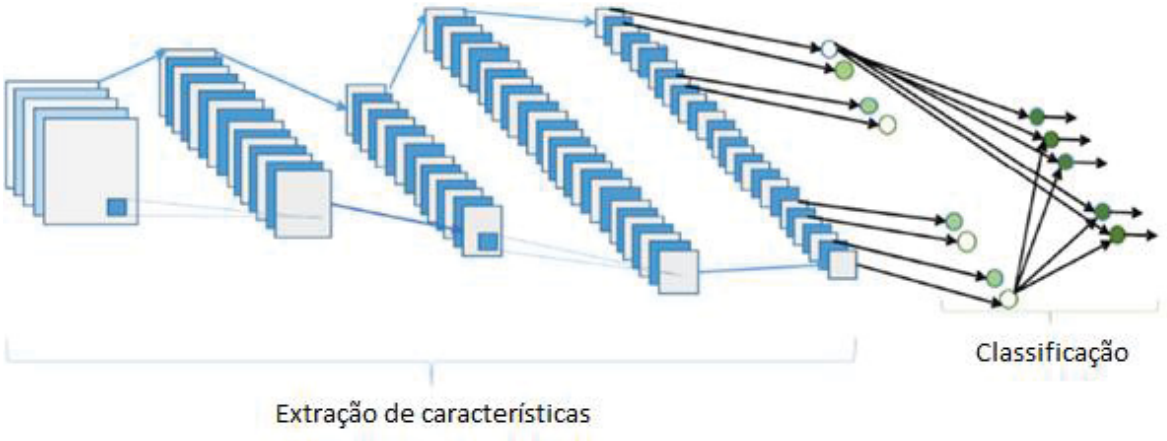
FONTE: Adaptado de Ferreira (2020).

2.1.2 Arquitetura das redes neurais convolucionais

A CNN emprega a operação matemática de convolução, isto é, utiliza-se a operação de multiplicação elemento a elemento (*element-wise multiplication*), em pelo menos uma de suas camadas – a camada de convolução (Goodfellow *et al*., 2016).

A arquitetura da CNN é composta por camadas sequenciais, onde cada camada recebe o resultado da camada anterior, como mostra a FIGURA 6. As camadas mais comuns empregadas nesse tipo de rede são: camada de convolução, camada de *pooling*, camada de ativação e camada totalmente conectada (TC).

FIGURA 6 – ESTRUTURA GERAL DE UMA CNN



FONTE: Adaptado de Alom *et al*. (2019).

De modo geral, a arquitetura da CNN pode ser dividida em duas partes, a primeira onde ocorre a extração das características e a segunda onde ocorre a classificação (Alom *et al*., 2019).

Na parte da extração das características, onde ficam localizadas as camadas de convolução e de *pooling*, a partir da entrada, as camadas iniciais são responsáveis pela extração de características simples, enquanto as camadas mais profundas são especializadas em características complexas. Esse tipo de estrutura torna a rede mais robusta às variações da entrada. Por exemplo, se a entrada da rede é uma imagem, a primeira camada poderia detectar apenas as orientações das linhas; a segunda camada poderia detectar cantos; e a terceira uma combinação das anteriores (Lecun *et al*., 2015).

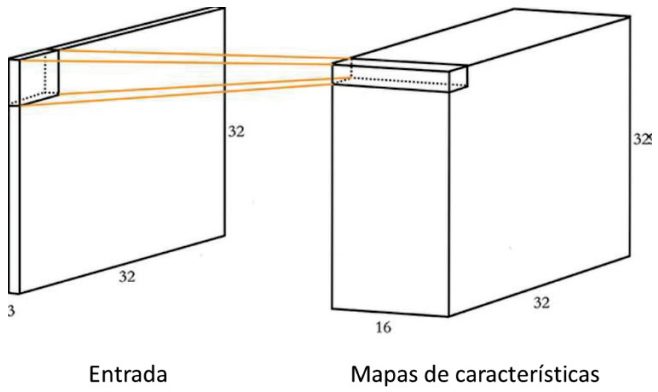
Para a segunda parte da arquitetura, a classificação, são utilizadas as camadas TC seguidas de uma função classificadora, como a função *softmax.* Com base na classificação feita, a rede apresenta uma resposta.

2.1.2.1 Camada de convolução

Rawat, Waseem e Wang (2017) definem as camadas de convolução como extratoras de características, que têm a função de aprender os padrões e características apresentadas pelos dados de entrada.

Como resultado das camadas de convolução, temos a geração de mapas de características, ilustrados na FIGURA 7. Cada neurônio desses mapas possui um conjunto de pesos treináveis, também chamados de filtros, ligados à camada anterior da rede. Ou seja, os dados de entrada sofrem a operação de convolução com os filtros para formar um mapa de característica. Todos os filtros, e consequentemente, os mapas gerados pela camada de convolução possuem pesos diferentes, de modo a maximizar as características extraídas dos dados de entrada (Rawat; Waseem; Wang, 2017). O resultado dessa operação pode ser seguido por outras camadas como: camada de *pool,* camada de ativação ou camada TC.

FIGURA 7 – CAMADA DE CONVOLUÇÃO



FONTE: Adaptado de Krohn *et al.* (2020).

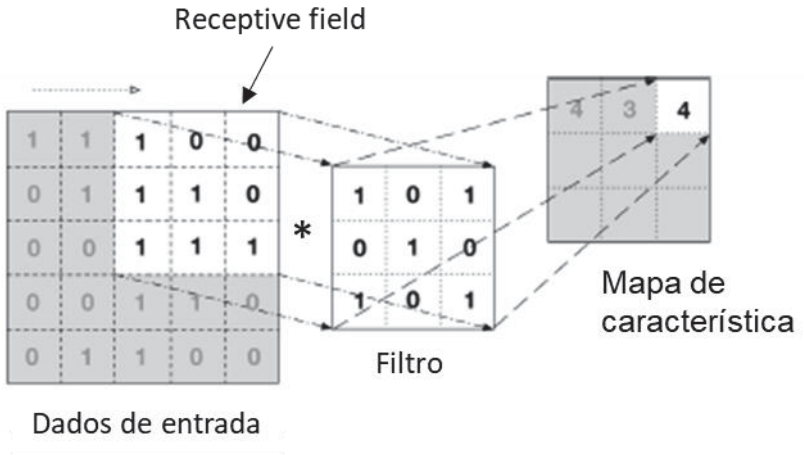
Os mapas de características, que são o resultado da operação da camada de convolução, são definidos como:

$$\boldsymbol{Y}_i^l = \boldsymbol{B}_i^l + \sum_{j=1}^{j^{l-1}} \boldsymbol{F}_{i,j}^l * \boldsymbol{Y}_j^{l-1}, \tag{1}$$

onde, $\boldsymbol{B}_i^l$ é o viés da camada l do mapa de característica i com $i \in [1, \ldots J^l]$; $\boldsymbol{F}_{i,j}^l$ o filtro que conecta o j-ésimo mapa de característica $\boldsymbol{Y}_j^{l-1}$, da camada $l-1$, com o i-ésimo filtro da camada l, sendo $j \in [1, \ldots, J^{l-1}]$; $\boldsymbol{Y}_i^l$ o mapa de característica i da camada l; J^{l-1} a quantidade de mapas de característica da camada $l-1$ e "$*$" a operação de convolução (Chari, 2018).

A FIGURA 8 ilustra o processo de geração de um mapa de característica. De maneira simplificada, o filtro "desliza" sobre os dados de entrada e a cada passo que é dado – em cada região (*receptive field*) - é realizada a operação de multiplicação elemento a elemento gerando um elemento do mapa de característica. Para um único mapa de característica, o processo de multiplicação elemento a elemento é expresso pela equação (2):

FIGURA 8 – OPERAÇÃO DE CONVOLUÇÃO



FONTE: Adaptado de Patterson e Gibson (2017).

$$mc_{i,j} = \sum_q \sum_r I_{i+q-1,j+r-1} F_{q,r}\,, \tag{2}$$

onde, $mc_{i,j}$ é o elemento i,j do mapa de característica resultante da operação de convolução, obtido a partir da multiplicação dos pesos de entrada $I_{i+q-1,j+r-1}$ com os pesos do filtro $F_{q,r}$, que possui dimensão Q x R, com $q \in [1, \ldots, Q]$ e $r \in [1, \ldots, R]$.

Quando se define uma rede neural também é necessário definir algumas configurações de funcionamento. Essas configurações são parâmetros denominados hiperparâmetros; eles estão presentes nas camadas da rede e nos algoritmos de

aprendizagem. De forma geral, eles são responsáveis por modelar o resultado da rede, isto é, como são extraídas as características dos dados de entrada, tempo de treinamento entre outros fatores. Logo, os valores assumidos são variados e não existe um valor correto; eles usualmente são escolhidos pela experiência do pesquisador ou por extensivos trabalhos da literatura.

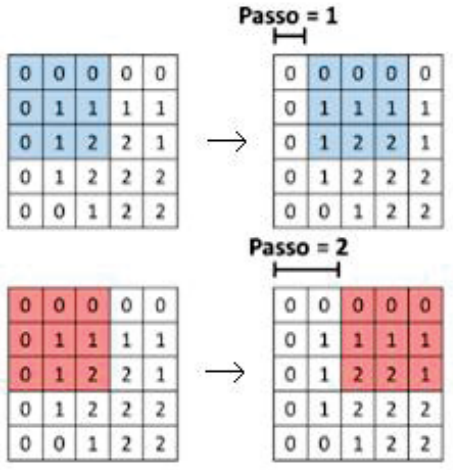
A camada de convolução possui três hiperparâmetros que serão definidos a seguir: tamanho do filtro, s*tride* (ou passo) e z*ero-padding*.

O tamanho do filtro influencia a dimensão de saída do mapa de característica e consequentemente, a extração de características das camadas seguintes. Tamanhos comuns dos filtros variam de 7x7 até 1x1, mas podem chegar a cobrir toda a dimensão da imagem de entrada. O tamanho do filtro depende da finalidade da rede, filtros menores vão detectar características menos complexas enquanto filtros maiores podem captar estruturas mais complexas - pois cobrem um *receptive field* maior - porém, as redes possuem melhor desempenho com filtros menores do que com filtros maiores (Ahmed e Karim, 2020; Khanday e Dadvandipour, 2020 e Maitra *et al*., 2018).

O *Stride* regula o tamanho do “passo” que o filtro realiza sobre a imagem de entrada. Ele influencia o quanto os *receptive fields* irão se sobrepor (Patterson; Gibson, 2017). Os valores mais comuns para o passo são 1 e 2 (FIGURA 9). Valores superiores a 2 não são uma boa abordagem pois podem ocorrer “saltos” de alguma região da imagem, consequentemente, pode se perder alguma informação. Por outro lado, valores acima de 2 fazem com que a rede seja mais rápida pois ocorre a diminuição da quantidade de parâmetros produzidos e operações realizadas. Em outras palavras, o mapa de característica resultante é menor quando o *stride* aumenta, uma vez que a varredura feita na imagem possui menos sobreposições (Krohn e*t al*., 2020).

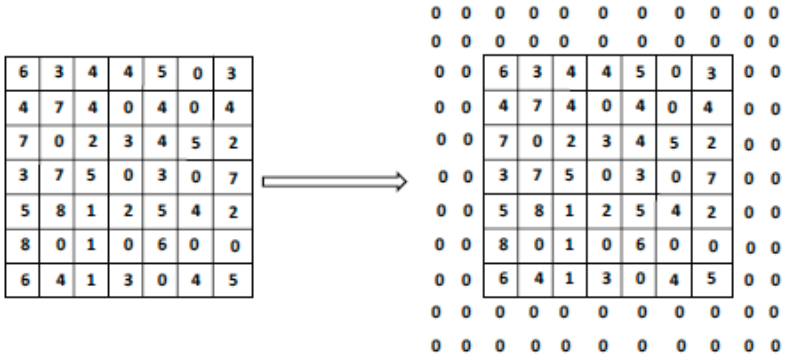
O hiperparâmetro *zero-padding* é utilizado para controlar a dimensão de saída do mapa de característica e como recurso para não perder informações das bordas nas imagens. Funcionalmente, o *zero-padding* consiste em adicionar zeros em torno das bordas da imagem de entrada, como mostra a FIGURA 10.

FIGURA 9 – DIFERENTES VALORES DE PASSO

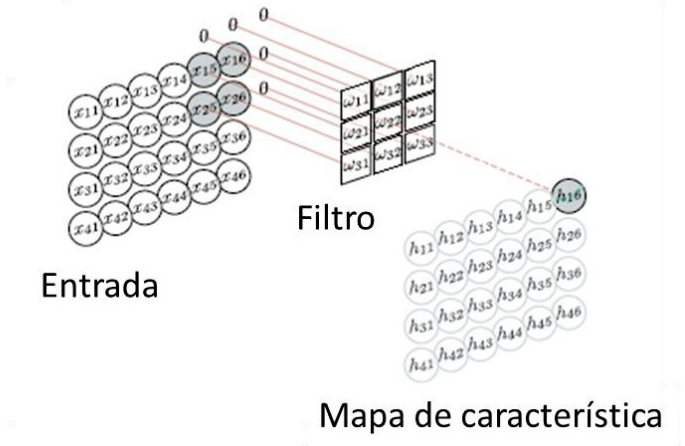


FONTE: Adaptado de Araújo (2018).

FIGURA 10 – ZERO-PADDING



(a)



(b)

FONTE: Adaptado de (a) Aggarwal (2018) e (b) Prince (2023).

O valor utilizado no *zero-padding* depende da construção da rede. Quando não é utilizado o *zero-padding,* ocorre uma redução da dimensão dos dados que são passados pelas camadas, que para alguns casos pode ser uma redução excessiva de acordo com o tamanho do filtro utilizado, comprometendo a quantidade de camadas possíveis a serem adicionadas posteriormente na rede e o nível de detalhamento das características extraídas. O caso oposto, quando o *zero-padding* faz com que não se percam dados, permite a utilização de mais camadas na arquitetura, sem correr o risco de degenerar os dados de entrada (Goodfellow *et al*., 2016).

O cálculo da dimensão dos mapas de característica é dado pela equação (3):

$$\mathrm{DF}_C = \frac{\mathrm{E} - \mathrm{f} + 2P}{S} + 1, \tag{3}$$

onde, E é o tamanho da imagem (ExE), f o tamanho do filtro ($\mathrm{f}x\mathrm{f}$), P a quantidade de *zero-padding* que a imagem possui, S o tamanho do passo e DF_C o resultado da operação. Quando o tamanho da imagem e do filtro não são simétricos é possível utilizar a mesma equação (3) de forma isolada para cada componente, isto é, apenas para a largura e para a altura, usando as respectivas dimensões para a imagem de entrada e filtro.

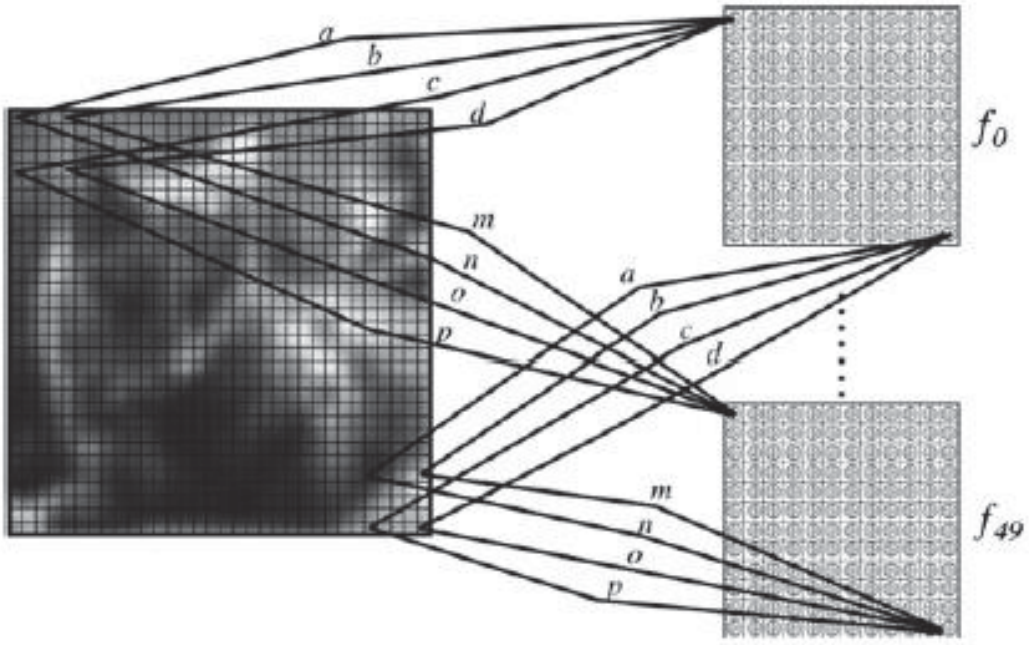
Por causa desses mecanismos de funcionamento, Goodfellow *et al*. (2016) e Lecun *et al*. (2015) destacam três características que fizeram as camadas de convolução, por consequência as CNN, populares. A dimensionalidade das conexões, compartilhamento de parâmetros e representação invariante.

A dimensionalidade das conexões é o resultado da utilização de filtros com dimensões menores que a imagem de entrada. Isso implica que a rede precisa de menos memória para guardar os parâmetros, pois existem menos conexões entre os neurônios do que nas RNAs; consequentemente, existe a redução na quantidade de operações realizadas, aumentando a eficiência da rede.

O compartilhamento de parâmetros é a principal característica da CNN. Essa característica torna a rede menos complexa, necessitando de menos parâmetros e, consequentemente, menos memória (Gorach, 2018). Compartilhamento de parâmetros significa que ao invés de aprender um conjunto de parâmetros para cada localização, apenas um parâmetro é aprendido por vez, como ilustra a FIGURA 11.

Isso só é possível porque são utilizados pequenos filtros que se deslocam pela imagem capturando as suas características.

FIGURA 11 – COMPARTILHAMENTO DE PARÂMETROS



FONTE: Habibi e Jahani (2017).

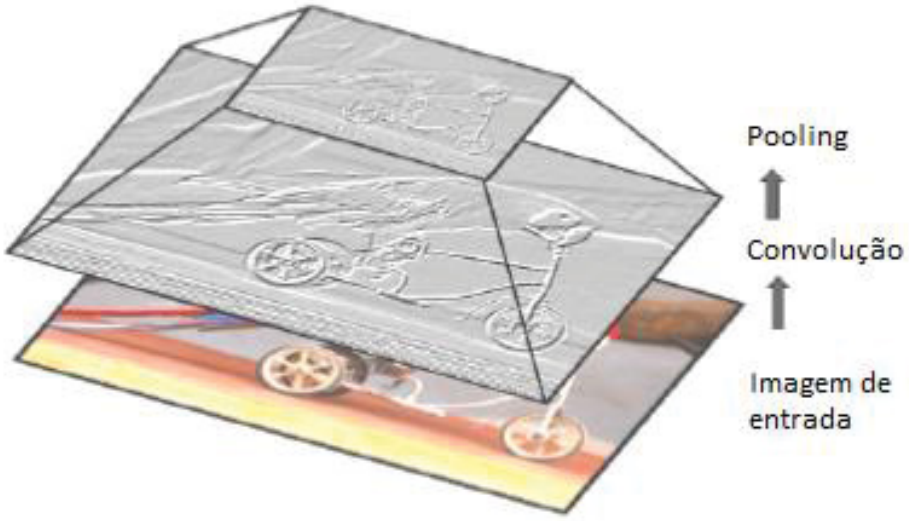
Como consequência do compartilhamento de parâmetros a rede ganha mais uma propriedade, a invariância da representação; ou seja, se a entrada varia a resposta muda da mesma forma (Goodfellow *et al*., 2016). Essa invariância da rede deve-se à forma que a rede aprende, ou seja, a rede aprende as características da imagem por regiões. Após aprender todas as regiões da imagem, a rede consegue aprender uma representação geral da imagem (Patterson; Gibson, 2017). Por exemplo, em uma imagem de uma face a rede pode aprender que existe alguma característica específica no lado esquerdo do rosto, e, ao longo das variações presentes das imagens da mesma face ela é capaz de identificar a existência dessa característica.

2.1.2.2 Camada de *pooling*

A camada de *pooling* é uma operação de redução de dimensão (*downsample)*, ou seja, é utilizada para reduzir o tamanho da imagem - ou a resolução

- de entrada combinando as características de uma região em um valor representativo (Gorach, 2018; Lecun *et al*., 2015), como ilustrado na FIGURA 12.

FIGURA 12 – OPERAÇÃO DE POOLING



FONTE: Adaptado de Alom *et al.* (2019).

A operação de *pooling* traz alguns benefícios para a rede. Primeiro, reduzindo o tamanho do mapa de característica torna o mapa mais invariante, isto é, as pequenas translações e distorções que os dados de entrada podem apresentar não afetam o resultado final. Dessa forma, a operação age como uma “peneira” definindo se o mapa possui ou não alguma característica (Goodfellow *et al*., 2016). A localização exata de cada característica não é fundamental, pois a posição relativa de cada característica é mantida, ou seja, a “distância” de uma característica em relação às outras não é alterada (Khan *et al*., 2020). Segundo, torna a rede menos complexa, pela redução de parâmetros, e ajuda a melhorar a generalização. Por último, reduz o *overfitting* da rede evitando que a rede decore os dados de treinamento (Khan *et al.*, 2020; Wang *et al.*, 2012).

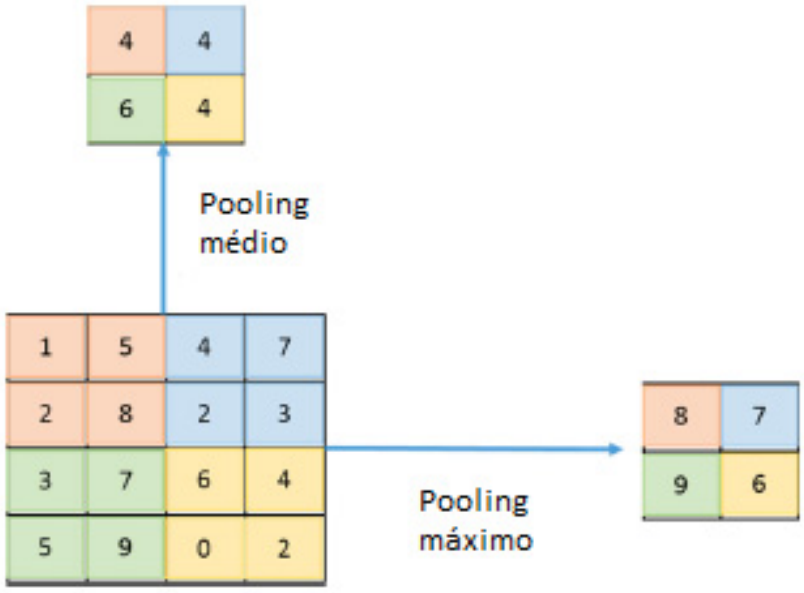
Assim como na camada de convolução, a camada de *pooling* possui os mesmos hiperparâmetros e a dimensão de saída é dada pela equação (4):

$$DF_p = \frac{D - \mathrm{f}}{S} + 1, \tag{4}$$

onde D representa o tamanho da entrada, f o tamanho do filtro utilizado, S o valor do passo e DF_p a dimensão de saída da operação.

Existem diferentes tipos de camadas de *pooling*, as duas mais comuns são o *pooling* máximo e o *pooling* médio, como mostra a FIGURA 13.

FIGURA 13 – POOLING MÁXIMO E POOLING MÉDIO



FONTE: Adaptado de Alom *et al*. (2019).

O *pooling* médio foi usado primeiramente por Lecun *et al*. (1998), enquanto a rede AlexNet (Krizhevsky *et al*., 2012) introduziu a variação de *pooling* máximo durante o ILSVRC-2012.

A operação de *pooling* máximo seleciona o maior elemento de cada *receptive field* e descarta os outros valores, tornando o elemento selecionado o seu representante. Já a operação chamada de *pooling* médio usa a média aritmética dos elementos do *receptive field*, gerando um representante com base em todos os elementos.

Formalmente, a operação de *pooling* máximo pode ser definida como:

$$\boldsymbol{Y}_k(i,j) = \mathrm{Max}_{(p,q)\in\mathcal{R}_{ij}}(x_{pq}), \tag{5}$$

onde, o resultado da operação é denotado por $\boldsymbol{Y}_k$ que é associado ao k-ésimo mapa de característica. O $x_{p,q}$ denota o elemento na localização na posição (p, q), contido no *receptive field* $\mathcal{R}_{ij}$ na posição (i, j) (Yu *et al.*, 2014). A região $\mathcal{R}_{ij}$ é definida como a janela de tamanho H x W, deslocada com passo S, com $p \in [iS, iS + H - 1]$ e $q \in [jS, jS + W - 1]$.

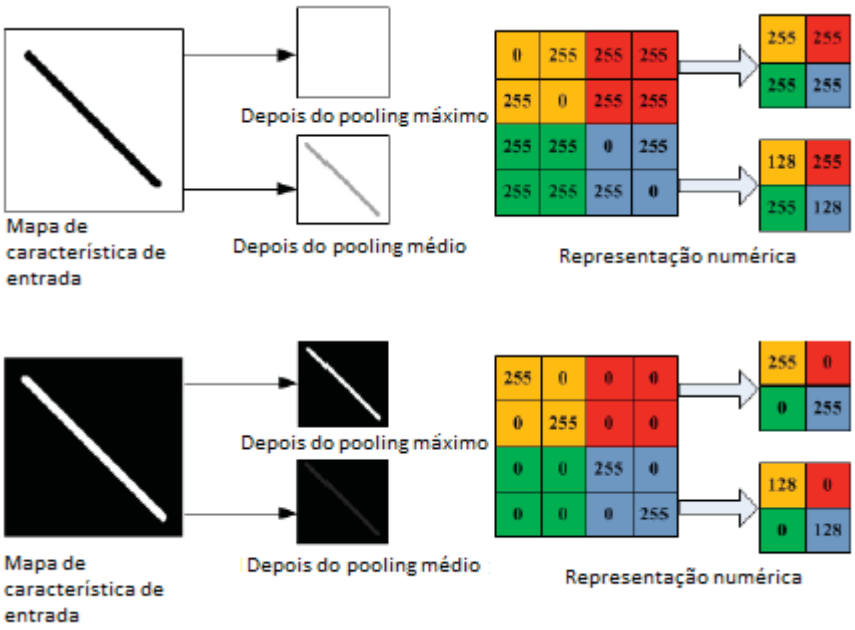
A formulação matemática para o *pooling* médio é

$$\boldsymbol{Y}_k(i,j) = \frac{1}{|\mathcal{R}_{ij}|} \sum_{(p,q)\in\mathcal{R}_{ij}} x_{pq}, \tag{6}$$

onde, $|\mathcal{R}_{ij}| = H$ x W representa a quantidade de elementos da região $\mathcal{R}_{ij}$ (Yu *et al.*, 2014).

Embora as duas opções de *pooling* sejam muito utilizadas, a FIGURA 14 ilustra que ambas possuem falhas e que a escolha da utilização de uma ou outra deve recair sobre o conjunto de dados.

FIGURA 14 – FALHAS PARA CADA TIPO DE POOLING



FONTE: Adaptado de Yu *et al.* (2014).

2.1.2.3 Camada de ativação

As camadas de ativação podem ser utilizadas depois das camadas de convolução, *pooling* ou TC. Elas utilizam as FA realizando transformações não lineares nos pesos, dessa forma, transformam o seu valor o que pode acarretar a anulação ou ampliação desses. Khan *et al.* (2020) apontam que as funções de ativação servem também como aceleradores do aprendizado da rede e são funções de decisão que ajudam na aprendizagem dos padrões complexos dos dados.

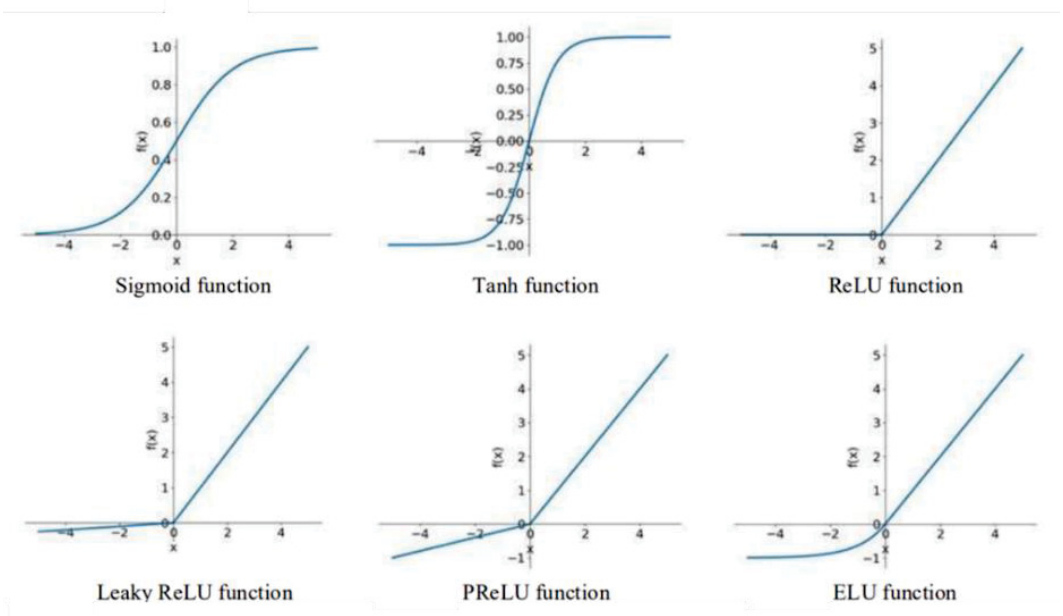
A operação da função ativação pode ser definida como:

$$G^l = g_a(\boldsymbol{Y}^{l-1}), \tag{7}$$

onde, $\boldsymbol{Y}^{l-1}$ é o resultado da camada anterior, $g_a(.)$ a função ativação e $\boldsymbol{G}^l$ o resultado da l-ésima camada após a ativação.

Existem diversas FA, e as mais utilizadas são ilustradas na FIGURA 15.

FIGURA 15 – GRÁFICO DAS FUNÇÕES DE ATIVAÇÃO



Fonte : Adaptado de Li *et al.* (2020).

As equações das FA Sigmoide (Werbos,1974), PReLU (He *et al*., 2015b), ELU (Clevert *et al*., 2016), ReLU (Nair; Hinton, 2010), Leaky ReLU ou LReLU (Maas *et al*., 2013), CReLU (Hannun *et al*., 2014), e Tanh são descritas na TABELA 1.

TABELA 1 – EQUAÇÃO DAS FUNÇÕES DE ATIVAÇÃO

Função	Equação	
Sigmoide	$f(x) = \frac{1}{1+e^{-x}}$	(8)
ELU	$f(x) = \begin{cases} x, & x \geq 0 \\ (e^x - 1), & x < 0 \end{cases}$	(9)
PReLU	$f(x) = \begin{cases} x, & x \geq 0 \\ \alpha x, & x < 0 \end{cases}$	(10)
ReLU	$f(x) = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases}$	(11)
Leaky ReLU	$f(x) = \begin{cases} x, & x \geq 0 \\ 0.01 * x, & x < 0 \end{cases}$	(12)
CReLU	$f(x) = \begin{cases} 0, & x < 0 \\ x, & 0 \leq x < 5 \\ 5, & x \geq 5 \end{cases}$	(13)
Tanh	$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	(14)

FONTE: O autor (2025).

Tanh foi uma das primeiras FA usadas nas RNA. Ela é centrada em zero e restrita ao intervalo [-1,1], o que reduz o esforço computacional para os ajustes de parâmetros, e aumenta a velocidade do algoritmo *backpropagation* (Nwankpa *et al.,* 2018b). Além disso, o maior valor da sua derivada ocorre quando a função a entrada é igual a zero, o que implica em gradientes maiores e uma atualização mais eficiente e, consequentemente, convergência mais rápida. A formulação matemática é dada pela equação (8).

Nair e Hinton (2010) propuseram a FA ReLU, equação (11), com o objetivo de fazer o aprendizado ser ainda mais rápido. A função tem uma parte linear para valores positivos e valores nulos para entradas negativas, implicando que a função converte as entradas para valores maiores ou iguais a zero. Essa função é a mais usada nas RNPs, pois ela não tem operações complexas ou valores exponenciais tornando-a facilmente otimizável; porém, ela oferece uma chance maior de *overfitting*, possibilidade de matar neurônios e treinamento lento quando o gradiente é constantemente igual a zero (Ajit *et al.*, 2020).

Para resolver o problema dos neurônios mortos, a FA LReLU desenvolveu a seguinte formulação: uma inclinação positiva para valores abaixo de zero – como mostra a equação (12). De forma generalizada, a FA PReLU expande o conceito da FA LReLU e cria variações na inclinação da reta, possibilitando uma gama maior de

possibilidades de ativações. As FA ReLU e LReLU apresentam maior estabilidade durante o treinamento e são menos sensíveis à inicialização dos pesos e a normalização dos dados (Li *et al*., 2021).

Outra forma de escapar dos elementos nulos é utilizar as FA ELU (equação (9)). Semelhante a FA LReLU, ELU adota uma função exponencial para as entradas negativas. Dessa maneira, a FA ELU produz valores próximos de zero, aumentando a velocidade de treinamento e podendo apresentar uma leve melhora em relação as outras FAs (Li *et al*., 2021).

A FA CReLU tem quase o mesmo comportamento da função ReLU. A diferença é que a FA CReLU adota um limiar máximo, como mostra a equação (13). Essa função é usada para ajudar no problema de gradiente explosivo.

2.1.2.4 Camada totalmente conectada

Como parte da classificação nas CNNs as camadas TC têm como função interpretar todas as características extraídas pelas camadas anteriores (Rawat; Waseem; Wang, 2017). A quantidade de camadas e neurônios a serem usados depende da arquitetura da rede; podendo ser usadas várias camadas para aprimorar a precisão da rede. Porém, como essa parte da rede é a mais densa em conexões ela acarreta em um alto custo computacional pela quantidade de parâmetros que possui (Aggarwal, 2018). Especificamente para as CNNs, a última camada TC possui o número de neurônios igual à quantidade de classes em que a rede é usada.

2.1.2.5 Camada Softmax

A função *softmax* é utilizada como última parte da rede após a camada TC, trata-se de uma função classificadora que irá converter as ativações dos neurônios em *scores* para o resultado da rede. Esses *scores* variam de 0 a 1 e a soma de todos é igual a 1. Para a tarefa de classificação de imagem, os *scores* representam a probabilidade de tal entrada ser da classe indicada. Dessa forma, a classe que possui o maior escore é o resultado da rede.

A função *softmax* é dada pela equação (15):

$$\sigma(\boldsymbol{z})_c = \frac{e^{z_c}}{\sum_{c=1}^{C} e^{z_c}}, \tag{15}$$

onde, $\boldsymbol{Z}$ é o vetor de saída da camada TC, c o índice dos neurônios da camada de saída com $c = 1, \ldots, C$, onde C é o número de classes, e $\sigma(\boldsymbol{z})_c$ o *score* para o elemento c (Gorach, 2018b).

2.1.2.6 Camada de normalização

A camada de normalização (Ioffe; Szegedy, 2015) tem a função de deslocar e redimensionar os pesos dentro de uma batelada $\mathcal{B}$ (um subconjunto do conjunto de dados). Originalmente, a função da camada de normalização era reduzir o problema do *internal covariate shift*, ou seja, atenuar a mudança na distribuição das ativações das camadas, que era alterada pelo processo de atualização dos pesos no algoritmo de *backpropagation*. As distribuições dos pesos entre as camadas, se forem muito diferentes, podem tornar o treinamento da rede mais lento e instável uma vez que o algoritmo de aprendizagem precisa ficar constantemente compensando essa diferença entre elas.

As fórmulas usadas durante o treinamento e a inferência da rede são distintas (Chollet, 2023). Durante o treinamento a fórmula usada é dada pela expressão (16):

$$\gamma * \left(\frac{\mathcal{B} - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \right) + \beta \tag{16}$$

onde, ϵ é uma pequena constante para evitar a divisão por zero, γ é a taxa de aprendizagem, β é o fator deslocamento aprendido. Durante a inferência a normalização é feita pela expressão (17):

$$\gamma * \left(\frac{\mathcal{B} - \mu_{moving}}{\sqrt{\sigma_{moving}^2 + \epsilon}} \right) + \beta \tag{17}$$

onde, μ_{moving} tem um valor inicial de 0 e σ^2_{moving} recebe o valor inicial de 1, representadas pelas equações (18) e (19), respectivamente.

$$\mu_{moving} = \mu_{moving} * momentum + \mu_B * (1 - momentum) \quad (18)$$

$$\sigma^2_{moving} = \sigma^2_{moving} * momentum + \sigma^2_B * (1 - momentum) \quad (19)$$

Neste contexto, o parâmetro $momentum$ funciona como coeficiente de amortecimento das estatísticas móveis, controlando a influência das médias e variâncias anteriores na atualização atual; tipicamente, adota-se valores entre 0,9 e 0,99 para equilibrar estabilidade e adaptabilidade das estimativas.

Esses parâmetros adicionais aumentam a complexidade e deixam a rede computacionalmente mais cara. Além disso, junto com a FA ReLU aumenta a probabilidade do gradiente explodir (Yang, G. *et al.*, 2019). Porém, a camada de normalização traz os seguintes benefícios (Prince, 2023):

- Propagação estável na fase *forward*: Se β tem um valor inicial de 0 e γ recebe o valor inicial de 1, os pesos terão variância igual a 1. Essa característica é especialmente desejável nas RNP residuais, que são propensas ao gradiente explosivo.
- Taxa de aprendizado maior: A superfície de perda das redes que possuem a camada de normalização torna-se mais suave e o gradiente muda de forma menos drástica. Isso permite o uso de uma taxa de aprendizado maior (Bjorck *et al*., 2018).
- Regularização: Como a camada de normalização transforma os dados de entrada e esses dados são diferentes entre as bateladas, ela funciona como se fosse uma adição de ruído. Isto é, dado um subconjunto inicial do conjunto de dados os outros subconjuntos são como variações do subconjunto inicial, com a adição de um ruído. Dessa forma, aumenta-se a variabilidade com pequenas modificações aumentando a performance da rede.

2.2 INICIALIZAÇÃO DOS PESOS

Um dos passos para a construção das CNN é a escolha dos pesos iniciais. O processo de treinamento vai modificar esses pesos iniciais diversas vezes até que se

chegue ao melhor valor possível. Duch e Korczak (1998) apontam que o processo de treinamento não compensa uma inicialização de pesos ruim, implicando em um treinamento mais lento e com alta probabilidade de ficar preso em mínimos locais. Em outras palavras, uma boa inicialização faz com que as CNNs performem melhor e mais rápido (Narkhede *et al*., 2022; Yam; Chow, 2000).

Três inicializações são descritas a seguir: Glorot (Glorot; Bengio, 2010), He (He *et al.*, 2015b), e *Narrow-normal* (Krizhevsky *et al*., 2012). Essas inicializações são parte de um grupo chamado de *variance-scaling-based initialization*, em que o foco é conservar a magnitude dos pesos, isto é, manter a variância dos pesos perto de 1 para cada camada, tentando evitar o problema do gradiente que explode ou desaparece. As inicializações de pesos são descritas na TABELA 2.

A inicialização Glorot, representada pela expressão (20) e também conhecida como inicialização Xavier, foi proposta em 2010 para atingir uma convergência mais rápida da rede e também para que cada camada da rede tivesse a mesma variância. A distribuição uniforme gera os fatores de escala In e Out que são relativos à camada anterior.

TABELA 2 – INICIALIZAÇÃO DOS PESOS

Inicialização	Expressão	
Glorot	$\theta \sim U\left(0, \frac{2}{In + Out}\right)$	(20)
He	$\theta \sim N\left(0, \frac{2}{In}\right)$	(21)
Narrow-normal	$\theta \sim N(0, 0.01)$	(22)

FONTE: O autor (2025).

LEGENDA: θ são os pesos, In o resultante da multiplicação da dimensão dos filtros da camada (largura × altura × número de canais), e Out é o resultante da multiplicação dos mapas de características da mesma camada (largura × altura × número de canais).

A inicialização He foi desenvolvida para ser mais eficiente com as FAs ReLU e PReLU. Essa inicialização é baseada na distribuição normal, com os mesmos fatores de escala da inicialização Glorot (expressão (21)). Além disso, essa inicialização faz com que o desvio padrão seja perto do valor 1 e a média perto de 0, o que acelera o treinamento (Olimov *et al*., 2021).

A última inicialização é a *Narrow-normal* representada pela expressão (22). A distribuição normal gera pesos com média igual a 0 e desvio padrão igual a 0,01.

2.3 TÉCNICAS DE PODA

Zhang *et al*. (2019) classificam os métodos de aceleração das CNN em três grandes áreas: redução de redundância, exploração de hardware e, por último, aprimoramento do treinamento e velocidade de inferência. Para essa última, as pesquisas são relacionadas a algoritmos de otimização, como variantes do método do Gradiente Descendente (GD, do inglês, *Gradient Descendent)*. A exploração de hardware busca inovações em implementação de GPUs, como o *Tensor Processing Unit* (TPU), um circuito integrado desenvolvido pela Google especificamente para aprendizagem de máquina. Por último, a redução de redundância utiliza técnicas de poda, compactação de redes ou direcionamento de dados de maneira mais eficiente.

2.3.1 Definição de poda

O cerne da técnica de poda é identificar os elementos redundantes ou menos úteis para a RNA, removendo o maior número deles com o menor impacto no desempenho. Nesse contexto, Qin *et al*. (2019) mostraram que existe redundância entre os filtros, os quais são semelhantes e possuem a mesma funcionalidade.

Como resultado e redutor de redundância, as técnicas de poda reduzem os parâmetros das RNA, podendo chegar a uma redução acima de 90%, diminuindo a memória necessária para a execução do algoritmo e melhorando o desempenho computacional sem interferir na acurácia (Frankle; Carbin, 2019; LeCun *et al*., 1990). Deste modo, uma RNA inicializada com pesos aleatórios, quando podada, pode atingir qualquer alvo, demonstrando que a RNA podada também é um aproximador universal e não sofre prejuízos em sua capacidade de generalização (Malach *et al*., 2020).

A equação (23) descreve a poda da rede:

$$\min L\left(\mathcal{N}(\boldsymbol{\theta^l} \odot \boldsymbol{m^l}), D\right) = \min \frac{1}{n} \sum_{i=1}^{n} L\left(\mathcal{N}\left(\boldsymbol{\theta^l} \odot \boldsymbol{m^l}\right), (x_i, y_i)\right) \tag{23}$$

onde, procura-se $Min\ \left(\left|\left|\boldsymbol{m^l}\right|\right|_{\mathbf{0}}\right)$. A variável $D = (x_i, y_i)$ é o conjunto de dados com os elementos $i = 1, \ldots, n.$; x_i o dado de entrada com o seu respectivo rótulo y_i; $\mathcal{N}$ é a rede neural com o um conjunto de camadas $\boldsymbol{\theta^l} \in \mathbb{R}^{\mathrm{d}^l \times \mathrm{d}^{l-1} \times \mathrm{h}^l \times \mathrm{w}^l}$ onde d^l, h^l e w^l

representam o número de filtros, altura e comprimento do filtro na l_{th}-ésima camada; $L(\cdot)$ é a função perda, e $\boldsymbol{m^l}$ é a máscara binária que seleciona os elementos que são podados, sujeitos a maximização do número de elementos a serem podados.

2.3.2 História das técnicas de poda

A primeira técnica de poda foi proposta por Sietsma e Dow (1988); a ideia apresentada era eliminar conexões repetidas ou não relevantes. Então, LeCun *et al*. (1990) introduziram a técnica *Optimal Brain Damage* (OBD) e da mesma forma que o *Optimal Brain Surgeon* (OBS) (Hassibi, B.; Stork, 1992; Hassibi *et al*., 1993) definiram a poda como o ajuste de conexão para zero. Ao mesmo tempo, diversos critérios para a seleção desses elementos que seriam podados foram aparecendo, tais como: importância (Mozer; Michael; Smolensky, 1989; Mozer; Michael; Smolensky, 1989), magnitude (Janowsky, 1989) e sensibilidade (Chauvin, 1989; Jianchang Mao *et al*., 1994; Karnin, 1990; Weigend *et al*., 1991). Além disso, diferentes mecanismos, também foram propostos, como um esquema de regeneração dos neurônios (Tresp et al. 1997), poda com um algoritmo genético (Whitley, 1990) e algoritmo iterativo (Murase *et al*., 1991).

Embora outros trabalhos (Chung; Lee, 1992; Karnin, 1990; Stepniewski; Keane, 1997) também tenham proposto técnicas de poda, esses foram aplicados apenas às camadas TCs por causa da limitação computacional que os computadores tinham naquela época.

O retorno do interesse nas técnicas de poda só ocorreu após o trabalho de Hinton *et al.* (2012). Agora, com computadores mais potentes, é possível a execução das técnicas de poda e ainda aplicá-las nas RNPs. Nessa mesma época, a técnica de poda mais conhecida, a *dropout (*Krizhevsky *et al.*, 2012), começa a ser utilizada como uma técnica para melhorar a generalização e regularização da rede, isto é, para ajudar a reduzir o *overfitting*. Essa técnica inicialmente foi aplicada apenas na parte mais densa da rede, nas camadas TCs, e consiste na poda aleatória em $p\%$ dos pesos.

No entanto, no cenário atual, um problema gerado pelo aumento do tamanho da arquitetura foi a super parametrização. Alguns filtros têm a mesma funcionalidade para a rede (Qin *et al.,* 2019), ou seja, as redes têm tantas camadas e pesos treináveis que muitos são irrelevantes no processo de aprendizado ou se tornam repetitivos. Choi *et al.* (2008) também apontam que o número excessivo de neurônios é o principal

fator para ficar preso em mínimos locais. Dessa forma, Reed (1993) afirma que a regra geral para a melhor generalização de uma RNA é escolher a menor arquitetura possível para cumprir a tarefa desejada. No entanto, determinar o número correto de camadas ou pesos em uma rede é um desafio complexo. Ishikawa (1996) tentou definir o número ideal de neurônios, e Hagiwara (1993) mostrou que é necessário encontrar um equilíbrio entre o número de neurônios, precisão e esforço computacional, pois reduzir os neurônios pode causar uma convergência mais lenta e ter mais neurônios não implica em maior precisão.

Conforme discutido anteriormente, existem vários métodos de poda; porém, a técnica é aplicada de diversas formas. Essa variabilidade levanta questões sobre a técnica, como: Qual estrutura deve ser podada? Qual elemento devo podar? Quando é o momento de podar? Quais as formas de podar? Existe uma arquitetura ou conjunto de dados apropriado? A seguir, são definidas as características de funcionamento das técnicas de poda para responder estas perguntas.

2.3.3 Características das técnicas de poda

A seção anterior mostra uma visão geral do algoritmo de poda. Porém, a poda possui uma variedade em seu funcionamento, ou seja, a execução passo a passo muda dependendo do critério de escolha do elemento e do local onde é aplicado.

Por meio de uma classificação proposta das técnicas de poda existentes, são analisados os elementos que compõem cada abordagem de poda. Assim, com base em quatro características, cada uma visando algum aspecto da técnica, as técnicas de poda foram classificadas, por meio de: critério (como selecionar os candidatos a podar), estrutura (como a poda afeta a arquitetura), dinâmica (como efetuar as podas) e escala (como encarar o conjunto de pesos). A FIGURA 16 ilustra a classificação proposta.

O **critério** está relacionado com a forma que é feita a seleção de pesos e é separado em oito opções:

- Comparação fixa – neste critério, um limiar η é estabelecido; se o peso for maior ou menor que o limite, a poda é realizada. Este tipo de critério também pode ser encontrado na literatura como *data-free* porque os elementos podados não dependem diretamente do conjunto de dados;

FIGURA 16 – CLASSIFICAÇÃO DAS TÉCNICAS DE Ppoda

FONTE: O autor (2025).

- Comparação aleatória - um valor aleatório é usado como limiar η para a seleção de elementos, ou seja, geram-se valores a partir de uma distribuição estatística ou outra técnica estatística;
- Sensibilidade – o primeiro passo desse critério é uma pré-poda seguida pela avaliação da performance da rede. Esse processo estima se esses elementos podados geram alguma melhoria no desempenho da rede. A poda final é realizada com base nos elementos que afetam minimamente a precisão da rede em sua ausência. Esta técnica é uma poda dependente de dados, pois o critério precisa de alguma parte do conjunto de dados para verificar se a poda foi benéfica para a rede;
- Ordenação – a principal característica é a classificação. É comum usar esse critério com o critério sensível, onde a influência dos elementos na rede é ordenada, eliminando aqueles com maior influência na perda. Consequentemente, esse critério de classificação pode gerar um limite para seleção de elementos de poda, mas, diferentemente do critério de comparação fixa, o critério de classificação não é estático. Dessa forma, é possível definir dinamicamente uma correspondência entre uma porcentagem ou um número de elementos podados, fornecendo melhor controle para a técnica;

- Híbrido – esta categoria inclui critérios como heurísticas novas ou aquelas já conhecidas da literatura, procedimentos com múltiplas etapas ou uma combinação dos critérios anteriores;
- Norma – avalia os pesos da rede por alguma equação que usa a norma ℓ_p com $p \in [0,1,2]$, ou uma combinação deles;
- Gradiente – Este método modifica e usa o GD para impor a poda; ele usa diferentes penalizações, como a regularização ou alguma regra específica de atualização de peso;
- Grafo – esta técnica aproveita as propriedades inerentes dos grafos para avaliar os elementos da rede, usando os elementos como parte dos critérios de escolha, como conectividade, centralidade e comprimento do caminho.

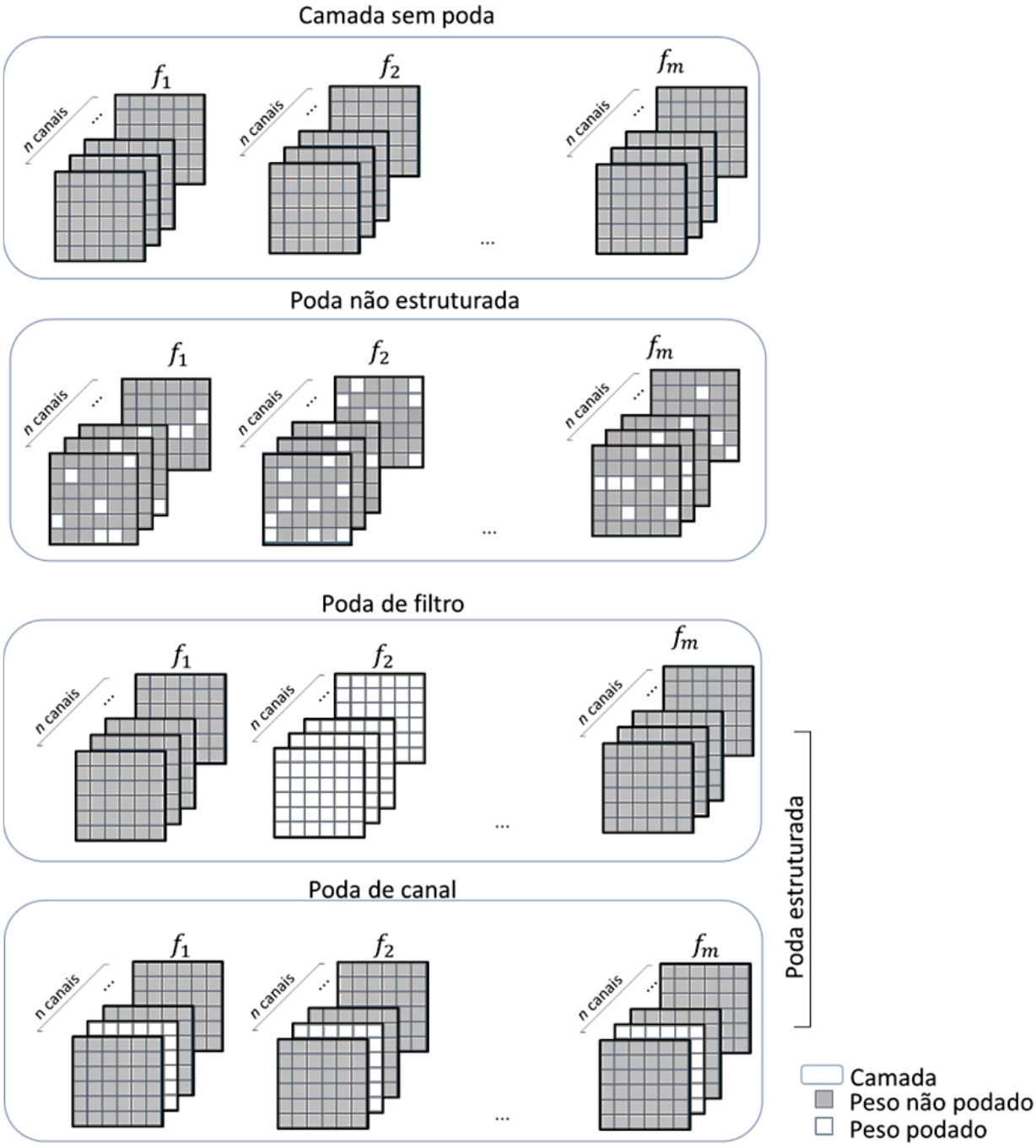
A característica da **estrutura** é definida de duas maneiras, que correspondem às definições apresentadas na literatura: poda estruturada e poda não estruturada, conforme ilustrado na FIGURA 17. Essa característica diz respeito à existência de uma mudança na arquitetura da rede, ou seja, uma mudança no número de filtros ou canais da rede.

A poda estruturada é feita no nível de grupos de pesos. A poda é realizada em uma linha ou coluna inteira de um filtro, em um canal inteiro ou até mesmo camadas. Sze *et al.* (2017) destacam que nesse tipo de poda o processamento de dados é fácil, pois reduz o custo necessário para sinalizar a localização dos pesos diferentes de zero, o que melhora a compressão de parâmetros, reduz o custo de armazenamento e a execução. Por outro lado, a desvantagem é a limitação dos graus de liberdade no processo de esparsificação, ou seja, a perda da elasticidade do aprendizado na rede.

A poda não estruturada aumenta a esparsidade por uma escolha arbitrária dos pesos. Esse processo produz uma matriz mista com pesos intocados e pesos nulos, consequentemente produzindo um modelo não estruturado, o que leva à necessidade de lembrar os índices dos pesos iguais zero. Além disso, essa abordagem adiciona quantidades consideráveis de dados e processamento computacional por causa das estruturas de índice, induzindo um procedimento menos eficiente. Além disso, a literatura apresenta mais duas classificações: poda de filtro rígido (HFP, do inglês, *hard filter pruning*) e poda de filtro suave (SFP, do inglês*, soft*

filter pruning). Enquanto o HFP não atualiza os filtros de poda, o SFP permite que os filtros podados se recuperem do processo de poda, o que não afetaria a capacidade do modelo. Aqui essa característica de recuperação é encarada como um atributo definido pelo autor do critério de poda.

FIGURA 17 – ESTRUTURA DA PODA



FONTE: O autor (2025).

A **dinâmica** diz respeito a “Como podar?” e é dividida em duas categorias: poda única ou iterativa. Na poda iterativa (Algoritmo 1), a rede é podada e retreinada por rodadas, ou seja, uma rodada inclui o treinamento da rede no conjunto de dados e uma etapa de poda. O número de elementos totais podados cresce em cada rodada de poda. Consequentemente, deve-se determinar adequadamente a quantidade de rodadas de poda a serem realizadas, uma vez que a rede pode apresentar perda de performance quando se aumenta a esparsidade. De forma geral, a quantidade de rodadas é definida experimentalmente por limitações de tempo de execução, recursos computacionais ou pela especificidade da técnica empregada. Como resultado dessas escolhas, a rede chega a um nível de esparsidade desejado. A técnica iterativa usa a ideia de que uma baixa taxa de poda permite que a rede recupere e fortaleça os elementos restantes, explorando a ideia da elasticidade da rede, ou seja, que na ausência de alguns pesos os restantes se adaptam para compensar a falta dos deles.

Algoritmo 1: Poda iterativa

Entrada: Rede neural pré-treinada $\mathcal{N}(\boldsymbol{\theta^l})$; critério de poda C_p; taxa de poda p (%) com $p \in [0,1]$, critério de parada $Stop$.

Enquanto não atingir o critério $Stop$ **faça:**

- Pode os parâmetros $\boldsymbol{\theta^l}$ do modelo baseado no critério de poda C_p com a taxa p
- Ajuste fino
- Atualização de hiperparâmetros.

Saida: Rede podada $\widehat{\mathcal{N}}(\boldsymbol{\theta^l})$

Na poda única (Algoritmo 2), apenas uma rodada de poda é realizada e pode ser seguida por ajuste fino; neste caso, uma grande parte da rede é podada de uma vez. O ajuste fino visa ajustar os pesos restantes da rede para compensar a ausência dos pesos podados.

A **escala** das técnicas de poda pode ser dividida em local ou global. A poda global interpreta os pesos de todas as camadas como um único conjunto de pesos; após a etapa de poda, há uma assimetria no número de elementos podados em cada camada. Enquanto isso, a poda local lida com cada camada individualmente, e o critério de poda pode variar de acordo com cada camada. Além disso, é possível

escolher quais camadas serão podadas, deixando propositalmente algumas camadas fora do processo de poda.

Algoritmo 2: Poda única

Entrada: Rede neural pré-treinada $\mathcal{N}(\boldsymbol{\theta^l})$; critério de poda C_p ; taxa de poda p (%) com $p \in [0,1]$.

Se $\boldsymbol{\theta^l}$ satisfazer o critério C_p **então:**

    Pode o modelo com p.

Ajuste fino

Saida: Rede podada $\widehat{\mathcal{N}}(\boldsymbol{\theta^l})$

Outro atributo pertinente às técnicas de poda é quando se deve podar. A poda antes do treinamento, realizada após a inicialização dos pesos, faz com que a rede atinja os mínimos locais, porém, tenha dificuldades para encontrar máscaras adequadas uma vez que a poda não é feita nos pesos finais, ou seja, existe uma perda na seleção de candidatos mais apropriados. Enquanto a poda após o treinamento pode não atingir os mínimos locais e são necessárias mais algumas épocas de treinamento para a rede chegue na sua performance máxima, o ajuste fino (*fine-tuning*).

2.3.4 Técnicas de poda na literatura

A técnica de poda mais conhecida, ou que utiliza uma técnica que pode se chamar de poda, é a *dropout,* proposta por Krizhevsky *et al*. (2012). Ela é utilizada para melhorar a generalização e regularização da rede, isto é, para ajudar a reduzir o *overfitting*. Inicialmente, foi aplicada apenas nas camadas TCs e consiste na poda de forma aleatória em $p\%$ dos pesos de camada alvo. De forma semelhante, Luo *et al*. (2020) propuseram a técnica *random mask,* onde é aplicada uma máscara aleatória nos mapas de características; embora essa técnica não diminua o número de parâmetros - uma vez que o número de parâmetros continua o mesmo – ela diminui o custo computacional e proporciona maior robustez enquanto mantém o nível de acurácia. De outra forma, Hu *et al*. (2016) propuseram a poda utilizando a porcentagem média de zeros na ativação (APoZ, do inglês, A*ctivations Average*

Percentage of Zeros), que mede a quantidade de zeros após a FA ReLU nos mapas de característica; porém, a técnica é dependente dessa FA em específico.

Polyak e Wolf (2015) propuseram a poda de canais inteiros para obter uma rede mais veloz, uma vez que isso reduz drasticamente a quantidade de operações e conexões, caracterizando-se como uma poda estruturada. Da mesma forma, Li *et al*. (2016) utilizaram a poda de filtros junto com suas conexões e mapas de características, e Zhang *et al*. (2020) empregaram um módulo de atenção (PFAM, do inglês, *Pruning Filter with Attention Mechanism*), em que os filtros podados são aqueles que têm a menor correlação com o módulo proposto - diferente dos dois métodos anteriores, o filtro podado é reconstruído para as próximas épocas podendo ainda contribuir para a rede.

Em relação à poda de comparação fixa, Han *et al.* (2015) apresentam uma seleção de pesos baseada em um limiar estabelecido, um múltiplo do desvio padrão dos pesos da camada, pesos abaixo desse limiar serão podados. Com o critério de sensibilidade, Lin *et al*. (2020) propuseram a poda dinâmica com feedback (DFP, do inglês, *Dynamic Pruning with Feedback*); essa poda possui uma alocação dinâmica dos padrões de esparsidade e incorpora o erro do *feedback* da etapa de avaliação dos candidatos em uma etapa do treinamento; dessa forma, o algoritmo avalia o modelo esparso e aplica correções no modelo denso para corrigir qualquer erro gerado pelo processo de poda.

Para o critério de norma, Oliveira *et al.* (2024) propuseram uma técnica baseada na normal ℓ_2 e ℓ_0. A norma ℓ_2 é utilizado para evitar o *overfitting*, enquanto a norma ℓ_0 transforma os valores para perto de 0. Em outro trabalho, Ding *et al*. (2018), com o *Auto-balanced Filter Pruning* (AFP), os filtros são avaliados pela norma ℓ_1, descartando os redundantes. E para fortalecer os filtros ‘fortes’, uma norma ℓ_2 personalizada foi adicionada como uma penalidade para filtros considerados ‘fracos’.

No que diz respeito ao critério de gradiente, Zhang *et al.* (2023) apresentaram uma estrutura de poda de múltiplas granularidades (MGPF, do inglês, *Multi-Granularity Pruning Filter*) que utiliza uma máscara suave treinável para podar diferentes níveis de estrutura, como a poda de peso, poda de canal e poda de filtro. O treinamento da máscara suave e dos parâmetros da rede é feito simultaneamente com a norma ℓ_1.

Por fim, para os critérios que envolvem grafo, Shi *et al.* (2023) propuseram uma técnica menos suscetível aos dados de entrada, o método de poda baseado em

entropia de gráfico de von Neumann (VNGEP), que combina as informações intra e inter-canais da camada para identificar filtros que são menos informativos e mais substituíveis por outros filtros. Wang *et al.* (2021) apresentaram uma técnica de redução da redundância estrutural (SRR, do inglês, S*tructural Redundancy Reduction*), utilizando grafos para medir o nível de redundância de cada camada de convolução e selecionando as mais redundantes; a redundância é calculada pelo número de l −cobertura e pela equação que utiliza o peso de probabilidade que equilibra o tamanho do espaço quociente (o número total de classes de equivalência) e o número de filtros da camada.

2.3.5 Hipótese do bilhete premiado

Com as técnicas de poda bem estabelecidas dentro das CNNs e com resultados promissores dos trabalhos anteriores, como o de Han *et al*. (2015), utilizando poda por magnitude dos pesos, Frankle e Carbin (2019) propuseram o bilhete premiado.

A hipótese do bilhete premiado (Lth, do inglês, *The Lottery ticket hypothesis*) define que para uma rede iniciada aleatoriamente $\mathcal{N}(x, \theta_0)$, existe pelo menos uma sub-rede $\mathcal{N}'(x, m \odot \theta_0)$ - o bilhete premiado (Lt, do inglês, *lottery ticket*) - de no mínimo igual acurácia e com ao menos o mesmo número de épocas de treinamentos realizados.

Em outras palavras, uma rede iniciada aleatoriamente $\mathcal{N}(x, \theta_0)$, quando otimizada com o gradiente descendente estocástico (SGD, do inglês, *Stochastic Gradient descent*) em um conjunto de dados, chega a uma perda na validação l em j iterações com o teste de acurácia a. A sub-rede $\mathcal{N}'(x, m \odot \theta_0)$, treinada isoladamente com o SGD e o mesmo conjunto de dados, onde m é a máscara binária sobre os parâmetros θ do treinamento da rede $\mathcal{N}(x, \theta_0)$, alcança a acurácia a' e a perda na validação l' em j' iterações. Sendo a hipótese de que $\exists\ m$ em que $j' \leq j$ (treinamento proporcional), $a' \geq a$ (acurácia proporcional) e $\left||m|\right|_0 \ll |\theta|$ (menos parâmetros) (Frankle; Carbin, 2019).

O algoritmo para encontrar o bilhete premiado é dado no quadro abaixo:

QUADRO 1 – ALGORITMO DO BILHETE PREMIADO

1. Aleatoriamente inicializar a rede neural $\mathcal{N}(x, m \odot \theta_0)$, ($\theta_0 \sim \mathcal{D}$ inicializados por uma distribuição normal e m sendo uma máscara);
2. Treinar a rede por j iterações, chegando aos parâmetros θ_j;
3. Podar $p\%$ dos parâmetros em θ_j criando a máscara binária atualizada m';
4. Reiniciar os parâmetros que não foram podados para seus valores originais θ_0, criando um ticket premiado $\mathcal{N}'(x, m' \odot \theta_0)$;
5. Fazer $m = m'$ e repetir 2-4 até o nível de esparsidade desejado ser alcançado.

Fonte: Adaptado de Frankle; Carbin (2019).

Em outras palavras, o algoritmo do bilhete premiado é uma poda iterativa onde, a cada rodada de poda, aqueles pesos que não foram podados retornam para o seu valor inicial θ_0 que é o seu valor gerado pela distribuição de inicialização antes da realização do treinamento. A máscara m, que é matriz binária que controla quais pesos foram podados, inicia nula no primeiro passo do algoritmo e ao longo das rodadas de poda é atualizada, garantindo que o número de pesos podados ao longo das rodadas aumente e que o peso que já foi podado nas rodadas anteriores permaneça podado.

Diferente dos outros trabalhos de poda, o trabalho de Frankle e Carbin (2019) mostrou um aspecto estrutural presente nas CNNs. Após encontrar o bilhete premiado, quando a mesma rede é aleatoriamente reiniciada, ela chega a um desempenho pior que a do bilhete encontrado com os pesos originais. Isto é, a estrutura sozinha não é suficiente para encontrar a rede ótima, mas, se a rede for inicializada corretamente é possível alcançar o bilhete premiado. Dessa forma, a estrutura e a inicialização dos pesos da rede estão intrinsicamente ligadas.

2.4 TREINAMENTO

2.4.1 Gradiente descendente estocástico

O SGD é uma variação do GD. O SGD adiciona um fator estocástico a cada iteração do algoritmo escolhendo aleatoriamente um subconjunto $\mathcal{B}$ do conjunto de treinamento, a batelada, e computa o gradiente desse subgrupo. A regra de atualização é definida pela expressão (24):

$$\theta_{t+1} \leftarrow \theta_t - \gamma \cdot \sum_{i \in \mathcal{B}_t} \frac{\partial \ell_i[\theta_t]}{\partial \theta} \tag{24}$$

onde o subíndice t contém os índices do par entrada/resposta da batelada atual, ℓ_i é a função perda do elemento i, γ é a taxa de aprendizagem.

Essas bateladas não se repetem, ou seja, bateladas diferentes terão imagens diferentes. Quando o algoritmo utiliza todas as bateladas disponíveis dentro do conjunto de dados isso é definido como uma época de treinamento. O tamanho da batelada, ou seja, a quantidade de imagens, pode variar de acordo com a capacidade computacional, tempo disponível para execução ou com alguma definição para o conjunto de dados específico. Quando a batelada tem o tamanho de todo o conjunto de treinamento o algoritmo volta a ser não estocástico, ou seja, o GD.

Um ponto importante ao utilizar o SGD é que o resultado é inteiramente dependente do ponto inicial, apesar do algoritmo aumentar a performance do modelo ao longo das iterações o ponto de partida é crucial para o número de épocas e chegada ao mínimo global (Prince, 2023).

Uma prática comum é utilizar o SGD com taxa de aprendizado variada ao longo das épocas. Isto é, se utiliza uma taxa maior no início do treinamento e a cada N épocas essa taxa é reduzida. A ideia é que no começo do treinamento, com uma taxa de aprendizagem alta, o algoritmo consiga explorar diversas regiões, e nas épocas finais, em uma região mais favorável, ocorra um ajuste fino.

A variação do SGD utilizado nesse trabalho foi o SGD com momento. Essa variação adiciona a direção do gradiente da iteração anterior na iteração atual. Matematicamente a atualização dos pesos pode ser descrita pelas expressões (25) e (26).

$$v_t \leftarrow \beta \cdot m_t + (1-\beta) \sum_{i \in \mathcal{B}_t} \frac{\partial \ell_i[\theta_t]}{\partial \theta} \tag{25}$$

$$\theta_{t+1} \leftarrow \theta_t - \gamma \cdot v_t \tag{26}$$

onde, m_t é o momento e $\beta \in [0,1)$ é o grau de suavização ao longo do tempo.

2.4.2 Adaptive moment estimation

Outro algoritmo de aprendizagem utilizado nesta tese foi o *Adaptive moment estimation* (ADAM). Esse algoritmo normaliza o gradiente de atualização evitando que grandes gradientes produzam grandes atualizações, o que poderia levar a solução para longe do mínimo local, ou que pequenos gradientes produzam atualizações insuficientes, o que implica ficar preso em um mínimo local. A regra de atualização é dada pelas equações (27), (28) e (29):

$$\theta_{t+1} \leftarrow \theta_t - \gamma \cdot \frac{m_t}{\sqrt{v_t + \epsilon}} \tag{27}$$

$$m_{t+1} \leftarrow \beta_1 \cdot m_t + (1 - \beta_1) \sum_{i \in \mathcal{B}_t} \frac{\partial \ell_i[\theta_t]}{\partial \theta} \tag{28}$$

$$v_{t+1} \leftarrow \beta_2 \cdot v_t + (1 - \beta_2) \left(\sum_{i \in \mathcal{B}_t} \frac{\partial \ell_i[\theta_t]}{\partial \theta} \right)^2 \tag{29}$$

onde, ϵ é uma pequena constante para evitar a divisão por zero, β_1 e β_2 são o momento para as duas variáveis e pertencem ao intervalo $[0,1)$.

Um ponto negativo desse algoritmo é que ele pode ficar indo e voltando em torno do mínimo e só converge quando cai exatamente no mínimo global. Porém, ele é menos sensível ao valor da taxa de aprendizagem inicial e menos dependente do decaimento da taxa de aprendizagem ao longo das épocas de treinamento (Prince, 2023).

2.4.3 Early stopping

A técnica de *early stopping* é utilizada para parar o treinamento do modelo de acordo com algum critério visando evitar o *overfitting.* Para realizar tal tarefa o conjunto de dados é dividido em três subconjuntos: o conjunto treinamento, validação e teste. Durante o processo de treinamento, o *early stopping* utiliza o conjunto de dados de validação de forma comparativa ao conjunto de dados de treinamento, avaliando o erro da rede, permitindo uma avaliação contínua do desempenho. Se o erro do conjunto de dados de treinamento tende a ficar menor enquanto o erro do conjunto de

dados da validação tende a aumentar, isso é um indício que a rede está em estado de *overfitting,* isto é, a rede está decorando o conjunto de dados de treinamento. Esse comportamento é altamente prejudicial, pois, ao invés do modelo generalizar os padrões dos dados, ele os decora, tornando o modelo incapaz de se adaptar a dados desconhecidos.

2.4.4 Entropia cruzada categórica

A função perda utilizada para calcular o erro da rede foi a entropia cruzada categórica (CCE, do inglês, *categorical cross-entropy*). Ela mede a proximidade entre a distribuição prevista e a distribuição verdadeira. A função é definida pela equação (30).

$$L = CCE(y, \hat{y}) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{C} y_{ij} \ln(\hat{y}_{ij}) \tag{30}$$

onde, y_{ij} representa a classe verdadeira em codificação *one-hot*, isto é, 1 para a classe verdadeira e 0 para as demais. O valor $\hat{y}_{ij}$ é a probabilidade prevista pelo modelo para cada classe. O índice i refere-se aos n exemplos do conjunto de dados, e j o índice para as C classes existentes.

2.5 CONSIDERAÇÕES

De acordo com a revisão de literatura apresentada nesse capítulo, as RNPs são compostas de diferentes tipos de camadas e por muitas delas. Cada camada possui seus próprios hiperparâmetros e cada um deles influencia de uma forma específica a rede. Sobre a arquitetura, não existe um consenso na literatura da composição ideal de camadas a ser seguido. Por esses motivos a construção da rede, isto é, a definição de uma arquitetura com certa sucessão de camadas juntamente com seus hiperparâmetros, é complexa.

Além disso, devido ao tamanho das redes, a quantidade de parâmetros que são utilizados chega aos milhões, por consequência, os algoritmos de otimização são necessários para a redução da dimensão dos pesos ou para empregar uma melhor

finalidade para eles. No sentido de redução da quantidade de pesos, foi apresentada uma série de trabalhos da literatura que envolvem diversos critérios de poda. Em especial, a hipótese do bilhete premiado que vincula a arquitetura da rede com seus pesos iniciais.

Com essa fundamentação, no próximo capítulo, a metodologia proposta nesta tese é descrita junto com a proposta do algoritmo de poda iterativa.

3 MATERIAL E MÉTODOS

Com base na revisão da literatura, foi proposto um algoritmo de poda iterativa que se encontra descrito no Capítulo 4. No entanto, para o desenvolvimento do mesmo, utilizou-se uma série de conceitos fundamentais e necessários para o treinamento e validação da proposta. Dessa forma, neste capítulo são descritos todos os materiais e métodos utilizados.

3.1 HARDWARE E SOFTWARE

Os experimentos desta tese foram executados em diferentes computadores de acordo com a necessidade e disponibilidade, foram eles:

- Máquina com uma única placa de vídeo NVIDIA GeForce® RTX 2070 SUPER (8 GB) GPU, processador Intel® core™ i7-9700K e 16 GB RAM;
- Máquina com uma única placa de vídeo NVIDIA GeForce® RTX 4060 (16 GB) GPU, processador Intel® core™ i7-9700K e 32 GB RAM;
- Máquina com uma única placa de vídeo NVIDIA GeForce® RTX 3070 (8 GB) GPU, processador AMD Ryzen® 9 5950X e 128 GB RAM;
- Máquina com uma única placa de vídeo NVIDIA RTX A4000 (16 GB) GPU, processador Intel® core™ i7-12700KF e 128 GB RAM;
- Máquina com uma única placa de vídeo NVIDIA RTX A5000 (24 GB) GPU, processador Intel® core™ i7-12700KF e 128 GB RAM.

A linguagem de programação utilizada foi o Matlab para os experimentos com a Poda estocástica e a Camada Nula. Para o algoritmo Poda Estocástica com Camada Nula, a implementação foi realizada em linguagem Python, utilizando a biblioteca *TensorFlow*.

3.2 CONJUNTO DE DADOS

Os conjuntos de dados utilizados foram o *Modified National Institute of Standards and Technology* (MNIST), proposto por LeCun Yann *et al*. (1998), *Canadian*

Institute For Advanced Research (CIFAR-10), desenvolvido por Krizhevsky *et al*. (2009), *Flower* (Mamaev, 2021) e *Fruit* (Zhang, 2020).

O conjunto de imagens MNIST é composto por dígitos escritos à mão, em preto e branco. Possui 60 mil exemplares para o treinamento da rede e 10 mil exemplares para o teste. As imagens são de dimensão 28x28x1 pixels divididos em classes que vão do número 0 até o 9, como mostra a FIGURA 18.

FIGURA 18 – CONJUNTO DE DADOS MNIST

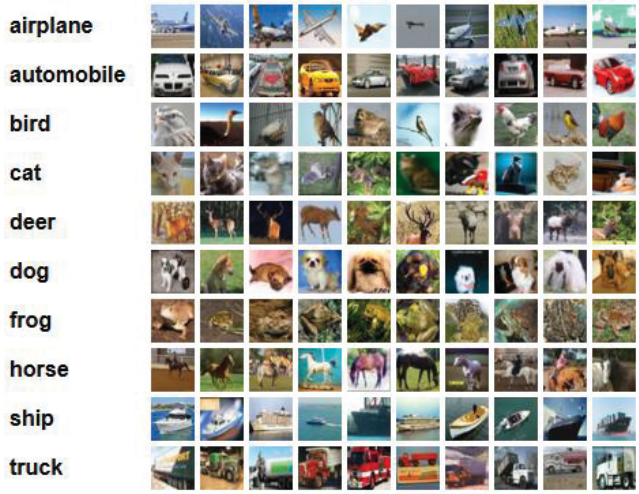
FONTE: Lecun *et al*. (1998).

O conjunto de dados CIFAR-10 (FIGURA 19) é composto por imagens coloridas, divididas em 10 classes distintas de objetos variados. Também possui 60 mil exemplares com tamanho de 32x32x3 pixels, divididos em 50 mil itens para treinamento e 10 mil para a classificação de teste.

O conjunto de dados *Flower* (Mamaev, 2021) é composto por 5 classes de flores, com média de 800 fotos por classe. As imagens são de alta resolução de tamanho 320x240x3 pixels e não há uma separação prévia do conjunto de fotos para o treinamento e teste. O conjunto é ilustrado na FIGURA 20.

O conjunto de dados Fruit possui 33 classes de frutas e 22495 imagens, que são divididas em 16854 para o treinamento e 5641 para o teste. As imagens são do tamanho 100x100x3 pixels e uma parte das classes é mostrado na FIGURA 21.

FIGURA 19 – CONJUNTO DE DADOS CIFAR-10

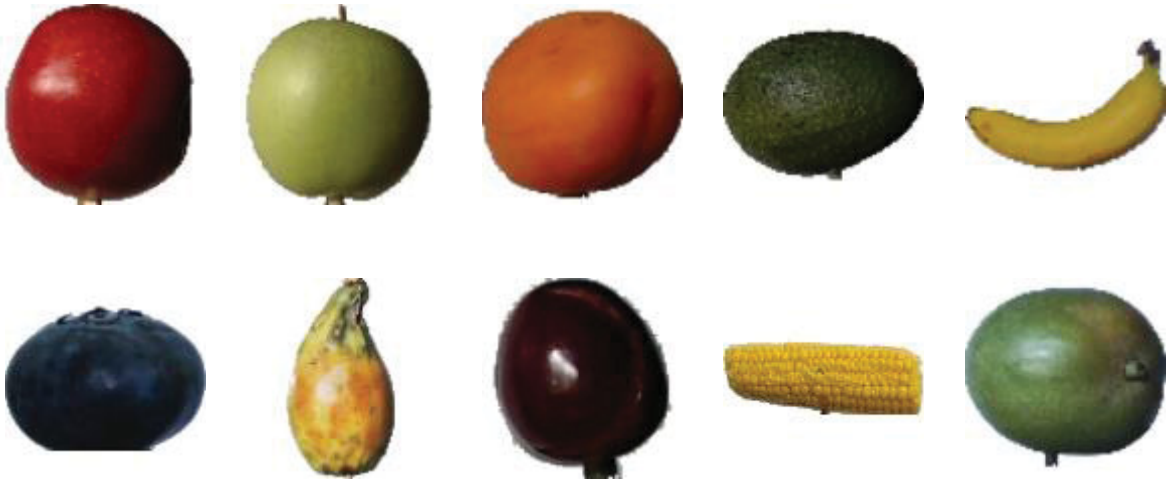


FONTE: Krizhevsky; Hinton (2009).

FIGURA 20 – CONJUNTO DE DADOS FLOWER

FONTE: Adaptado de Mamaev (2021).

FIGURA 21 – CONJUNTO DE DADOS FRUIT



FONTE: Adaptado de Zhang (2020).

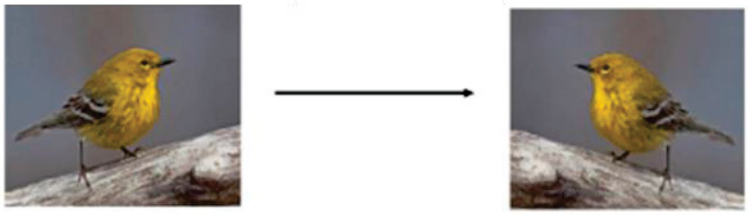
3.3 DATA AUGMENTATION

Data augmentation é uma das técnicas fundamentais em visão computacional e vem sendo usada nos primeiros e principais artigos de classificação de imagem, como no artigo de reconhecimento de caracteres em documentos (Lecun *et al*., 1998). Essa técnica pode ser definida como o par ordenado $(T(x), y)$ onde, x é a imagem de entrada, y é o rótulo respectivo e T é uma transformação (Geiping *et al*., 2022). Em outras palavras, essa técnica modifica as imagens originais e as transforma em imagens levemente alteradas.

Como a técnica é muito versátil, é possível criar ou gerar qualquer tipo de modificação imaginável em uma determinada imagem. Neste trabalho, são utilizadas apenas 3 transformações - espelhamento, translação e corte aleatório (Zhong *et al*., 2020) – as quais são descritas abaixo.

- Espelhamento: Nessa transformação é possível espelhar a imagem tanto na vertical quanto na horizontal, como mostrado na FIGURA 22.

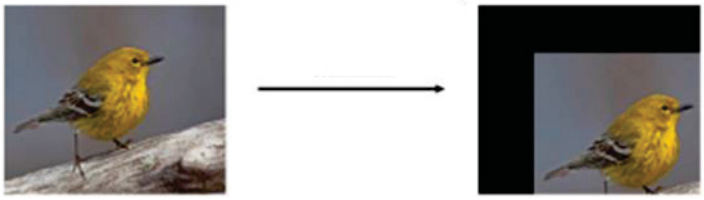
FIGURA 22 – DATA AUGMENTATION (ESPELHAMENTO)



FONTE: Adaptado de Kumar *et al.* (2024).

- Translação: Essa operação movimenta a imagem verticalmente e horizontalmente, como mostra a FIGURA 23.

FIGURA 23 – DATA AUGMENTATION (TRANSLAÇÃO)



FONTE: Adaptado de Kumar *et al.* (2024).

- Corte aleatório: Essa técnica imita cortes, ocultando paralelepípedos de tamanho aleatório e de coloração diversa nas imagens. A FIGURA 24 mostra exemplos da aplicação.

FIGURA 24 – DATA AUGMENTATION (CORTE ALEATÓRIO)

FONTE: Adaptado de Kumar *et al.* (2024).

A escolha da transformação a ser aplicada deve levar em consideração o conjunto de dados e deve ser feita com cautela. Por exemplo, no conjunto de dados MNIST a transformação de espelhamento pode confundir o número 6 com o número 9, ou ainda se a imagem do número 8 sofrer um deslocamento muito severo ele pode chegar a se assemelhar ao número 3.

Outros pontos importantes que devem ser levados em consideração são: quantidade de transformações aplicadas, pois ao criar novas imagens, deve-se tomar o cuidado de não criar um viés no conjunto de dados (Shorten; Khoshgoftaar, 2019); o aumento do custo computacional, pois as transformações no conjunto de dados demanda processamento computacional, que é adicionado ao processamento da RNA.

Porém, os benefícios da *data augmentation* podem ser listados, como: aumento da generalização do modelo, tornando-o mais invariante aos dados de entrada; redução do *overfitting,* pois torna as imagens mais diversas e evita que o modelo decore os padrões apresentados; e o aumento da acurácia, por tornar os padrões aprendidos mais robustos.

3.4 ARQUITETURAS

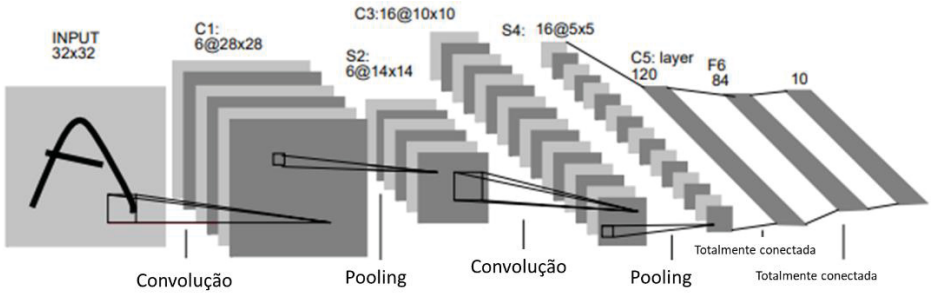
3.4.1 CNN3

As camadas de convolução da arquitetura CNN3 têm o tamanho de entrada definido de acordo com o conjunto de dados a ser utilizado. As camadas de convolução aumentam o número de filtros de acordo com a profundidade da camada (16, 32 e 128 filtros) e todas possuem *stride* igual a 1, a única camada que utiliza *zero-padding* é a última camada de convolução. As camadas de convolução são seguidas por camadas de *pooling* médio, com exceção da última camada, que é seguida pela camada TC. As FAs estão localizadas após a primeira camada de pooling e após a camada TC. A última camada da rede é a de classificação *Softmax*.

3.4.2 LeNet-5

A rede LeNet-5 (LeCun *et al.*, 1990) foi a primeira CNN utilizada em larga escala e originalmente foi proposta para o reconhecimento de dígitos escritos a mão. Além disso, introduziu as camadas de *Pooling* médio. Esta rede é composta por um total de 5 camadas, sendo 3 camadas de convolução e 2 camadas TC, como mostra a FIGURA 25.

FIGURA 25 – ARQUITETURA LENET-5



FONTE: Adaptado de Lecun *et al*. (1998).

A entrada da LeNet-5 é uma imagem 32x32. A primeira camada de convolução (C1) possui 6 filtros, a camada de *pooling* (S2) possui 6 mapas de características, diminuindo as dimensões de 28x28 para 14x14. A terceira camada de

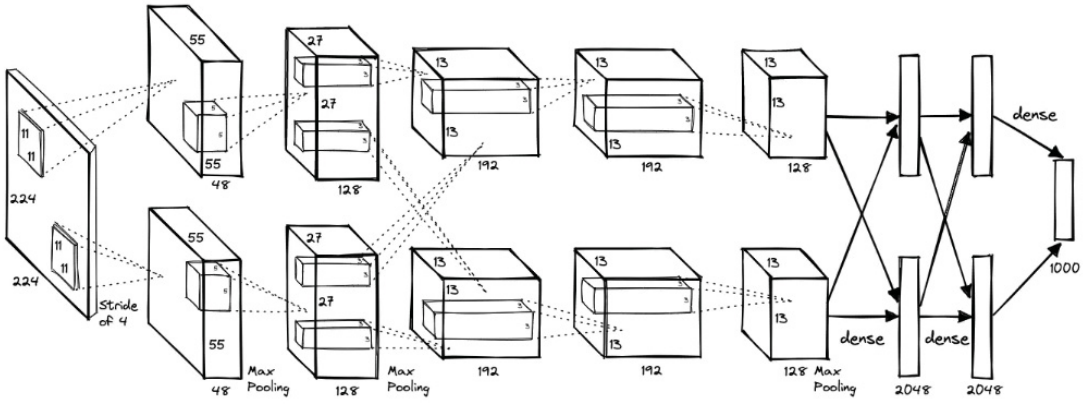
convolução (C3) possui 16 filtros e a quarta camada (S4) também possui 16 mapas de características, reduzindo as dimensões de 10x10 para 5x5. A quinta camada de convolução (C5) é representada como uma camada totalmente conectada pois a saída dos mapas de características são de tamanho 1x1. A sexta e a sétima camadas são totalmente conectadas.

3.4.3 AlexNet

A rede AlexNet foi de grande importância para as CNNs, pois a partir dela que as CNNs começam a ser usadas amplamente em diversos problemas de classificação de imagem. Além disso, foi a primeira a superar a classificação humana, assim como introduziu diversas otimizações nas CNNs, como o processamento de GPUs em paralelo (Alzubaidi *et al.*, 2021).

A sua arquitetura contém cinco camadas de convolução e três camadas TC. Os dados de entrada (224x224) passam pela primeira camada de convolução que possui 96 filtros de tamanho 11x11 com stride de 4. O resultado da primeira camada de convolução passa por uma normalização e uma camada de *pooling*, em seguida, passam para a segunda camada de convolução, que possui 256 filtros com stride de 5. A terceira e quarta camada de convolução possuem 384 filtros com stride de 3. A quinta camada também possui stride de 3, porém com 256 filtros. As duas primeiras camadas TCs possuem 4096 neurônios e a última, com o número de neurônios igual ao número de classes do conjunto de dados utilizado. A rede AlexNet está ilustrada na FIGURA 26.

FIGURA 26 – ARQUITETURA ALEXNET



FONTE: Adaptado de Pinecone e Systems (2024).

3.4.4 VGG

A arquitetura VGG (*Visual Geometry Group*), venceu as tarefas de identificação e classificação da competição ILSVRC 2014 e na época era uma das mais profundas que existia. Desenvolvida com filtros de tamanho 3x3 chegou a usar até 19 camadas. A FIGURA 27 ilustra a arquitetura da VGG16 e VGG19.

De modo geral, a arquitetura VGG possui os dados de entrada com imagens de tamanho 224x224. A quantidade de filtros das camadas convolucionais aumenta de acordo com a profundidade, as duas primeiras possuem 64 filtros, a terceira e quarta possuem 128 filtros, a quinta até a oitava camada possuem 256 filtros e as demais têm 512 filtros. As FA são colocadas sempre após uma camada de convolução e TC. A última camada da arquitetura é a função *Softmax*.

FIGURA 27 – ARQUITETURA VGG

ConvNet Configuration	
D	E
16 weight layers	19 weight layers
conv3-64 conv3-64	conv3-64 conv3-64
conv3-128 conv3-128	conv3-128 conv3-128
conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool	
FC-4096	
FC-4096	
FC-1000	
soft-max	

FONTE : Adaptado de Simonyan e Zisserman (2014).

LEGENDA: Coluna da esquerda (D) representa a arquitetura VGG-16 e a coluna da direita (E) representa a arquitetura VGG-19.

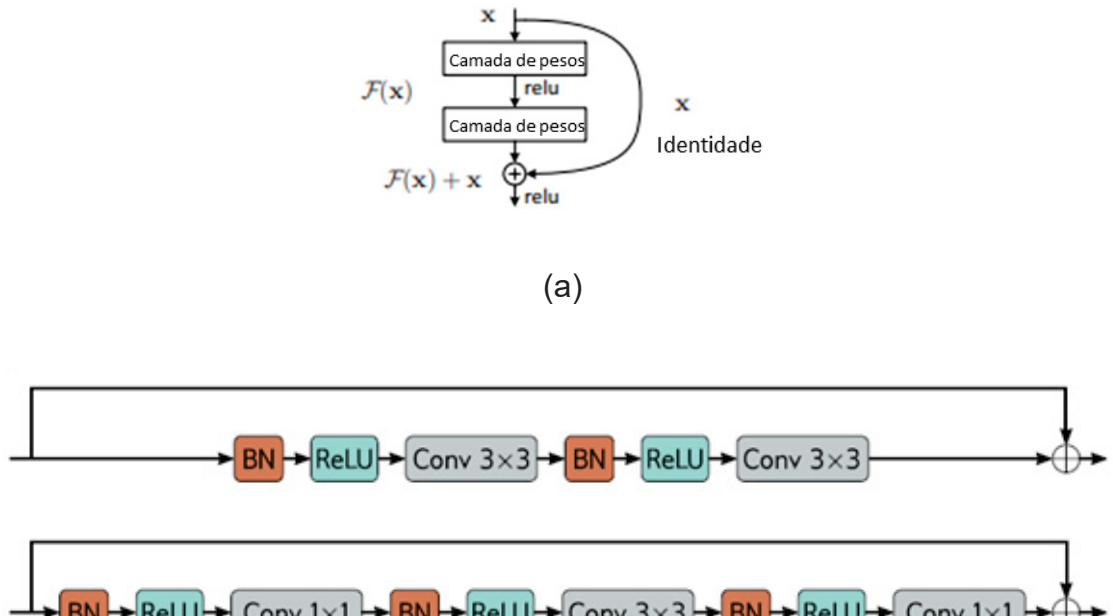
3.4.5 ResNet

As redes residuais (*Residual Network*), ou apenas ResNet (He et al., 2015a), receberam destaque por ganharem o concurso ILSVRC-2015 chegando a uma taxa de erro de apenas 3,57% e, além disso, por permitirem a existência de modelos profundos mitigando o problema do desvanecimento do gradiente (Li *et al*., 2021). As ResNets podem ser cerca de 20 vezes mais profundas que a AlexNet e até mais que 8 vezes para as redes VGG. Apesar de terem sido lançadas há 10 anos, elas continuam sendo utilizadas como base para muitas comparações de resultados em várias tarefas da visão computacional.

As ResNets possuem esse nome pelo uso dos blocos residuais. Cada bloco residual possui uma ou mais camadas e tem um atalho de conexão (*shortcut connection*). De forma ilustrativa, a FIGURA 28(a) mostra a sequência de uma camada de convolução, seguido pela FA e outra camada de convolução. Após os dados passarem por essa sequência de camadas o resultado é somado com a entrada original da primeira camada – o atalho de conexão. Esse empilhamento com a soma dos valores de entrada é chamado de bloco residual. Os blocos residuais assumem diversas configurações na composição das suas camadas, porém, os blocos modernos geralmente são maiores e usam a camada de normalização, como mostra a FIGURA 28(b).

Originalmente, a ResNet foi construída com 18, 34, 50, 101, 152 e 1202 camadas. Porém, as mais usadas são as com 20, 56 e 110 camadas - A FIGURA 29 ilustra a ResNet-20. Dessa forma, neste trabalho, as arquiteturas utilizadas foram as ResNet com 20, 56 e 110 camadas. O bloco residual utilizado foi o empilhamento de uma camada de convolução, seguido da FA, outra camada de convolução, terminando com uma camada de normalização.

FIGURA 28 – BLOCO RESIDUAL



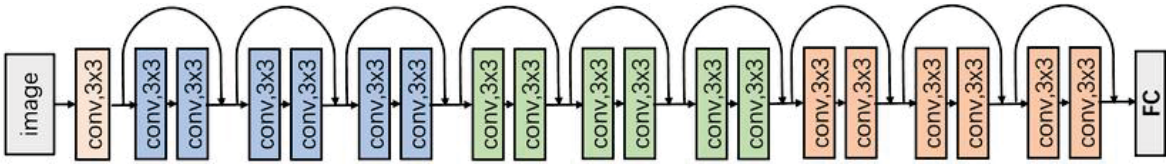
(a)

(b)

FONTE: Adaptado de (a) He *et al.* (2015a) e (b) Prince (2023).

LEGENDA: BN: são as camadas de normalização; ReLU: função de ativação; Conv: camada de convolução.

FIGURA 29 – ARQUITETURA RESNET-20



FONTE: Adaptado de Chen *et al.* (2022).

3.5 MÉTRICA DE AVALIAÇÃO

3.5.1 Acurácia

A métrica utilizada para avaliar o conjunto teste foi a precisão categórica, ou seja, a acurácia. Essa métrica cria duas variáveis, uma que conta o total de exemplares no conjunto e outra que conta a quantidade de elementos que o modelo previu corretamente. A operação efetuada é a divisão da quantidade prevista corretamente pelo total de exemplares do conjunto.

3.5.2 Esparsidade e operações em pontos flutuantes

A esparsidade é uma das métricas presentes quando se avaliam algoritmos de poda. Ela mede a proporção de elementos podados em uma RNA. Ou seja, dada uma arquitetura com n elementos treináveis a esparsidade indica a quantidade de elementos que adquiriam o valor igual a 0 ao longo das iterações de poda. O cálculo realizado é o número de elementos podados dividido pelo total de parâmetros na rede, o resultado é um valor percentual.

A outra métrica é o FLOP que avalia a quantidade de operações em pontos flutuantes realizadas pela RNA. Essa métrica é utilizada para comparação de eficiência computacional. O cálculo envolve contabilizar cada operação de multiplicação, soma, subtração ou divisão.

Para as camadas de convolução os FLOPs são calculados como

$$FLOPs = 2HW(C_{in}K^2 + 1)C_{out} \tag{31}$$

onde, H, W, e C_{in} são a altura, largura e número de filtros, K é o tamanho do kernel (se simétrico) e C_{out} são o número de filtros do mapa de característica (Molchanov ***et al.***, 2016). Já para a camada totalmente conectada o FLOP é calculado como:

$$FLOPs = (2I - 1)O \tag{32}$$

onde, I é o tamanho da entrada e O é o tamanho da saída (Molchanov ***et al.***, 2016).

3.6 CONFIGURAÇÕES DE TREINAMENTO PARA O MÉTODO PROPOSTO

As configurações de treinamento das CNNs - ou seja, a escolha dos hiperparâmetros de acordo com a arquitetura e os respectivos conjuntos de dados - foram estabelecidas pelos dados da literatura, garantido uma aderência metodológica e comparabilidade com publicações de referência. Dessa forma, os hiperparâmetros descritos abaixo foram escolhidos manualmente, por meio do processo de tentativas, avaliando em quais combinações se obteve o melhor o melhor desempenho.

Para o treinamento das ResNets 20/56/110 foi utilizada uma batelada de tamanho 128, com 180 épocas de treinamento, com taxa de aprendizagem inicial de

0,1 com decaimento (multiplicação) de 0,1 nas épocas 100 e 150, SGD com momentum de 0.9 e entropia cruzada para função perda. De forma complementar, no conjunto de dados foram usadas as transformações de espelhamento, translação e corte aleatório.

Para a arquitetura VGG-16 foi utilizada uma batelada de tamanho 128, com 165 épocas de treinamento, com taxa de aprendizagem inicial de 0,1 e decaimento de 0,00001 a cada 20 épocas, SGD com momentum de 0.9 e entropia cruzada categórica para função perda. De forma complementar, no conjunto de dados foram usadas as transformações de espelhamento e translação.

Para a arquitetura LeNet-5 foi utilizada uma batelada de tamanho 128, com 120 épocas de treinamento, com taxa de aprendizagem inicial de 0,0002, otimizador ADAM com $\beta_1, \beta_2, \epsilon$ iguais a 0,9, 0,999 e $1e^{-7}$, respectivamente. O conjunto de dados foi utilizado no seu formato original sem *data augmentation.*

Para as arquiteturas CNN3, AlexNet e VGG-19 foi utilizada uma batelada de tamanho 128, com 120 épocas de treinamento com *early stopping* de 3 épocas, com taxa de aprendizagem inicial de 0,0001, otimizador SGD com momentum de 0,8. Os conjuntos de dados utilizados foram fracionados dos conjuntos de dados originais. No conjunto de dados MNIST foram utilizados para o treinamento 18 mil exemplares distribuídos uniformemente entre as classes e para o conjunto de teste 3 mil exemplares. Para o conjunto de dados CIFAR-10, o treinamento utilizou 15 mil exemplares e para a teste 2500 exemplares, ambos divididos de forma equalitária entre as classes.

As FAs usadas foram ELU para a arquitetura ResNet, ReLU para VGG-16 e Tanh para a LeNet-5. As arquiteturas CNN3, AlexNet e VGG-19 foram utilizadas com várias FAs como será mostrado no experimento no capítulo seguinte.

Os hiperparâmetros número de filtros, *stride* e *padding* de cada camada de convolução das arquiteturas acima seguiram os modelos da literatura.

4 MÉTODO PROPOSTO

4.1 PODA ITERATIVA COM CRITÉRIO ESTOCÁSTICO

O método proposto nesta tese utiliza três conceitos: a magnitude dos pesos, o comportamento desses pesos e a aleatoriedade.

Conforme apresentado no trabalho de Frankle e Carbin (2019), no critério de magnitude, o corte é realizado a uma taxa $p\%$ naqueles pesos que possuem a menor magnitude a cada rodada até que se chegue ao nível de esparsidade desejado. Dessa forma, o corte consiste em atribuir o valor zero para o peso que atendeu o critério de poda.

Posteriormente, Zhou *et al.* (2019) mostraram que os pesos com menor magnitude tendem a reduzir ainda mais seus valores durante o processo de treinamento da rede, não exercendo influência no resultado, ou seja, aqueles valores que estão próximos de zero tendem a se tornar zero ao longo do treinamento da rede.

E assim como a técnica *dropout*, que possui o fator de aleatoriedade em sua escolha de poda, o método proposto introduz a aleatoriedade em seu algoritmo iterativo para achar uma rede super esparsa.

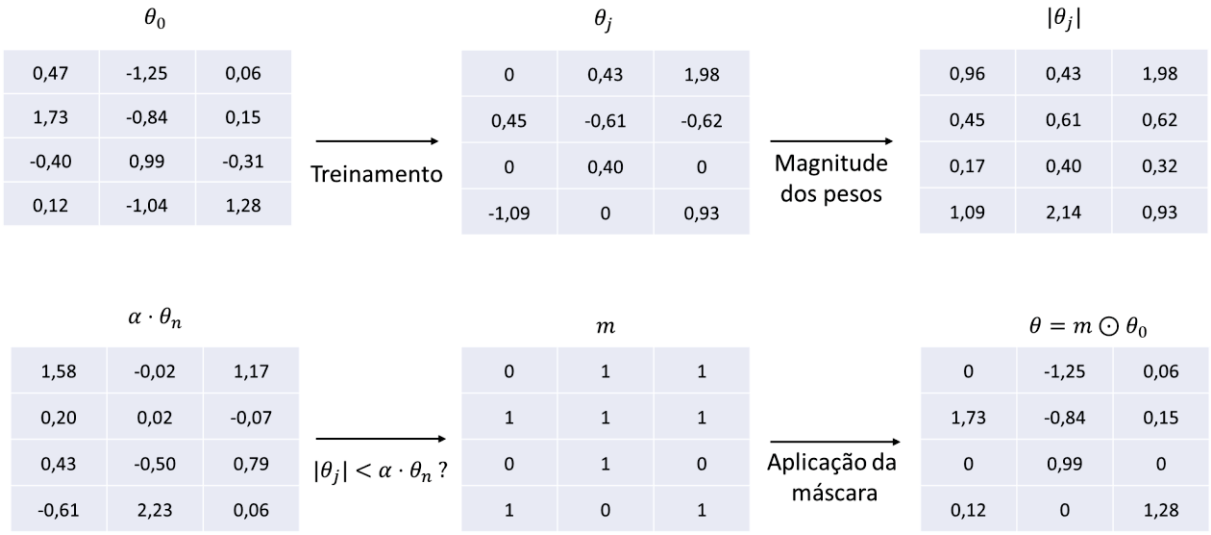
Dessa forma o critério proposto de poda, ou o critério gerador de máscaras para a rede, tem a seguinte configuração: para cada camada da rede e para cada peso dessas camadas são gerados valores aleatórios amortecidos $\theta_{aux} = \alpha * \theta_n$, onde $\theta_n \sim N(0,1)$, $\alpha \in \mathbb{Q}^{+*}$ e $\alpha < 1$. Se θ_{aux} for maior que o peso ao qual é comparado $|\theta_j|$, então o peso inicial θ_0 ($\theta_0 \sim \mathcal{D}$, onde $\mathcal{D}$ é gerado por He *et al.*, 2015b) é podado ($\theta_0 = 0$); isto é, os pesos utilizados para a próxima rodada do algoritmo serão $\theta = m \odot \theta_0$, onde m é a máscara binária gerada entre a comparação dos pesos $|\theta_j|$ e θ_{aux}. Assim, $\mathcal{N}'(x, \theta)$ é a rede esparsa procurada, ou o candidato ao bilhete premiado, gerado pela poda. A FIGURA 30 mostra o processo do critério de escolha da poda.

O critério proposto, ao utilizar a distribuição normal para gerar os pesos aleatórios aproveitamos o seu comportamento natural, no qual existe maior frequência de valores em torno da média, assim tendo maior probabilidade de serem podados. Ou seja, os pesos podados tem menor magnitude.

O algoritmo proposto, apesar de podar todas as camadas, é um algoritmo de poda local. De forma que, o parâmetro α é adicionado como forma reguladora da

velocidade de poda. Caso alguma camada sofra de estagnação nos cortes dos seus pesos, o valor de α pode ser ajustado, permitindo que o algoritmo continue podando ao longo das rodadas; ou de forma contrária, caso a camada seja podada de forma acelerada α serve para reduzir essa velocidade. Dessa forma, se necessário, é possível o ajuste personalizado para cada camada, uma vez que as camadas nas CNNs possuem quantidades variadas de pesos e impactam diretamente no resultado da rede.

FIGURA 30 – MÉTODO DE PODA ESTOCÁSTICO



FONTE: O autor (2025).

O critério apresentado será referido como Poda Estocástica (PE). O algoritmo é descrito no QUADRO 2.

QUADRO 2 – ALGORITMO DE PODA ITERATIVA COM CRITÉRIO ESTOCÁSTICO

1. Iniciar a rede neural $\mathcal{N}(x, \theta_0)$ aleatoriamente. ($\theta_0 \sim \mathcal{D}$);
2. Treinar a rede por j iterações, chegando aos parâmetros θ_j;
3. Para cada peso de cada filtro, criar um peso auxiliar $\theta_{aux} = \alpha \cdot \theta_n$, $\theta_n \sim N(0,1)$;
4. Comparar os pesos θ_{aux} e $|\theta_j|$. Se $\theta_{aux} > |\theta_j|$, então $\theta_0 = 0$, criando a máscara binária m;
5. Reiniciar os parâmetros que não foram podados para seus valores originais θ_0, criando um bilhete premiado $\mathcal{N}'(x, \theta)$, onde $\theta = m \odot \theta_0$;
6. Atualizar o amortecimento α se necessário;
7. Repetir os passos 2 – 6, até chegar ao nível de esparsidade desejada.

Fonte: O autor (2025).

Comparativamente, a utilização de um limiar para critério de corte é aplicado com sucesso em Frankle e Carbin (2019) e Han *et al.* (2015), porém, ambos os critérios não têm uma flexibilidade adaptativa. Enquanto Han *et al.* (2015) utiliza um múltiplo do desvio padrão como critério de poda para todas as camadas, Frankle e Carbin (2019) podam $p\%$ dos menores valores absolutos. O método PE traz um critério de poda menos determinístico ao longo das iterações. Diferentemente da abordagem do bilhete premiado que remove uma porcentagem fixa dos pesos em todas as iterações do algoritmo, na PE a quantidade de pesos podados varia de acordo com o resultado da comparação feita entre os pesos da técnica. Além disso, existe um amortecimento α adaptável para cada camada possibilitando um controle refinado da taxa de poda. Ainda, a geração de pesos auxiliares, seguindo a distribuição normal, aumenta a probabilidade de poda de pesos próximos à média. Essa abordagem também incorpora a ideia de podar elementos que teriam valores absolutos similares, ou seja, aqueles pesos que não contribuem de forma significante para a rede.

4.1.1 Resultados investigativos da poda iterativa com critério estocástico

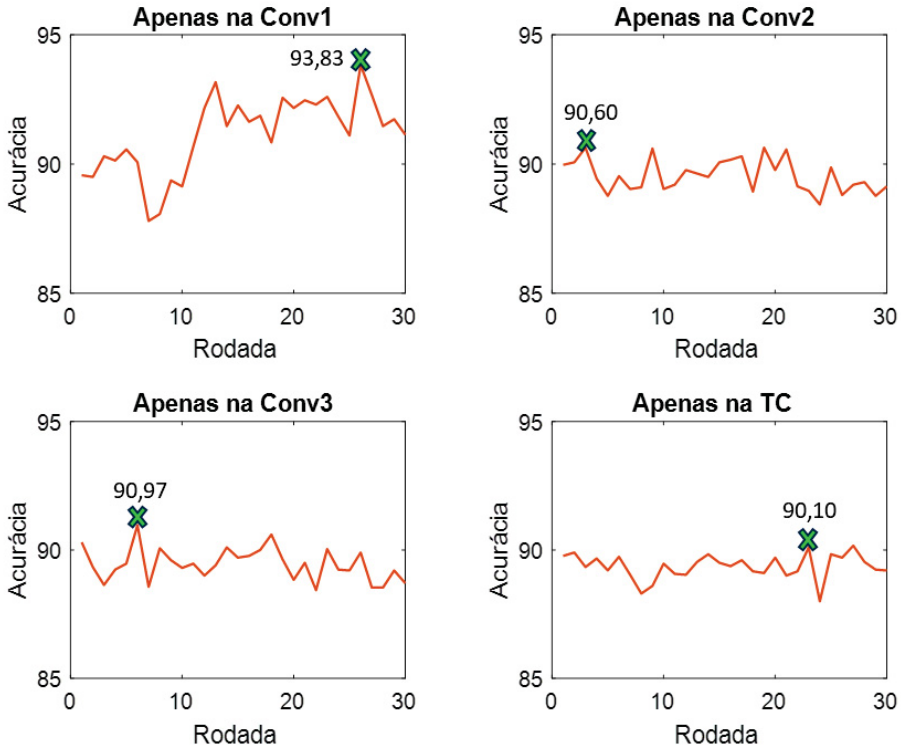
A seguir é analisado o comportamento do algoritmo de PE, para o experimento foi utilizada arquitetura CNN3 com as configurações definidas na Seção 3.6. Uma vez que o algoritmo proposto poda todas as camadas, o objetivo é examinar como o desempenho da rede é afetado ao alterar o local de poda. A análise é dividida em dois casos. O primeiro, quando a poda é aplicada em apenas uma camada específica – sinalizado como ‘Apenas na camada #’. O segundo, quando a poda é aplicada em toda a rede com exceção de uma única camada – sinalizada como ‘sem poda # camada’. Adicionalmente, a melhor performance ao longo das iterações é sinalizada com um “x”, acompanhada do valor numérico respectivo.

Os valores de amortecimento α utilizados nos experimentos dos gráficos a seguir foram definidos empiricamente. Especificamente, atribuiu-se os valores de 0,8 à primeira camada de convolução, para a segunda e terceira o valor α inicial foi de 0,04 até a 20ª rodada e, após isso, 0,03. Para as camadas TC o valor α foi de 0,02 em todas as rodadas.

O GRÁFICO 1 mostra o comportamento da rede ao utilizar o critério PE em apenas uma camada de cada vez, o conjunto de dados empregado foi o MNIST. É

possível observar que a rede conseguiu chegar a uma acurácia maior quando a poda foi aplicada apenas na primeira camada de convolução; quando foi aplicada isoladamente nas outras camadas o comportamento das redes foi semelhante, de modo que chegaram a valores próximos na sua acurácia máxima e não tiveram grande aumento na acurácia durante as rodadas.

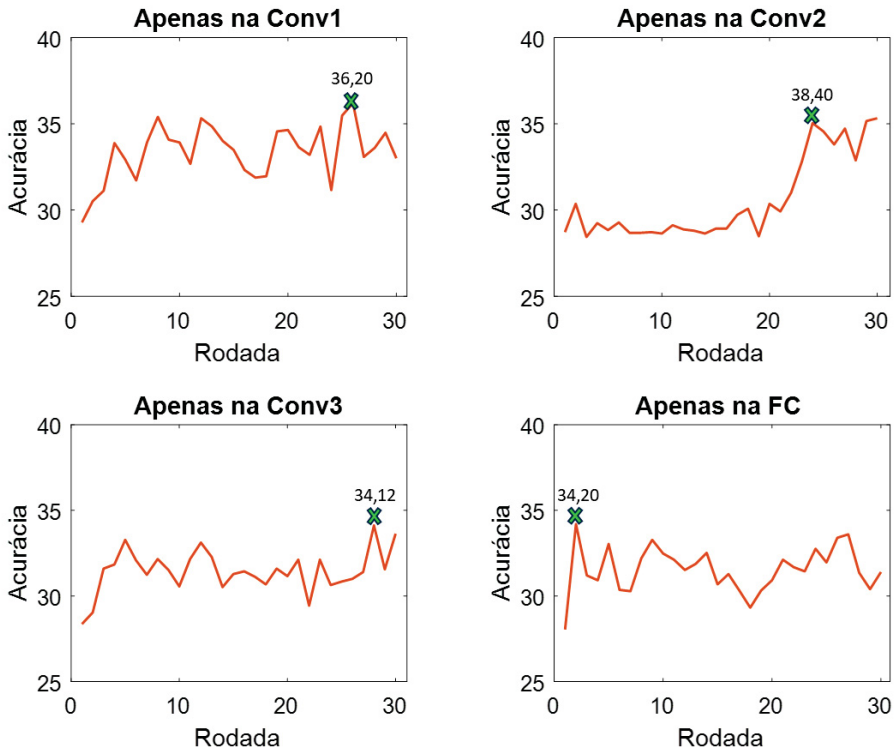
GRÁFICO 1 – CRITÉRIO DE PODA EM UMA ÚNICA CAMADA – CONJUNTO DE DADOS MNIST



FONTE: O autor (2025).

O mesmo tipo de análise foi realizado para o conjunto de dados CIFAR-10, como mostra o GRÁFICO 2. O comportamento das redes entre si é semelhante em quase todos os casos, conseguindo aumentar a acurácia logo no início das rodadas do algoritmo. A exceção é na segunda camada de convolução, onde a rede ficou mais estável e só aumenta a acurácia nas últimas rodadas. A maior acurácia, entre as quatro camadas, foi atingida quando o método foi aplicado apenas na primeira camada de convolução.

GRÁFICO 2 – CRITÉRIO DE PODA EM UMA ÚNICA CAMADA – CONJUNTO DE DADOS CIFAR-10



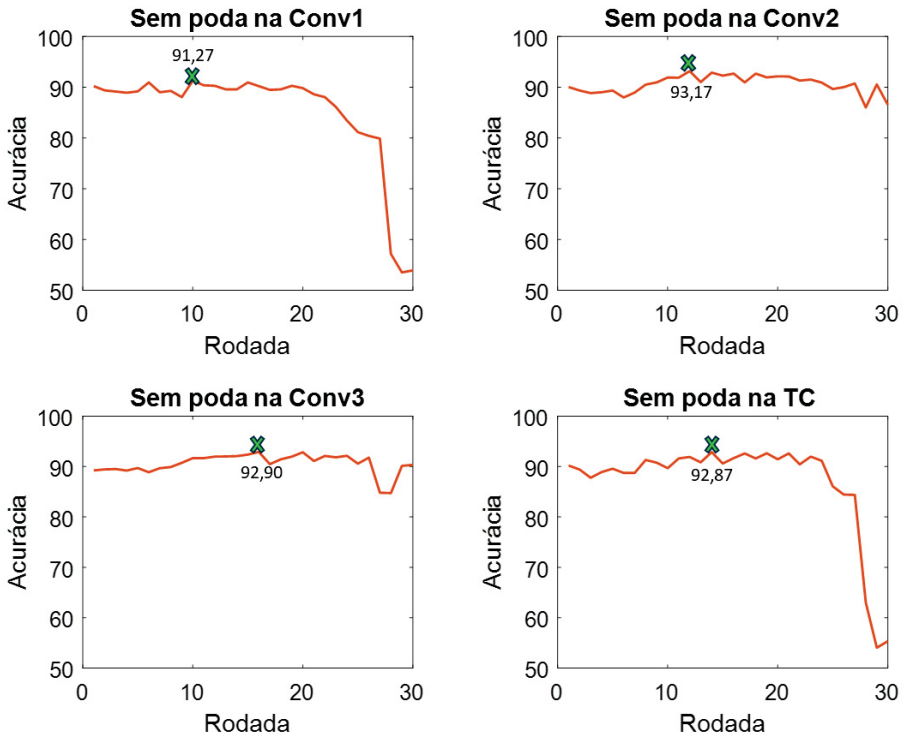
FONTE: O autor (2025).

O resultado do desempenho do algoritmo da PE, quando é retirada a poda de apenas uma camada da rede, é exibido no GRÁFICO 3 para o conjunto de dados MNIST e no GRÁFICO 4 para o conjunto de dados CIFAR-10.

Para os dois conjuntos de dados a rede atingiu uma taxa de acurácia menor quando não foi aplicada a técnica de poda na primeira camada de convolução. A rede se comportou de forma semelhante quando foi retirada a poda da Conv2 e Conv3, ou seja, houve o aumento da acurácia ao longo das rodadas do algoritmo. Esse comportamento é observado em ambos os conjuntos de dados.

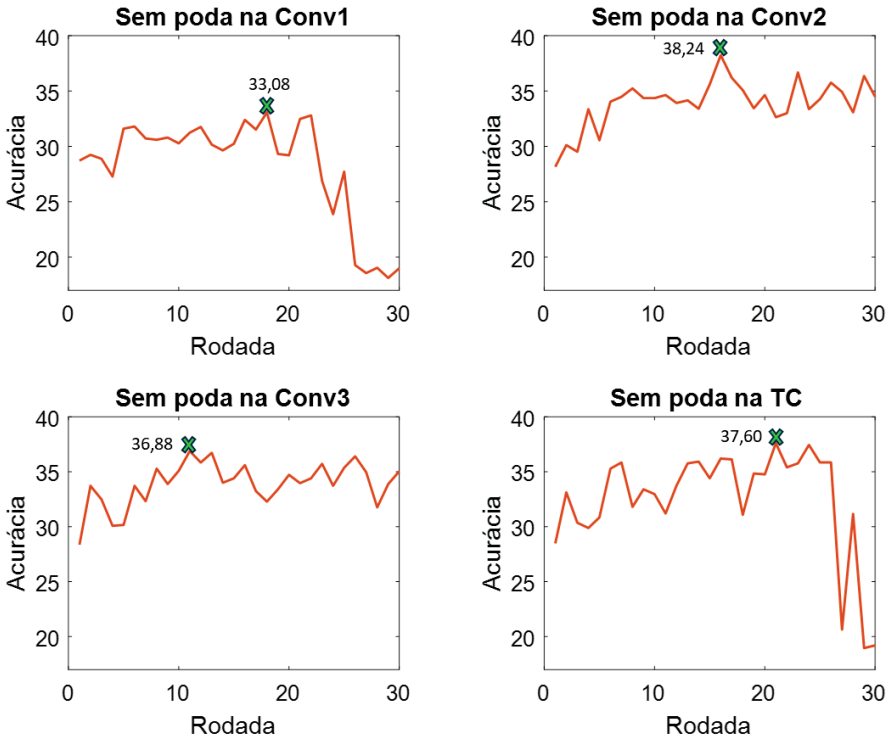
Na ausência da poda na camada TC para os dados MNIST, a rede não teve um aumento expressivo na acurácia, como é visto nas camadas Conv2 e Conv3, no final das rodadas a acurácia tende a ficar menor. De modo contrário, para o conjunto de dados CIFAR-10, a rede conseguiu aumentar a acurácia mesmo sem a poda na camada TC.

GRÁFICO 3 – CRITÉRIO DE PODA RETIRANDO UMA ÚNICA CAMADA – CONJUNTO DE DADOS MNIST



FONTE: O autor (2025).

GRÁFICO 4 – CRITÉRIO DE PODA RETIRANDO UMA ÚNICA CAMADA – CONJUNTO DE DADOS CIFAR-10



FONTE: O autor (2025).

Como resultado dessa análise, é observável que a presença da poda na primeira camada de convolução tem um papel fundamental para o desempenho da rede. Essa poda faz com que a rede tenha um salto maior na acurácia durante as rodadas do algoritmo.

4.2 CAMADA NULA

Com o resultado do critério PE, observou-se que ao aumentar a esparsidade da primeira camada da rede pode-se gerar um impacto positivo na performance. Logo, para explorar essa característica foi proposto o método da Camada Nula (CN).

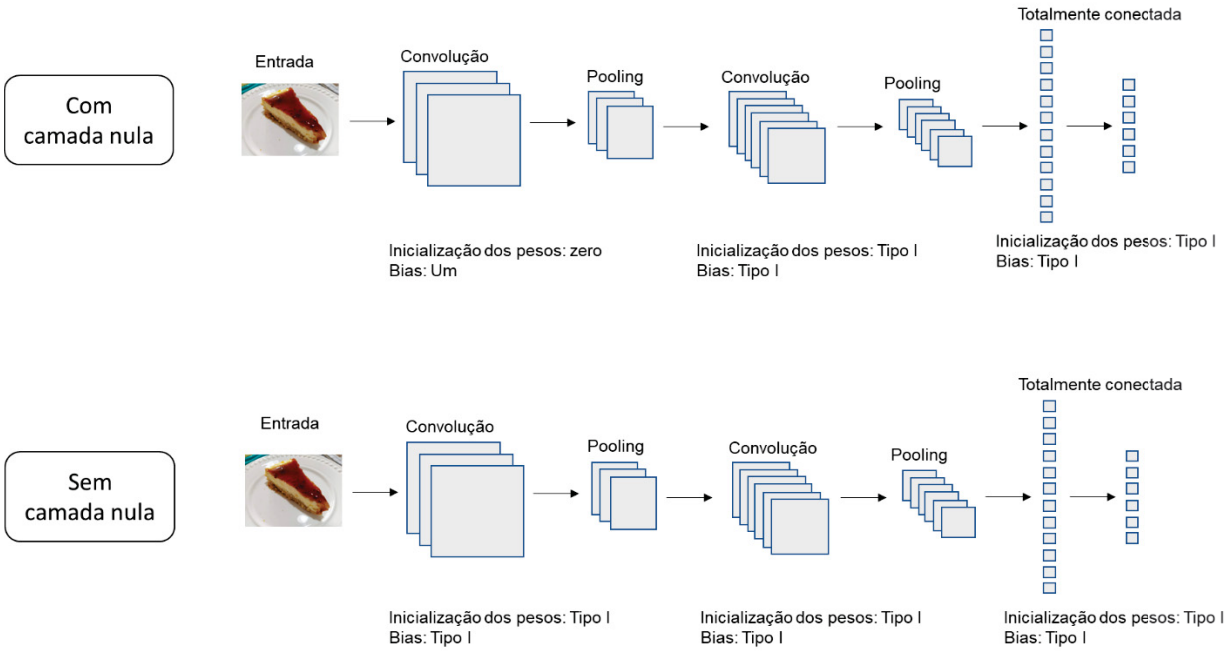
4.2.1 Hipótese proposta

O método de inicialização proposto é chamado de método da camada nula (CN). O método impõe que os pesos iniciais da primeira camada de convolução sejam nulos, ou seja, os pesos gerados para essa primeira camada serão iguais a zero e o viés da camada será igual a 1; os pesos iniciais das demais camadas devem seguir outro tipo de inicialização.

A forma tradicional, de iniciar os pesos das camadas de convolução é a inicialização com um único tipo de distribuição ou procedimento, que é aplicado para todas as camadas. A ilustração de uma CNN com a inicialização comum e o método CN é mostrada na FIGURA 31.

Essa hipótese fundamenta-se nas evidências do critério PE e, de modo análogo à regra de Hebb (1949), ao inicializar os pesos da primeira camada com valor igual a zero e viés igual a um, confere-se à rede um ponto de partida neutro, no qual não existe tendência prévia que possa privilegiar qualquer padrão de entrada em detrimento de outro. E ainda, a CN permite um refinamento adicional dos parâmetros iniciais, funcionando como um mecanismo de reforço das representações de características iniciais, de modo a reforçar a extração de padrões relevantes durante o processo de treinamento. Ademias, a inicialização dos pesos igual a zero e do viés com valor unitário garante ativações positivas com as FA, evitando a possível “morte” de neurônios e garantindo a não interrupção da propagação do gradiente na primeira camada.

FIGURA 31 – MÉTODO CAMADA NULA



FONTE: O autor (2025).

4.2.2 Resultados

O resultado do experimento é exibido na TABELA 3. Os valores numéricos apresentados são a média aritmética entre 5 ensaios realizados com as mesmas especificações de treinamento, variando os pesos iniciais. A avaliação das redes foi realizada nos conjuntos de teste, e as configurações utilizadas para cada arquitetura correspondem às especificadas na Seção 3.6.

A TABELA 3 mostra que para todos os testes o método CN obteve acurácia maior em 53,89% dos casos em relação ao treinamento regular. E, ainda, a diferença de acurácia quando o método tradicional é superior ao método CN, é em média 3,46% maior. De modo contrário, quando o método CN é superior ao método tradicional, em média atinge 7,90% a mais de acurácia. O empate dos dois métodos ocorreu apenas em 15,56% dos experimentos.

Analisando os resultados do método em relação à FA (inicialização de peso, arquitetura e conjunto de dados não foram considerados), o melhor desempenho para a CN foi alcançado com a FA Tanh. Com essa FA, o método CN teve acurácia superior em 75% dos experimentos, o método tradicional obteve apenas 13,89% dos melhores resultados e o empate teve uma taxa de 11,11%. O método CN obteve um

desempenho superior para todas as FA. Exceto a FA ELU, onde ocorreu o empate entre os métodos comparados. Os dados são mostrados na TABELA 4.

TABELA 3 – ACURÁCIA MÉDIA PARA OS TESTES DA CAMADA NULA

			Tanh	Tanh + CN	CReLU	CReLU + CN	ReLU	ReLU + CN	ELU	ELU + CN	LReLU	LReLU + CN
CNN3	Mnist	Glorot	0,7269	**0,7994**	0,6057	**0,8071**	0,4919	**0,6735**	0,6104	**0,9195**	0,8807	**0,9146**
		He	0,7104	**0,8485**	0,4432	**0,8674**	0,0980	**0,8477**	0,098	**0,9203**	0,7970	**0,8952**
		Narrow-normal	0,4641	**0,6004**	0,4395	**0,4716**	**0,6903**	0,6104	**0,8439**	0,8345	0,6656	**0,6823**
	Cifar-10	Glorot	0,2484	**0,2813**	0,2308	**0,3471**	0,1000	**0,3639**	0,2086	**0,3766**	0,1946	**0,3667**
		He	0,2335	**0,2872**	0,1243	**0,2770**	0,1000	**0,2833**	0,1000	**0,3289**	0,2469	**0,2939**
		Narrow-normal	0,2041	**0,2085**	0,1875	**0,2145**	**0,2760**	0,2641	**0,3191**	0,3030	**0,3001**	0,2731
	Flower	Glorot	0,4991	**0,5074**	0,1787	**0,1833**	0,1759	0,1759	0,1759	**0,1898**	0,0000	**0,4593**
		He	**0,4602**	0,4343	0,1898	0,1898	0,1759	0,1759	0,1759	0,1759	0,0000	0,0000
		Narrow-normal	0,4907	**0,5019**	0,2815	**0,4315**	0,1769	**0,2611**	0,2630	**0,4028**	**0,4407**	0,3926
	Fruit	Glorot	0,9579	**0,9755**	0,6312	**0,7253**	0,1063	**0,5175**	0,0354	**0,9807**	0,3950	**0,9995**
		He	0,9353	**0,9636**	**0,5015**	0,4366	0,0297	0,0297	0,0297	0,0297	0,0000	0,0000
		Narrow-normal	0,8785	**0,8811**	0,6977	**0,7092**	0,7122	**0,8307**	**0,9948**	0,9945	**0,9998**	0,9995
AlexNet	Mnist	Glorot	0,8653	**0,8740**	0,8849	**0,8953**	**0,9432**	0,9320	**0,9543**	0,9514	**0,9408**	0,9331
		He	0,8586	**0,8748**	0,9113	**0,9170**	0,8536	**0,9419**	0,9306	**0,9560**	0,8423	**0,9417**
		Narrow-normal	0,5005	**0,5051**	0,2045	**0,2619**	**0,7072**	0,3389	**0,9082**	0,8959	**0,7393**	0,4427
	Cifar-10	Glorot	0,3489	**0,3695**	**0,3208**	0,3183	**0,3472**	0,3244	**0,4530**	0,4226	**0,3551**	0,3198
		He	0,2480	**0,3497**	**0,3378**	0,3252	**0,2693**	0,2937	**0,3926**	0,3843	0,2813	**0,3036**
		Narrow-normal	0,1753	**0,2318**	0,1378	**0,1730**	**0,1613**	0,1286	**0,3129**	0,2906	0,1474	**0,1633**
	Flower	Glorot	0,5750	**0,5972**	0,5731	**0,6176**	0,6528	**0,6556**	**0,7065**	0,7056	0,6731	**0,6759**
		He	0,5815	0,5815	0,5500	**0,5694**	0,3426	**0,5602**	0,4324	**0,6630**	0,4565	**0,5861**
		Narrow-normal	0,3852	**0,4204**	0,2741	**0,3065**	**0,4880**	0,3574	**0,6620**	0,6111	**0,5102**	0,4102
	Fruit	Glorot	0,9572	**0,9600**	0,9612	**0,9696**	**1,0000**	0,9995	**1,0000**	0,9995	**1,0000**	0,9995
		He	0,9867	**0,9883**	**0,9879**	0,9850	0,9919	**0,9995**	0,9988	**1,0000**	0,9960	**1,0000**
		Narrow-normal	0,0623	**0,0989**	0,0583	**0,0585**	0,0678	**0,0735**	**0,9413**	0,9191	0,2271	**0,0585**
VGG19	Mnist	Glorot	**0,9112**	0,9093	**0,16388**	0,1177	**0,9579**	0,1300	**0,9667**	0,9573	**0,9373**	0,1364
		He	0,9387	**0,9410**	0,9423	**0,9452**	0,2432	**0,9617**	0,0421	**0,9702**	0,2587	**0,9532**
		Narrow-normal	0,1135	0,1135	0,1135	0,1135	0,1135	0,1135	0,1135	0,1135	0,1135	0,1135
	Cifar-10	Glorot	0,4035	**0,4273**	**0,1586**	0,1182	**0,2972**	0,1436	**0,4926**	0,4629	**0,1666**	0,1555
		He	**0,4327**	0,4312	**0,3527**	0,3498	0,1640	**0,3410**	0,2497	**0,4490**	0,2270	**0,3311**
		Narrow-normal	0,0991	**0,1000**	0,1000	0,1000	0,1000	0,1000	0,1003	0,1000	0,1000	0,1000
	Flower	Glorot	0,6101	**0,6416**	**0,5120**	0,2453	**0,5046**	0,2453	0,6722	**0,6870**	**0,4796**	0,2416
		He	0,6212	**0,6379**	0,1962	**0,6074**	0,2064	**0,6324**	0,0851	**0,6500**	0,0842	**0,6185**
		Narrow-normal	0,2453	0,2453	0,2453	0,2453	0,2453	0,2453	0,2453	0,2453	0,2453	0,2453
	Fruit	Glorot	**0,9980**	0,9895	0,0801	**0,0582**	**0,9976**	0,0582	1,000	1,000	**0,9973**	0,0582
		He	**1,000**	0,9997	**1,000**	0,9997	0,8675	**0,9997**	0,9940	**1,000**	0,9422	**0,9980**
		Narrow-normal	0,0582	0,0582	0,0582	0,0582	0,0582	0,0582	0,0582	0,0582	0,0582	0,0582

FONTE: O autor (2025).

TABELA 4 – CN (MÉTODO VS. FA).

%	Tanh	CReLU	ReLU	ELU	LReLU
Normal	13,89	27,78	33,33	41,67	36,11
CN	**75,00**	**58,33**	**47,22**	41,67	**47,22**
Empate	11,11	13,89	19,44	16,67	16,67

FONTE: O autor (2025).

Da mesma forma, para a comparação dos métodos em relação à inicialização de peso, mostrados na TABELA 5, a inicialização que obteve os melhores resultados com o método CN foi a inicialização He, que obteve 71,67% dos melhores resultados; a inicialização Glorot conquistou 51,67% dos melhores resultados; e por último, a inicialização *Narrow-normal* foi melhor em 38,33% dos experimentos. O método CN foi superior ao método tradicional para todas as inicializações.

TABELA 5 – CN (MÉTODO VS. INICIALIZAÇÃO DOS PESOS).

%	Glorot	He	Narrow-normal
Normal	45,00	15,00	31,67
CN	**51,67**	**71,67**	**38,33**
Empate	3,33	13,33	30,00

FONTE: O autor (2025).

A TABELA 6 mostra a porcentagem obtida para cada método em comparação com o conjunto de dados (FA, arquitetura e inicialização de peso não foram consideradas). O método CN obteve a maior porcentagem de melhores resultados para todos os conjuntos de dados.

TABELA 6 – CN (MÉTODO VS. CONJUNTO DE DADOS)

%	Mnist	Cifar-10	Flower	Fruit
Normal	28,89	40,00	20,00	33,33
CN	**60,00**	**53,33**	**55,56**	**46,67**
Empate	11,11	6,67	24,44	20,00

FONTE: O autor (2025).

A última análise é o desempenho dos métodos em relação às arquiteturas (FA, conjuntos de dados e tipo de inicialização de peso não foram considerados). O método tradicional apresentou resultados melhores apenas para a arquitetura VGG19, conforme mostrado na TABELA 7. Dessa forma, pode-se inferir que, à medida que as CNNs se tornam mais profundas, há maior poder para extrair características das imagens, e o método CN perde seu desempenho.

TABELA 7 – CN (MÉTODO VS. ARQUITETURA)

%	CNN3	AlexNet	VGG19
Normal	16,67	40,00	**35,00**
CN	**70,00**	**58,33**	33,33
Empate	13,33	01,67	31,67

FONTE: O autor (2025).

Observou-se, também, que na arquitetura VGG19, quando utilizada a inicialização de pesos *Narrow-normal,* o modelo permaneceu estagnado sem conseguir realizar nenhum aprendizado ao longo das iterações de treinamento, independente dos hiperparâmetros (FA, método e conjunto de dados) utilizados — o que sugere uma sensibilidade às condições experimentais testadas.

Dados adicionais, como o desvio padrão dos 5 ensaios realizados para os resultados médios da CN (TABELA 3), é mostrado no APÊNDICE 1 – RESULTADOS COMPLEMENTARES DA CAMADA NULA. Como resultado, a média do desvio padrão do método CN foi de 0,022, enquanto a média do método tradicional foi de 0,041. O método CN obteve uma redução de 46,34% no desvio padrão médio, implicando em um método com menos variabilidade e maior estabilidade para o treinamento das redes.

Os valores máximos e mínimos dos ensaios também é mostra no APÊNDICE 1 – RESULTADOS COMPLEMENTARES DA CAMADA NULA.

4.3 PODA ESTOCÁSTICA COM CAMADA NULA

Com base nos resultados anteriores, isto é, os experimentos realizados com o critério PE e com o método CN, é proposto um ajuste para o método desenvolvido. É realizada a fusão da PE com a CN, que será chamada de Poda Estocástica com Camada Nula (PECN). O algoritmo segue o critério de seleção da PE e a inicialização dos pesos é realizada como na CN, o algoritmo PECN é detalhado no quadro abaixo.

No algoritmo PECN, por meio de experimentação, valores distintos de α foram definidos ao longo das épocas de treinamento e conforme o tipo de camada. Essa diferenciação foi necessária pois o comportamento das camadas e a velocidade de poda variam entre si. Assim, a variação do valor α garante que a rede atinja uma proporção de poda uniforme ao longo das iterações e maximize o seu melhor desempenho.

QUADRO 3 – ALGORITMO PECN

1. Aleatoriamente inicializar a rede neural $\mathcal{N}(x, \theta_0)$ com $\theta_0 \sim \mathcal{D}$;
2. Faça os pesos da primeira camada iguais a 0 e bias iguais a 1 (método CN);
3. Treinar a rede por j iterações, chegando aos parâmetros θ_j;
4. Para cada peso de cada filtro, criar um peso auxiliar $\theta_{aux} = \alpha \cdot \theta_n$, $\theta_n \sim N(0,1)$;
5. Para todas as camadas menos a primeira, comparar os pesos θ_{aux} e $|\theta_j|$. Se $\theta_{aux} > |\theta_j|$, então $\theta_0 = 0$, criando a máscara binária m;
6. Reiniciar os pesos da primeira camada e parâmetros das demais camadas que não foram podados para seus valores originais θ_0, criando uma rede esparsa $\mathcal{N}'(x, \theta)$, onde $\theta = m \odot \theta_0$;
7. Atualizar o amortecimento α se necessário;
8. Repetir os passos 3 – 7, até chegar ao nível de esparsidade desejada.

Fonte: O autor (2025).

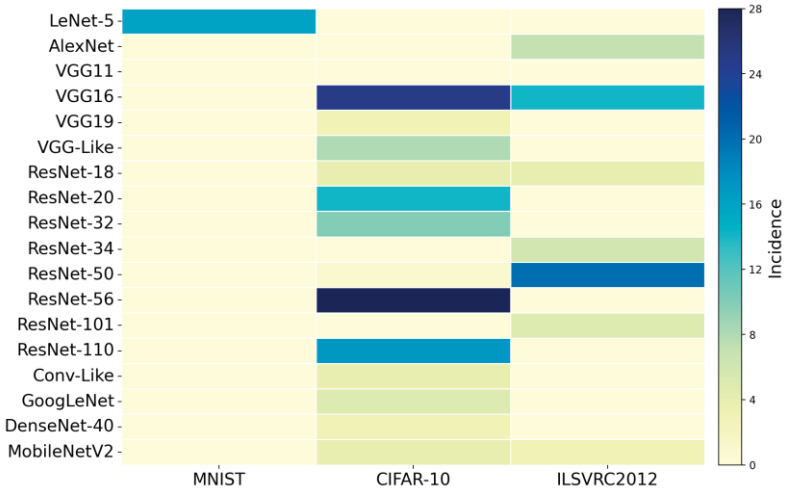
Para o viés foi utilizado um valor α de 0,8 para todos os tipos de camada. Para as camadas TC o valor de α inicial foi de 0,007, aumentando para 0,027 entre as rodadas 10 e 19, e terminando com o valor de 0,01 para as demais rodadas. Para as camadas Conv, o α inicial foi de 0,02, aumentando para 0,04 entre as rodadas 10 a 19, e depois disso 0,09. E para as arquiteturas as configurações utilizadas correspondem às especificadas na Seção 3.6.

O resultado do algoritmo PECN e a comparação com o método da literatura é exposto no capítulo seguinte.

5 RESULTADOS E DISCUSSÃO

Nesse capítulo é realizada uma análise dos resultados do algoritmo PECN. Os resultados apresentados utilizaram os conjuntos de dados MNIST e CIFAR-10, assim como as arquiteturas LeNet-5, VGG-16, ResNet-20/56/110. A escolha dos conjuntos de dados e arquiteturas baseou-se em um levantamento de 90 técnicas de poda presentes na literatura. Como isso, foram identificados os conjuntos de dados e arquiteturas mais utilizadas assim como a combinação mais frequente entre eles, conforme é ilustrado na FIGURA 32.

FIGURA 32 – COMBINAÇÃO DE CONJUNTO DE DADOS E ARQUITETURA



FONTE: O autor (2025).

5.1 RESULTADOS

A TABELA 8 apresenta o resultado comparativo entre a técnica proposta e os resultados da literatura com o conjunto de dados MNIST na arquitetura LeNet-5. Apesar do algoritmo PECN não ter atingido a maior acurácia, ele foi o único que apresentou um ganho na acurácia com o aumento da esparsidade. A quantidade de pesos podados apresentou um resultado competitivo com a literatura e a quantidade de FLOPs não apresentou vantagens.

TABELA 8 – RESULTADO COMPARATIVO LENET-5 / MNIST

Referência	Top-1 Acc	Acc.↓%	FLOPs↓%	Par.↓%
Han *et al*., 2015	99,23	-	-	-
Guo *et al*., 2016	99,09	-	-	-
Wu *et al*., 2019	99,26	-	-	62,50
Hu *et al*., 2016	**99,30**	-	-	-
Ding *et al*., 2018	-	0,76	48,63	95,13
G. Li *et a*l., 2018	99,09	-	-	-
Dong *et al*., 2017	98,34	-	-	-
Srinivas *et al*., 2017	99,19	-	-	95,84
Singh *et a*l., 2020	99,20	-	**95,56**	-
Yu *et a*l., 2018	-	0,02	-	50,00
Lee *et al*., 2018	99,20	-	-	**99,00**
Zhang *et al*., 2018	99,20	-	-	-
Geng; Niu, 2022	99,10	-	73,45	-
Oliveira *et al*., 2024	99,03	-	-	-
Shi *et al*., 2024	99,18	-	91,14	98,96
Autor, 2025	98,82	**+0,25**	57,85	93,58

FONTE: O autor (2025).
LEGENDA: [Top-1 Acc.]: Porcentagem de classificação correta da rede; [Acc.↓%]: Denota o decréscimo (%) na acurácia da rede com a rede podada; [FLOPs↓%]: denota a redução (%) de FLOPs em relação a rede não podada; [Par.↓%] denota a redução (%) dos parâmetros em relação a rede não podada.

Para o conjunto de dados CIFAR-10, o algoritmo PECN obteve resultados superiores em redução de FLOPs e parâmetros nas arquiteturas testadas, como mostram as TABELA 9, TABELA 10, TABELA 11 e TABELA 12. Além disso, para a arquitetura ResNet-56/110, mesmo o algoritmo não tendo aumentado a acurácia, foi obtida a maior acurácia entre os trabalhos comparados. De forma contrária, para a arquitetura VGG-16, apesar de obtida uma melhora no desempenho da rede, a acurácia final não ultrapassa o resultado da literatura.

TABELA 9 – RESULTADO COMPARATIVO VGG-16 / CIFAR-10

(continua)

Referência	Top-1 acc	Acc↓%	FLOPs↓%	Param↓%
Wu *et al*., 2019	93,13			
Zunino *et al*., 2021	92,73	-	-	-
He *et al*., 2017	67,80	-	-	-
Liu *et al*., 2019	87,66	-	70,47	-
Singh *et al*., 2020	93,60	-	82,80	92,50
Lin *et a*l., 2020	93,43	-	53,50	82,90
Ding *et al*., 2018	92,94	-	79,69	-
Huang *et a*l., 2020	93,98	-	48,50	-
Li *et al*., 2016	93,40	-	34,20	64,00
Lin *et a*l., 2020	93,08	-	73.68	88,68
Xu *et al*., 2021	93,83	-	-	-
Lee *et al*., 2018	92,91	-	-	-
Wang *et al*., 2020	87,13	-	-	-
Wang; Zhang, 2023	94,20	0,24	51,00	82,20

TABELA 9 – RESULTADO COMPARATIVO VGG-16 / CIFAR-10

(conclusão)

Referência	Top-1 acc	Acc↓%	FLOPs↓%	Param↓%
Hubens *et al*., 2022	87,48	-	-	-
Geng; Niu, 2022	-	0,19	91,20	88,47
Zhang *et al.*, 2023	93,88	0,04	55,20	90,86
Jiang *et al*., 2022	93,08	-	26,93	29,50
Oliveira *et al*., 2024	-	0,70	-	93,20
Zheng *et al*., 2024	93,48	0,35	63,50	-
Shi *et al*., 2023	**94,33**	**+0.37**	58,10	81,60
Huang *et al*., 2018	-	3,40	80,60	-
Lin *et al*., 2020	93,87	-	-	-
Yeom *et al*., 2021	93,42	-	-	-
Suau *et al*., 2020	-	+0,24	58,40	80,37
Autor, 2025	93,52	+0,10	**97,42**	**97,56**

FONTE: O autor (2025).

TABELA 10 – RESULTADO COMPARATIVO RESNET-20 / CIFAR-10

Referência	Top-1 Acc	Acc.↓%	FLOPs↓%	Par.↓%
Huang *et al*., 2020	91,19	-	50,00	48,10
Li *et al*., 2020	91,44	0,76	48,20	-
He *et al*., 2019	90,62	1,58	-	54,00
He *et al*., 2018	91,20	1,00	29,30	-
Li *et al*., 2021	91,13	1,07	48,20	-
Wang *et a*l., 2021	**92,48**	**0,21**	45,80	-
Ye *et a*l., 2018	-	1,00	-	37,00
Lin *et al*., 2020	88,01	-	-	95,00
Jiang *et a*l., 2022	91,58	-	49,46	61,49
Oliveira *et al*., 2024	-	1,00	-	80,00
Zhang *et al.*, 2023	91,05	1,25	54,38	-
Tessier *et al*., 2022	89,07	6,50	-	-
Autor, 2025	92,43	0,68	**96,00**	**96,69**

FONTE: O autor (2025).

TABELA 11 – RESULTADO COMPARATIVO RESNET-56 / CIFAR-10

(continua)

Referência	Top-1 Acc	Acc.↓%	FLOPs↓%	Par.↓%
Li *et al*., 2021	93,50	0,09	58,00	-
Suau *et al*., 2020	-	0,65	38,43	40,03
Chen *et al*., 2021	93,48	-	30,16	29,43
Li *et al.*, 2020	93,78	+0,19	47,00	
You *et al.*, 2019	-	0,33	60,10	53,50
Xu *et al*., 2020	-	0,16	54,60	53,90
Singh *et al*., 2020	93,09	-	68,40	-
Yu *et al*., 2018	-	0,03	43,61	42,60
Lin *et al*., 2020	93,52	-	29,30	16,80
Ding *et al.*, 2018	-	9,43	29,15	70,79

TABELA 11 – RESULTADO COMPARATIVO RESNET-56 / CIFAR-10

(conclusão)

Referência	Top-1 Acc	Acc.↓%	FLOPs↓%	Par.↓%
Huang *et al*., 2020	93,17	-	51,20	59,30
Lin *et al*., 2020	93,23	0,03	54,13	54,20
He *et al*., 2019	93,49	0,10	52,60	-
Li *et al.*, 2016	93,06	-	27,60	13,70
He *et al*., 2018	93,89	+0,30	14,70	-
Ding *et al*., 2019	93,44	0,23	60,85	-
Jiang *et al*., 2022	92,75	-	22,78	29,50
Zheng *et al.*, 2024	93,32	0,28	55,80	-
Wang *et al.*, 2021	93,75	0,37	53,80	-
Lin *et al.*, 2020	92,74	-	-	95,00
Shi *et al*., 2023	94,28	**+1,02**	28,00	22,30
Zhang *et al*., 2023	93,73	0,11	54,49	-
He, 2022	90,80	-	-	-
Yamamoto; Maeno, 2020	93,58	-	54,80	53,70
Autor, 2025	**94,80**	0,15	**95,22**	**96,75**

FONTE: O autor (2025).

TABELA 12 – RESULTADO COMPARATIVO RESNET-110 / CIFAR-10

Referência	Top-1 Acc	Acc.↓%	FLOPs↓%	Par.↓%
Li *et al*., 2021	93,77	0,09	46,70	-
Chen *et al*., 2021	94,11	-	30,07	29,11
Huang *et al.*, 2020	93,77	-	53,30	67,20
Li *et al*., 2020	93,96	+0,28	46,70	-
Lin *et al.*, 2020	93,58	+0,08	65,04	67,41
He *et al*., 2019	93,85	+0,17	-	52,30
He *et al.*, 2018	93,93	+0,25	28,20	-
He *et al*., 2020	93,37	0,31	40,80	-
Yu *et al*., 2018	-	0,18	43,78	43,25
Shi *et al.*, 2023	94.57	**+1,07**	52,10	48,30
Wang *et al.*, 2021	94.11	0,01	61,60	-
Zhang *et al.*, 2023	94.20	0,16	64,00	-
Ding *et al*., 2019	94.54	0,03	-	60,89
Autor, 2025	**95,17**	0,39	**94,32**	**94,61**

FONTE: O autor (2025).

5.2 DISCUSSÃO

5.2.1 Hiperparâmetros

Como demonstrado ao longo do texto desta tese, as RNPs possuem diversos hiperparâmetros relacionados tanto ao treinamento quanto a arquitetura, o que torna o treinamento da rede muito complexo. De maneira ilustrativa, no experimento da CN

(TABELA 3), onde foi utilizada a mesma arquitetura e hiperparâmetros de treinamento, a arquitetura VGG19 com a inicialização de pesos *Narrow-normal* falhou em aprender qualquer padrão apresentado pelos dados, enquanto para as outras configurações testadas os hiperparâmetros permitiram que a rede aprendesse normalmente.

Adicionalmente, existem os hiperparâmetros referentes à própria técnica de poda que também influenciam o desempenho da rede, como a taxa de poda inicial e a taxa de incremento para as próximas rodadas e o número de elementos selecionados.

Por exemplo, se a quantidade de elementos que forem podados entre rodadas for excessiva há o risco de um peso essencial ser podado. Ou ainda, se a poda for muito agressiva e rápida a rede pode perder a sua melhor característica que é a generalização. Outra possibilidade, se o número de épocas de treinamento não for o suficiente, tais pesos cruciais ainda não atingiram os valores necessários para que eles não fossem podados no estágio de seleção da poda pois a rede não teve “tempo” o suficiente para explorar todo o espaço de solução existente.

Neste sentido, a escolha de todos os hiperparâmetros torna-se uma chave fundamental para o sucesso de qualquer treinamento e técnica de poda.

5.2.2 Recursos computacionais

No treinamento das RNPs existe o gargalo que é a quantidade de memória necessária para a execução. Essa necessidade, que é elevada, surge devido à complexidade e dimensão dos modelos envolvidos, bem como da enorme quantidade de dados necessária para alimentar essas redes.

Para o treinamento da rede a memória é utilizada para armazenar os parâmetros do modelo que podem conter milhões ou até mesmo bilhões de parâmetros. Isso se torna especialmente problemático para as CNNs que acumulam muitas camadas e aumentam o tamanho dos seus tensores proporcionalmente.

Já para os conjuntos de dados, o número de elementos necessários no treinamento da rede é sempre o maior disponível; também exigindo uma quantidade significativa de memória para armazená-los e processá-los de forma eficiente durante o treinamento. Adicionalmente, há o processamento das imagens pela técnica *data augmentation.*

Esses fatores podem tornar o treinamento das RNPs impraticável em GPUs domésticas ou com memória limitada. Para exemplificar a demanda de memória a TABELA 13 e a TABELA 14 mostram a quantidade de memória teórica e prática necessária para executar algumas CNNs com alguns tamanhos de imagens que são comuns no campo da visão computacional.

Somada a essa necessidade de GPUs com grande quantidade de memória, há a ‘impossibilidade’ de execução dessas redes nas CPUs devido a quantidade de tempo necessária para executar as redes. Isso se deve à forma como a CPU e a GPU processam os dados. Enquanto a CPU executa os cálculos de forma sequencial em seus poucos núcleos, a GPUs paraleliza o processamento em seus diversos mininúcleos, resultando em um processamento muito mais eficiente.

TABELA 13 – USO TEÓRICO EM GB DE MEMÓRIA COM BATELADA DE 64

Arquitetura	Tamanho da imagem				
	(28x28x1)	(32x32x3)	(224x224x3)	(256x256x3)	(1000x1000x3)
Lenet-5	0,01	0,01	0,69	0,90	13,67
AlexNet	0,13	x	0,33	0,40	4,92
VGG-16	0,90	0,91	4,14	4,80	72,38
ResNet-50	0,27	0,27	8,94	11,64	177,28
ResNet-110	0,60	0,62	19,07	24,84	379,26
DenseNet-201	0,43	0,45	17,83	23,27	364,07
mobilenet_v2	0,12	0,12	5,21	6,80	104,03
inception_v3	0,68	0,68	4,07	5,44	93,62
Xception	2,01	0,26	8,76	10,87	168,39

FONTE: O autor (2025).

TABELA 14 – USO REAL EM GB DE MEMÓRIA COM BATELADA 64

Arquitetura	Tamanho da imagem			
	(28x28x1)	(32x32x3)	(224x224x3)	(256x256x3)
Lenet-5	0,40	0,38	2,27	4,27
AlexNet	0,80	0,73	1,30	1,30
VGG-16	4,27	4,27	14,45	14,14
ResNet-50	8,27	8,27	14,14	14,14
ResNet-110	8,27	12,27	14,16	-
DenseNet-201	8,29	8,25	14,14	-
mobilenet_v2	4,27	4,27	8,30	3,30
inception_v3	8,27	8,27	8,27	8,27
Xception	8,27	8,27	14,14	14,84

FONTE: O autor (2025).

Outro ponto a se considerar é a natureza do algoritmo de poda. Ou seja, para que a rede seja refinada o suficiente, são necessárias diversas rodadas de treinamento e poda. Se a execução de uma RNP com um conjunto de dados grande já exige uma quantidade considerável de memória, agora é adicionado o fator do

tempo necessário para a execução da técnica. Por exemplo, se para o treinamento da rede ResNet-110 são necessárias aproximadamente 8 horas (a depender da quantidade de épocas, batelada e conjunto de dados), em um algoritmo de poda onde são executados em torno de 30 rodadas de poda, são necessários 10 dias de execução para se alcançar ao resultado final da técnica. Adicionalmente, deve-se considerar qual o modelo de GPU é utilizado, uma vez que a velocidade de processamento de imagens por segundo é drasticamente diferente entre modelos disponíveis no mercado.

6 CONCLUSÕES

As CNNs têm um papel fundamental na visão computacional, e essas redes têm se tornado cada vez maiores e mais complexas ao longo dos anos. Neste cenário, as técnicas de poda são um método para superar o número escalável de parâmetros e reduzir o esforço computacional. Essa tese forneceu uma base sólida para as CNNs e para as técnicas de poda, detalhando as suas características e a sua execução. Nesse sentido, os algoritmos de poda demonstram que uma arquitetura esparsa pode alcançar resultados equivalentes ou melhores em comparação com a mesma arquitetura não podada.

Neste trabalho, foi apresentado um novo critério de seleção de pesos aplicado no algoritmo de poda iterativa, a PECN. Como resultado, o algoritmo conseguiu melhorar o desempenho em redes profundas como a ResNet-56 e, quando não houve melhora na acurácia, a perda foi de no máximo 0,68%. Além disso, atingiu-se uma taxa de esparsidade acima de 94% e redução de FLOPs superior a 57%.

Em contrapartida, a limitação encontrada na execução foram os recursos computacionais disponíveis, como: a placa de vídeo disponível e sua respectiva memória dedicada, a quantidade de memória RAM no sistema operacional e o tempo de treinamento das RNPs e da técnica iterativa. Outro fator, inerente a todas as RNPs, são os hiperparâmetros de treinamento, que são fundamentais para que a rede chegue a um bom desempenho, como a escolha adequada da taxa de aprendizagem e o seu decaimento ao longo das épocas de treinamento e o algoritmo de aprendizagem. Adicionalmente, existem os hiperparâmetros da técnica de poda, que acrescentam mais uma camada de complexidade para a definição e execução da rede, onde, para o algoritmo PECN a taxa de amortecimento é um fator crucial para que a rede apresente alta acurácia com alta esparsidade.

Por fim, por meio da classificação das técnicas de poda segundo seus mecanismos de funcionamento, esta tese contribuiu com a uma taxonomia esses métodos, permitindo a identificação das vantagens e limitações de cada abordagem. Além disso, foram apresentados avanços no desempenho em algumas CNNs melhorando o estado da arte. Esses resultados mostram que não é necessário aumentar a quantidade de parâmetros para obter um melhor desempenho, mas sim encontrar uma melhor forma de utilizá-los.

Para trabalhos futuros, pretende-se desenvolver a teoria e provar a convergência do método CN, assim como o da PECN. Além disso, buscar recursos computacionais para expandir ainda mais a aplicação do critério de poda proposto para arquiteturas mais profundas e complexas. Também, será investigada a sua eficácia em outros domínios da visão computacional, como a segmentação de imagens ou aplicações. Outra linha a ser explorada é a integração da técnica desenvolvida com outras técnicas de compressão de RNPs. E como ponto de melhoria do algoritmo, seria interessante automatizar a taxa de amortecimento α para que o a PECN necessite de menor intervenções do usuário.

REFERÊNCIAS

AGGARWAL, C. C. **Neural Networks and Deep Learning**. Cham: Springer International Publishing, 2018.

AHMED, W. S.; KARIM, A. AMIR A. The Impact of Filter Size and Number of Filters on Classification Accuracy in CNN. **2020 International Conference on Computer Science and Software Engineering (CSASE). IEEE**, p.88–93, 2020.

AJIT, A.; ACHARYA, K.; SAMANTA, A. A Review of Convolutional Neural Networks. **International Conference on Emerging Trends in Information Technology and Engineering, (ic-ETITE)**, IEEE, p.1–5, 2020. Disponível em: <https://ieeexplore.ieee.org/document/9077735/>.

ALMEIDA, M.; LASKARIDIS, S.; LEONTIADIS, I.; VENIERIS, S. I.; LANE, N. D. EmBench: Quantifying performance variations of deep neural networks across modern commodity devices. **EMDL 2019 - Proceedings of the 3rd International Workshop on Deep Learning for Mobile Systems and Applications, co-located with MobiSys 2019**, p. 1–6, 2019.

ALOM, M. Z.; TAHA, T. M.; YAKOPCIC, C.; et al. The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches., mar. 2018. Disponível em: <http://arxiv.org/abs/1803.01164>.

ALOM, M. Z.; TAHA, T. M.; YAKOPCIC, C.; et al. A State-of-the-Art Survey on Deep Learning Theory and Architectures. **Electronics**, v. 8, n. 3, p. 292, 2019. Disponível em: <https://www.mdpi.com/2079-9292/8/3/292>.

ALZUBAIDI, L.; ZHANG, J.; HUMAIDI, A. J.; et al. **Review of deep learning: concepts, CNN architectures, challenges, applications, future directions**. Springer International Publishing, 2021.

ARAÚJO, F. H. D. DE. **CONVNETS NA CARACTERIZAÇÃO, RECUPERAÇÃO E RANQUEAMENTO DE CÉLULAS**, Universidade Federal Do Ceará, 2018.

ASSIS, Y.; LIAO, L.; PIERRE, F.; ANXIONNAT, R.; KERRIEN, E. Intracranial aneurysm detection: an object detection perspective. **International Journal of Computer Assisted Radiology and Surgery**, v. 19, n. 9, p. 1667–1675, 2024. Disponível em: <https://link.springer.com/10.1007/s11548-024-03132-z>.

BHINGARKAR, S.; REVATHI, S. T.; KOLLI, C. S.; MEWADA, H. K. An effective optimization enabled deep learning based Malicious behaviour detection in cloud computing. **International Journal of Intelligent Robotics and Applications**, v. 7, n. 3, p. 575–588, 2023. Disponível em: <https://link.springer.com/10.1007/s41315-022-00239-x>.

BJORCK, J.; GOMES, C.; SELMAN, B.; WEINBERGER, K. Q. Understanding batch normalization. **Advances in Neural Information Processing Systems**, v. 2018-Decem, n. NeurIPS, p. 7694–7705, 2018.

BROCK, A.; DE, S.; SMITH, S. L.; SIMONYAN, K. High-Performance Large-Scale Image Recognition Without Normalization. , 2021. Disponível em: <http://arxiv.org/abs/2102.06171>.

CAI, Y.; ZHOU, Y.; HAN, Q.; et al. Reversible Column Networks, 2022. Disponível em: <http://arxiv.org/abs/2212.11696>.

CHANG, V.; XU, Q. A.; AKINLOYE, S. H.; BENSON, V.; HALL, K. Prediction of bank credit worthiness through credit risk analysis: an explainable machine learning study. **Annals of Operations Research**, 2024. Disponível em: <https://link.springer.com/10.1007/s10479-024-06134-x>.

CHARI, R. S. **Image recognition and study of hyperparameter optimization of convolutional neural networks using tensorflow and keras frameworks**, 2018.

CHAUVIN, Y. A Back-propagation Algorithm with Optimal Use of Hidden Units. **Advances in Neural Information Processing Systems**, v. 1, p. 519–526, 1989.

CHEN, Z.; XIE, L.; NIU, J.; et al. Network Adjustment: Channel and Block Search Guided by Resource Utilization Ratio. **International Journal of Computer Vision**, v. 130, n. 3, p. 820–835, 2022. Disponível em: <https://link.springer.com/10.1007/s11263-021-01566-5>.

CHEN, Z.; XU, T. B.; DU, C.; LIU, C. L.; HE, H. Dynamical Channel Pruning by Conditional Accuracy Change for Deep Neural Networks. **IEEE Transactions on Neural Networks and Learning Systems**, v. PP, n. 2, p. 799–813, 2021.

CHOI, B.; LEE, J.-H.; KIM, D.-H. Solving local minima problem with large number of hidden nodes on two-layered feed-forward artificial neural networks. **Neurocomputing**, v. 71, n. 16–18, p. 3640–3643, 2008. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/S0925231208002002>.

CHOLLET, F. Batch Normalization layer. Disponível em: <https://keras.io/api/layers/normalization_layers/batch_normalization/>.

CHOLLET, F.; INC. GOOGLE. Xception: Deep learning with depthwise separable convolutions. **Proceedings of the IEEE conference on computer vision and pattern recognitio.** p.1251–1258, 2017.

CHU, X.; ZHANG, B.; LI, Q.; XU, R.; LI, X. SCARLET-NAS: Bridging the Gap between Stability and Scalability in Weight-sharing Neural Architecture Search. , 2019. Disponível em: <http://arxiv.org/abs/1908.06022>.

CHUNG, F. L.; LEE, T. A node pruning algorithm for backpropagation networks. **International Journal of Neural Systems**, v. 03, n. 03, p. 301–314, 1992. Disponível em: <https://www.worldscientific.com/doi/abs/10.1142/S0129065792000231>.

CLEVERT, D. A.; UNTERTHINER, T.; HOCHREITER, S. Fast and accurate deep network learning by exponential linear units (ELUs). **arXiv preprint arXiv: 1511.07289**, 2016.

CUN, L.; BOSER; DENKER; et al. Handwritten Digit Recognition with a Back-Propagation Network. **Advances in neural information processing systems 2**, p. 396–404, 1990.

DENG, J.; DONG, W.; SOCHER, R.; et al. ImageNet: A large-scale hierarchical image database. **2009 IEEE Conference on Computer Vision and Pattern Recognition**. p.248–255, 2009. IEEE. Disponível em: <https://ieeexplore.ieee.org/document/5206848>.

DING, X.; DING, G.; GUO, Y.; HAN, J. Centripetal SGD for pruning very deep convolutional networks with complicated structure. **Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition**, v. 2019-June, p. 4938–4948, 2019.

DING, X.; DING, G.; HAN, J.; TANG, S. Auto-balanced filter pruning for efficient convolutional neural networks. **32nd AAAI Conference on Artificial Intelligence, AAAI 2018**, p. 6797–6804, 2018.

DONG, X.; CHEN, S.; PAN, S. J. Learning to prune deep neural networks via layer-wise optimal brain surgeon. **Advances in Neural Information Processing Systems**, v. 2017-Decem, n. Nips, p. 4858–4868, 2017. Disponível em: <http://arxiv.org/abs/1705.07565>.

DREYFUS, S. The numerical solution of variational problems. **Journal of Mathematical Analysis and Applications**, v. 5, n. 1, p. 30–45, 1962.

DU, F.; MA, X.-J.; YANG, J.-R.; et al. A Survey of LLM Datasets: From Autoregressive Model to AI Chatbot. **Journal of Computer Science and Technology**, v. 39, n. 3, p. 542–566, 2024. Disponível em: <https://link.springer.com/10.1007/s11390-024-3767-3>.

DUCH, W.; KORCZAK, J. Optimization and global minimization methods suitable for neural networks. **Neural computing surveys**, v. 2, n. 8, p. 163–212, 1998.

FERREIRA, W. D. **Detecção de Imagens Falsificadas Baseada em Descritores Locais de Textura e Rede Neural Convolucional**, Universidade Federal De Goiás, 2020.

FRANKLE, J.; CARBIN, M. The lottery ticket hypothesis: Finding sparse, trainable neural networks. **7th International Conference on Learning Representations, ICLR 2019. arXiv preprint arXiv:1803.03635**, p. 1–42, 2019.

FUKUSHIMA, K. Self-organization of a neural network which gives position-invariant response. **Proceedings of the 6th international joint conference on Artificial intelligence-Volume 1.** v. 1, p.291–293, 1979.

FUKUSHIMA, K. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. **Biological Cybernetics**, v. 36, n. 4, p. 193–202, 1980. Disponível em: <http://link.springer.com/10.1007/BF00344251>.

FUKUSHIMA, K. Neocognitron: A hierarchical neural network capable of visual pattern recognition. **Neural Networks**, v. 1, n. 2, p. 119–130, 1988.

GEIPING, J.; GOLDBLUM, M.; SOMEPALLI, G.; et al. How Much Data Are Augmentations Worth? An Investigation into Scaling Laws, Invariance, and Implicit Regularization. , 2022. Disponível em: <http://arxiv.org/abs/2210.06441>.

GENG, L.; NIU, B. Pruning convolutional neural networks via filter similarity analysis. **Machine Learning**, v. 111, n. 9, p. 3161–3180, 2022. Springer US. Disponível em: <https://doi.org/10.1007/s10994-022-06193-w>.

GLOROT, X.; BENGIO, Y. Understanding the difficulty of training deep feedforward neural networks. **Proceedings of the thirteenth international conference on artificial intelligence and statistics.** JMLR Workshop and Conference Proceedings. p.249–256, 2010. Disponível em: <https://proceedings.mlr.press/v9/glorot10a.html>.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. MIT press, 2016.

GORACH, T. Deep Convolutional Neural Networks - A Review. **International Research Journal of Engineering and Technology (IRJET)**, v. 5, n. 07, p. 439–452, 2018. Disponível em: <http://www.dbpia.co.kr/Journal/ArticleDetail/NODE07109492>

.

GUO, Y.; YAO, A.; CHEN, Y. Dynamic Network Surgery for Efficient DNNs. **Advances in Neural Information Processing Systems**, , n. Nips, p. 1387–1395, 2016. Disponível em: <http://arxiv.org/abs/1608.04493>.

HABIBI AGHDAM, H.; JAHANI HERAVI, E. **Guide to Convolutional Neural Networks**. Cham: Springer International Publishing, 2017.

HAGIWARA, M. Removal of hidden units and weights for back propagation networks. **Proceedings of the International Joint Conference on Neural Networks**, v. 1, p. 351–353, 1993. IEEE. Disponível em: <http://ieeexplore.ieee.org/document/713929/>.

HAN, K.; WANG, Y.; TIAN, Q.; et al. GhostNet: More features from cheap operations. **Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition**, p. 1577–1586, 2020. IEEE. Disponível em: <https://ieeexplore.ieee.org/document/9157333/>.

HAN, S.; POOL, J.; TRAN, J.; DALLY, W. J. Learning both weights and connections for efficient neural networks. **Advances in Neural Information Processing Systems**, v. 2015-Janua, p. 1135–1143, 2015.

HANNUN, A.; CASE, C.; CASPER, J.; et al. Deep Speech: Scaling up end-to-end speech recognition. **arXiv preprint arXiv:1412.5567**, 2014. Disponível em: <http://arxiv.org/abs/1412.5567>.

HASANPOUR, S. H.; ROUHANI, M.; FAYYAZ, M.; SABOKROU, M. Lets keep it simple, Using simple architectures to outperform deeper and more complex architectures. , 2016. Disponível em: <http://arxiv.org/abs/1608.06037>.

HASSIBI, B., & STORK, D. G. Second Order Derivatives for Network Pruning: Optimal Brain Surgeon. Advances in neural information processing systems, 5. p.164–171, 1992.

HASSIBI, B.; STORK, D. G. D. G.; WOLFF, G. J. G. J. Optimal Brain Surgeon and general network pruning. **1993 IEEE International Conference on Neural Networks.** p.293–299, 1993. IEEE. Disponível em: <http://ieeexplore.ieee.org/document/298572/>.

HE, K.; ZHANG, X.; REN, S.; SUN, J. Deep residual learning for image recognition. **Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition**, p. 770–778, 2015a.

HE, K.; ZHANG, X.; REN, S.; SUN, J. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. **2015 IEEE International Conference on Computer Vision (ICCV)**, v. 2015 Inter, p. 1026–1034, 2015b. IEEE. Disponível em: <http://arxiv.org/abs/1502.01852>.

HE, Y. Pruning Very Deep Neural Network Channels for Efficient Inference. **arXiv preprint arXiv: 2211.08339**, p. 1–12, 2022. Disponível em: <http://arxiv.org/abs/2211.08339>.

HE, Y.; DONG, X.; KANG, G.; et al. Asymptotic Soft Filter Pruning for Deep Convolutional Neural Networks. **IEEE Transactions on Cybernetics**, v. 50, n. 8, p. 3594–3604, 2020. Disponível em: <https://ieeexplore.ieee.org/document/8816678/>.

HE, Y.; KANG, G.; DONG, X.; FU, Y.; YANG, Y. Soft filter pruning for accelerating deep convolutional neural networks. **arXiv preprint arXiv: 1808.06866**, p. 2234–2240,

2018.

HE, Y.; LIU, P.; WANG, Z.; HU, Z.; YANG, Y. Filter Pruning via Geometric Median for Deep Convolutional Neural Networks Acceleration. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). v. 2019-June, p.4335–4344, 2019. IEEE. Disponível em: <https://ieeexplore.ieee.org/document/8953212/>.

HE, Y.; ZHANG, X.; SUN, J. Channel Pruning for Accelerating Very Deep Neural Networks. **2017 IEEE International Conference on Computer Vision (ICCV).** v. 2017-Octob, p.1398–1406, 2017. IEEE. Disponível em: <http://arxiv.org/abs/1707.06168>.

HEBB, D. O. The organization of behavior: a neuropsychological theory. Wiley; New York, NY, 1949.

HINTON, G. E.; SRIVASTAVA, N.; KRIZHEVSKY, A.; SUTSKEVER, I.; SALAKHUTDINOV, R. R. Improving neural networks by preventing co-adaptation of feature detectors. **arXiv preprint arXiv:1207.0580**, p. 1–18, 2012. Disponível em: <http://arxiv.org/abs/1207.0580>.

HOLLARD, L.; MOHIMONT, L.; GAVEAU, N.; STEFFENEL, L.-A. LeYOLO, New Scalable and Efficient CNN Architecture for Object Detection. , 2024. Disponível em: <http://arxiv.org/abs/2406.14239>.

HOWARD, A. G.; ZHU, M.; CHEN, B.; et al. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. **arXiv Prepr arXiv: 1704.04861**, 2017. Disponível em: <http://arxiv.org/abs/1704.04861>.

HU, H.; PENG, R.; TAI, Y.-W.; TANG, C.-K. Network Trimming: A Data-Driven Neuron Pruning Approach towards Efficient Deep Architectures. **arXiv preprint arXiv: 1607.03250**, 2016. Disponível em: <http://arxiv.org/abs/1607.03250>.

HUANG, G.; LIU, Z.; VAN DER MAATEN, L.; WEINBERGER, K. Q. Densely connected convolutional networks. **Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017**, 2016.

HUANG, Q.; ZHOU, K.; YOU, S.; NEUMANN, U. Learning to prune filters in convolutional neural networks. **Proceedings - 2018 IEEE Winter Conference on Applications of Computer Vision, WACV 2018**, v. 2018-Janua, p. 709–718, 2018.

HUANG, T.; DONG, W.; LIU, J.; et al. Accelerating Convolutional Neural Network via Structured Gaussian Scale Mixture Models: A Joint Grouping and Pruning Approach. **IEEE Journal on Selected Topics in Signal Processing**, v. 14, n. 4, p. 817–827, 2020.

HUBEL, D. H.; WIESEL, T. N. Receptive fields of single neurones in the cat's striate cortex. **The Journal of Physiology**, v. 148, n. 3, p. 574–591, 1959. Disponível em: <https://onlinelibrary.wiley.com/doi/10.1113/jphysiol.1959.sp006308>.

HUBEL, D. H.; WIESEL, T. N. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. **The Journal of Physiology**, v. 160, n. 1, p. 106–154, 1962.

HUBEL, D. H.; WIESEL, T. N. Receptive fields and functional architecture of monkey striate cortex. **The Journal of Physiology**, v. 195, n. 1, p. 215–243, 1968. Disponível em: <https://onlinelibrary.wiley.com/doi/10.1113/jphysiol.1968.sp008455>.

HUBENS, N.; MANCAS, M.; GOSSELIN, B.; PREDA, M.; ZAHARIA, T. Improve

Convolutional Neural Network Pruning by Maximizing Filter Variety. **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**, v. 13231 LNCS, p. 379–390, 2022.

HUYNH, N.; NGUYEN, K.-D. Real-Time Droplet Detection for Agricultural Spraying Systems: A Deep Learning Approach. **Machine Learning and Knowledge Extraction**, v. 6, n. 1, p. 259–282, 2024. Disponível em: <https://www.mdpi.com/2504-4990/6/1/14>.

IANDOLA, F. N.; HAN, S.; MOSKEWICZ, M. W.; et al. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size. **arXiv preprint arXiv:1602.07360**, p. 1–13, 2016. Disponível em: <http://arxiv.org/abs/1602.07360>.

IOFFE, S.; SZEGEDY, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. **32nd International Conference on Machine Learning, ICML 2015**, v. 1, p. 448–456, 2015.

ISHIKAWA, M. Structural learning with forgetting. **Neural Networks**, v. 9, n. 3, p. 509–521, 1996.

JANOWSKY, S. A. Pruning versus clipping in neural networks Steven. **Physical Review A**, v. 39, n. 12, p. 6600, 1989.

JIANCHANG MAO; MOHIUDDIN, K.; JAIN, A. K. Parsimonious network design and feature selection through node pruning. Proceedings of the 12th IAPR International Conference on Pattern Recognition, Vol. 3-Conference C: Signal Processing (Cat. No. 94CH3440-5). v. 2, p.622–624, 1994. IEEE Comput. Soc. Press. Disponível em: <http://ieeexplore.ieee.org/document/577060/>.

JIANG, D.; CAO, Y.; YANG, Q. On the Channel Pruning Using Graph Convolution Network for Convolutional Neural Network Acceleration. **IJCAI International Joint Conference on Artificial Intelligence**, p. 3107–3113, 2022.

KARNIN, E. D. A simple procedure for pruning back-propagation trained neural networks. **IEEE Transactions on Neural Networks**, v. 1, n. 2, p. 239–242, 1990. Disponível em: <http://ieeexplore.ieee.org/document/80236/>.

KELLEHER, J. D. **Deep Learning**. Cambridge, MA: MIT Press, 2019.

KHAN, A.; SOHAIL, A.; ZAHOORA, U.; QURESHI, A. S. A survey of the recent architectures of deep convolutional neural networks. **Artificial Intelligence Review**, p. 1–70, 2020.

KHANDAY, O. M.; DADVANDIPOUR, S. Convolutional Neural Networks and Impact of Filter Sizes on Image Classification. **Multidiszciplináris Tudományok**, v. 10, n. 1, p. 55–60, 2020.

KRIZHEVSKY, A.; HINTON, G. Learning multiple layers of features from tiny images. Disponível em: <https://www.cs.toronto.edu/~kriz/cifar.html>.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. ImageNet classification with deep convolutional neural networks. **Advances in neural information processing systems**, v. 25, 2012. Disponível em: <https://dl.acm.org/doi/10.1145/3065386>.

KROHN, J.; BEYLEVELD, G.; BASSENS, A. **Deep Learning Illustrated: A Visual, Interactive Guide to Artificial Intelligence**. Addison-Wesley, 2020.

KUMAR, T.; BRENNAN, R.; MILEO, A.; BENDECHACHE, M. Image Data Augmentation Approaches: A Comprehensive Survey and Future Directions. **IEEE**

Access, p. 1–1, 2024. Disponível em: <https://ieeexplore.ieee.org/document/10699340/>.

LECUN, Y. Generalization and Network Design Strategies. **Connectionism in perspective**, v. 19, n. 143–155, p. 18, 1989.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. **Nature**, v. 521, n. 7553, p. 436–444, 2015.

LECUN, Y.; BOTTOU, L.; BENGIO, Y.; HAFFNER, P. Gradient-based learning applied to document recognition. **Proceedings of the IEEE**, v. 86, n. 11, p. 2278–2324, 1998. Disponível em: <http://ieeexplore.ieee.org/document/726791/#full-text-section>.

LECUN, Y.; DENKER, J. S.; SOLLA, S. A. Optimal brain damage. **Advances in neural information processing systems**, v. 2, 1990.

LECUN YANN; CORTES CORINNA; BURGES CHRISTOPHER. The MNIST database of Handwritten Digits. Disponível em: <http://yann.lecun.com/exdb/mnist/>.

LEE, N.; AJANTHAN, T.; TORR, P. H. S. S. SNIP: Single-shot Network Pruning based on Connection Sensitivity. **7th International Conference on Learning Representations, ICLR 2019. arXiv preprint arXiv:1810.02340**, 2018. Disponível em: <http://arxiv.org/abs/1810.02340>.

LI, F. J.; YS. CS231n: Convolutional Neural Networks for Visual Recognition. Disponível em: <https://cs231n.github.io/convolutional-networks/>.

LI, G.; QIAN, C.; JIANG, C.; LU, X.; TANG, K. Optimization based layer-wise magnitude-based pruning for DNN compression. **IJCAI International Joint Conference on Artificial Intelligence**, v. 2018-July, p. 2383–2389, 2018.

LI, H.; KADAV, A.; DURDANOVIC, I.; SAMET, H.; GRAF, H. P. Pruning Filters for Efficient ConvNets. **5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings**, , n. 2016, p. 1–13, 2016. Disponível em: <http://arxiv.org/abs/1608.08710>.

LI, Q.; CAI, W.; WANG, X.; et al. Medical image classification with convolutional neural network. **2014 13th International Conference on Control Automation Robotics and Vision, ICARCV 2014**, v. 2014, n. December, p. 844–848, 2014.

LI, Q.; LI, C.; CHEN, H. Incremental filter pruning via random walk for accelerating deep convolutional neural networks. **WSDM 2020 - Proceedings of the 13th International Conference on Web Search and Data Mining**, p. 358–366, 2020.

LI, Q.; LI, C.; CHEN, H. Filter Pruning via Probabilistic Model-based Optimization for Accelerating Deep Convolutional Neural Networks. **WSDM 2021 - Proceedings of the 14th ACM International Conference on Web Search and Data Mining**, p. 653–661, 2021.

LI, Z.; LIU, F.; YANG, W.; PENG, S.; ZHOU, J. A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects. **IEEE Transactions on Neural Networks and Learning Systems**, p. 1–21, 2021. Disponível em: <http://arxiv.org/abs/2004.02806>.

LI, Z.; YANG, W.; PENG, S.; LIU, F. A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects. , 2020. Disponível em: <http://arxiv.org/abs/2004.02806>.

LIN, M.; CHEN, Q.; YAN, S. Network in network. 2nd International Conference on

Learning Representations, ICLR 2014 - Conference Track Proc. p.1–10, 2014.

LIN, M.; JI, R.; WANG, Y.; et al. Hrank: Filter pruning using high-Rank feature map. **Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition**, p. 1526–1535, 2020. IEEE. Disponível em: <https://ieeexplore.ieee.org/document/9156677/>.

LIN, M.; JI, R.; ZHANG, Y.; et al. Channel pruning via automatic structure search. **arXiv preprint arXiv: 2001.08565**, v. 2021-Janua, p. 673–679, 2020. California: International Joint Conferences on Artificial Intelligence Organization. Disponível em: <https://www.ijcai.org/proceedings/2020/94>.

LIN, T.; STICH, S. U.; BARBA, L.; DMITRIEV, D.; JAGGI, M. Dynamic Model Pruning With Feedback. **8th International Conference on Learning Representations, ICLR 2020. arXiv preprint arXiv:2006.07253**, 2020. Disponível em: <http://arxiv.org/abs/2006.07253>.

LIU, H.; XIN, B.; MU, S.; ZHU, Z. Pruning the deep neural network by similar function. **Journal of Physics: Conference Series**, v. 1187, n. 4, 2019.

LIU, Y.; KONG, H.; YU, P. Automatic Compression Ratio Allocation for Pruning Convolutional Neural Networks. **ACM International Conference Proceeding Series**, 2019.

LU, Z.; SREEKUMAR, G.; GOODMAN, E.; et al. Neural Architecture Transfer. , 2020. Disponível em: <http://arxiv.org/abs/2005.05859>.

LUO, T.; CAI, T.; ZHANG, M.; CHEN, S.; WANG, L. RANDOM MASK: Towards Robust Convolutional Neural Networks. **arXiv preprint arXiv: 2007.14249**, 2020. Disponível em: <http://arxiv.org/abs/2007.14249>.

MAAS, A. L.; HANNUN, A. Y.; NG, A. Y. Rectifier nonlinearities improve neural network acoustic models. Proc. icml. v. 30, p.3, 2013.

MAAZ, M.; SHAKER, A.; CHOLAKKAL, H.; et al. EdgeNeXt: Efficiently Amalgamated CNN-Transformer Architecture for Mobile Vision Applications. , 2022. Disponível em: <http://arxiv.org/abs/2206.10589>.

MAITRA, S.; OJHA, R. K.; GHOSH, K. Impact of Convolutional Neural Network Input Parameters on Classification Performance. 2018 4th International Conference for Convergence in Technology, I2CT 2018. p.1–5, 2018. IEEE.

MALACH, E.; YEHUDAI, G.; SHALEV-SHWARTZ, S.; SHAMIR, O. Proving the Lottery Ticket Hypothesis : Pruning is All You Need. International Conference on Machine I Learning. PMLR. p.6682–6691, 2020.

MAMAEV, A. Flowers Recognition. Disponível em: <https://www.kaggle.com/alxmamaev/flowers-recognition>.

MOLCHANOV, P.; TYREE, S.; KARRAS, T.; AILA, T.; KAUTZ, J. Pruning Convolutional Neural Networks for Resource Efficient Inference. **arXiv preprint arXiv:1611.06440**, 2016.

MOZER, MICHAEL C; SMOLENSKY, P. Skeletonization: A Technique for Trimming the Fat from a Network via Relevance Assessment. **Advances in Neural Information Processing Systems**, v. 1, p. 107–115, 1989. Disponível em: <http://papers.neurips.cc/paper/119-skeletonization-a-technique-for-trimming-the-fat-from-a-network-via-relevance-assessment.pdf>.

MOZER, MICHAEL C.; SMOLENSKY, P. Using Relevance to Reduce Network Size Automatically. **Connection Science**, v. 1, n. 1, p. 3–16, 1989.

MURASE, K.; MATSUNAGA, Y.; NAKADE, Y. A Back-Propagation Algorithm which Automatically Determines the Number of Association Units. 1991 IEEE International Joint Conference on Neural Networks. p.783–788, 1991.

NAIR, V.; HINTON, G. E. Rectified linear units improve Restricted Boltzmann machines. ICML 2010 - Proceedings, 27th International Conference on Machine Learning. p.807–814, 2010.

NAJAFABADI, M. M.; VILLANUSTRE, F.; KHOSHGOFTAAR, T. M.; et al. Deep learning applications and challenges in big data analytics. **Journal of Big Data**, v. 2, n. 1, p. 1, 2015. Disponível em: <http://www.journalofbigdata.com/content/2/1/1>.

NARKHEDE, M. V.; BARTAKKE, P. P.; SUTAONE, M. S. A review on weight initialization strategies for neural networks. **Artificial Intelligence Review**, v. 55, n. 1, p. 291–322, 2022. Springer Netherlands. Disponível em: <https://doi.org/10.1007/s10462-021-10033-z>.

NWANKPA, C.; IJOMAH, W.; GACHAGAN, A.; MARSHALL, S. Activation Functions: Comparison of trends in Practice and Research for Deep Learning. , p. 1–20, 2018a. Disponível em: <http://arxiv.org/abs/1811.03378>.

NWANKPA, C.; IJOMAH, W.; GACHAGAN, A.; MARSHALL, S. Activation Functions: Comparison of trends in Practice and Research for Deep Learning. **arXiv preprint arXiv:1811.03378**, 2018b. Disponível em: <http://arxiv.org/abs/1811.03378>.

OLIMOV, B.; KARSHIEV, S.; JANG, E.; et al. Weight initialization based-rectified linear unit activation function to improve the performance of a convolutional neural network model. **Concurrency and Computation: Practice and Experience**, v. 33, n. 22, p. 1–11, 2021. Disponível em: <https://onlinelibrary.wiley.com/doi/10.1002/cpe.6143>.

PATTERSON, J.; GIBSON, A. **Deep learning: A Pratitioner's Approach**. O'Reilly Media, Inc., 2017.

PINECONE SYSTEMS, I. AlexNet and ImageNet: The Birth of Deep Learning. Disponível em: <https://www.pinecone.io/learn/series/image-search/imagenet/>.

POLYAK, A.; WOLF, L. Channel-level acceleration of deep face representations. **IEEE Access**, v. 3, p. 2163–2175, 2015. IEEE.

POOLE, D. L.; MACKWORTH, A. K. **Artificial Intelligence**. f Cambridge University Press, 2010.

PRINCE, SIMON J. D. **Computer Vision: Model, Learning and Inference**. 2012.

PRINCE, S. J. D. Understanding Deep Learning. , , n. December 2023, p. 986, 2023.

QIAN, Y.; WOODLAND, P. C. Very deep convolutional neural networks for robust speech recognition. 2016 IEEE Spoken Language Technology Workshop (SLT). v. 1, p.481–488, 2016. IEEE. Disponível em: <http://ieeexplore.ieee.org/document/7846307/>.

QIN, Z.; YU, F.; LIU. CHENCHEN; CHEN, X. Interpretable convolutional filter pruning. , 2019. Disponível em: <https://openreview.net/forum?id=BJ4BVhRcYX>.

RAWAT, W.; WANG, Z. Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review. **Neural Computation**, v. 29, n. 9, p. 2352–2449, 2017.

Disponível em: <http://arxiv.org/abs/1803.01446>.

REAL, E.; AGGARWAL, A.; HUANG, Y.; LE, Q. V. Regularized Evolution for Image Classifier Architecture Search. , 2018. Disponível em: <http://arxiv.org/abs/1802.01548>.

REED, R. Pruning Algorithms - A survey. **IEEE TRANSACTIONS ON NEURAL NETWORKS**, v. 4, n. 5, p. 740–747, 1993.

DE RESENDE OLIVEIRA, F. D.; BATISTA, E. L. O.; SEARA, R. On the compression of neural networks using ℓ0-norm regularization and weight pruning. **Neural Networks**, v. 171, n. December 2023, p. 343–352, 2024.

SANDLER, M.; HOWARD, A.; ZHU, M.; ZHMOGINOV, A.; CHEN, L.-C. MobileNetV2: Inverted Residuals and Linear Bottlenecks. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. p.4510–4520, 2018. IEEE. Disponível em: <https://ieeexplore.ieee.org/document/8578572/>.

SEGEE, B. E.; CARTER, M. J. Fault tolerance of pruned multilayer networks. IJCNN-91-Seattle International Joint Conference on Neural Networks. v. 2, p.447–452, 2018. IEEE. Disponível em: <http://ieeexplore.ieee.org/document/155374/>.

SHARMA, P.; NAYAK, D. R.; BALABANTARAY, B. K.; TANVEER, M.; NAYAK, R. A survey on cancer detection via convolutional neural networks: Current challenges and future directions. **Neural Networks**, v. 169, p. 637–659, 2024. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/S0893608023006287>.

SHEN, X.; WANG, YAOHUA; LIN, M.; et al. DeepMAD: Mathematical Architecture Design for Deep Convolutional Neural Network. , 2023. Disponível em: <http://arxiv.org/abs/2303.02165>.

SHETTY, D.; HARSHAVARDHAN, C. A.; VARMA, M. J.; NAVI, S.; AHMED, M. R. Diving Deep into Deep Learning:History, Evolution, Types and Applications. **International Journal of Innovative Technology and Exploring Engineering**, v. 9, n. 3, p. 2835–2846, 2020. Disponível em: <http://www.ijitee.org/wp-content/uploads/papers/v9i3/A4865119119.pdf>.

SHI, C.; HAO, Y.; LI, G.; XU, S. VNGEP: Filter pruning based on von Neumann graph entropy. **Neurocomputing**, v. 528, p. 113–124, 2023. Elsevier B.V. Disponível em: <https://doi.org/10.1016/j.neucom.2023.01.046>.

SHI, Y.; TANG, A.; NIU, L.; ZHOU, R. Sparse optimization guided pruning for neural networks. **Neurocomputing**, v. 574, n. October 2023, p. 127280, 2024. Elsevier B.V. Disponível em: <https://doi.org/10.1016/j.neucom.2024.127280>.

SHORTEN, C.; KHOSHGOFTAAR, T. M. A survey on Image Data Augmentation for Deep Learning. **Journal of Big Data**, v. 6, n. 1, 2019. Springer International Publishing. Disponível em: <https://doi.org/10.1186/s40537-019-0197-0>.

SIETSMA, J.; DOW, R. J. F. Neural net pruning - why and how. IEEE 1988 international conference on neural networks. p.325–333, 1988. San Diego, CA.

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. **arXiv preprint arXiv:1409.1556**, 2014.

SINGH, P.; VERMA, V. K.; RAI, P.; NAMBOODIRI, V. P. Acceleration of Deep Convolutional Neural Networks Using Adaptive Filter Pruning. **IEEE Journal on Selected Topics in Signal Processing**, v. 14, n. 4, p. 838–847, 2020.

SRINIVAS, S.; SUBRAMANYA, A.; BABU, R. V. Training Sparse Neural Networks. **IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops**, v. 2017-July, p. 455–462, 2017.

STEPNIEWSKI, S. W.; KEANE, A. J. Pruning backpropagation neural networks using modern stochastic optimisation techniques. **Neural Computing & Applications**, v. 5, n. 2, p. 76–98, 1997. Disponível em: <http://link.springer.com/10.1007/BF01501173>.

SUAU, X.; ZAPPELLA, L.; APOSTOLOFF, N. Filter distillation for network compression. **Proceedings - 2020 IEEE Winter Conference on Applications of Computer Vision, WACV 2020**, p. 3129–3138, 2020.

SZE, V.; CHEN, Y.-H.; YANG, T.-J.; EMER, J. S. Efficient Processing of Deep Neural Networks: A Tutorial and Survey. **Proceedings of the IEEE**, v. 105, n. 12, p. 2295–2329, 2017. Disponível em: <http://ieeexplore.ieee.org/document/8114708/>.

SZEGEDY, C.; LIU, W.; JIA, Y.; et al. Going deeper with convolutions. **Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition**, v. 07-12-June, n. 8, p. 1–9, 2015. IEEE. Disponível em: <http://ieeexplore.ieee.org/document/7298594/>.

TAN, M.; LE, Q. V. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. International conference on machine learning. PMLR. p.6105–6114, 2019. Disponível em: <http://arxiv.org/abs/1905.11946>.

TESSIER, H.; GRIPON, V.; LÉONARDON, M.; et al. Rethinking Weight Decay for Efficient Neural Network Pruning. **Journal of Imaging**, v. 8, n. 3, p. 1–24, 2022.

TIAN, K.; JIANG, Y.; DIAO, Q.; et al. Designing BERT for Convolutional Networks: Sparse and Hierarchical Masked Modeling. , 2023. Disponível em: <http://arxiv.org/abs/2301.03580>.

TRESP, V.; NEUNEIER, R.; ZIMMERMANN, H. G. Early brain damage. **Advances in Neural Information Processing Systems**, p. 669–675, 1997.

WANG, J.; LIU, L.; PAN, X. Pruning algorithm of convolutional neural network based on optimal threshold. **ACM International Conference Proceeding Series**, p. 50–54, 2020.

WANG, S.; ZHANG, Z. ScoringNet: A Neural Network Based Pruning Criteria for Structured Pruning. **Scientific Programming**, v. 2023, 2023.

WANG, T.; WU, D. J.; COATES, A.; NG, A. Y. End-to-end text recognition with convolutional neural networks. Proceedings - International Conference on Pattern Recognition, 2012.

WANG, Z.; LI, C.; WANG, X. Convolutional neural network pruning with structural redundancy reduction. **Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition**, p. 14908–14917, 2021.

WEIGEND, A. S.; RUMELHART, D. E.; HUBERMAN, B. A. Generalization by weight-elimination with application to forecasting. **Neural Information Processing Systems**, v. 3, n. i, p. 875–882, 1991.

WERBOS, P. **Beyond regression: New tools for prediction and analysis in the behavioral sciences**, 1974. Harvard University. Disponível em: <https://perceptrondemo.com/assets/PJW_thesis_Beyond_Regression_1974-4b63aa5f.pdf>.

WERBOS, P. J. Applications of advances in nonlinear sensitivity analysis. **System modeling and optimization**, p. 762–770, 1982.

WHITLEY, D. The evolution of connectivity: Pruning neural networks using genetic algorithm. Proceedings of IJCNN-90. p.134–137, 1990.

WIGHTMAN, R.; TOUVRON, H.; JÉGOU, H. ResNet strikes back: An improved training procedure in timm. , 2021. Disponível em: <http://arxiv.org/abs/2110.00476>.

WU, C.; PANG, W.; LIU, H.; LU, S. Group pruning with group sparse regularization for deep neural network compression. **2019 IEEE 4th International Conference on Signal and Image Processing, ICSIP 2019**, p. 325–329, 2019. IEEE.

XIE, S.; KIRILLOV, A.; GIRSHICK, R.; HE, K. Exploring Randomly Wired Neural Networks for Image Recognition. , 2019. Disponível em: <http://arxiv.org/abs/1904.01569>.

XU, K.; ZHANG, D.; AN, J.; et al. GenExp: Multi-objective pruning for deep neural network based on genetic algorithm. **Neurocomputing**, v. 451, p. 81–94, 2021. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/S092523122100549X>.

XU, Y.; LIAO, Y.; ZHAO, Y. Filter Pruning Based on Connection Sensitivity. **PervasiveHealth: Pervasive Computing Technologies for Healthcare**, 2020.

XUE-WEN CHEN; XIAOTONG LIN. Big Data Deep Learning: Challenges and Perspectives. **IEEE Access**, v. 2, n. 2, p. 514–525, 2014. Disponível em: <http://ieeexplore.ieee.org/document/6296526/>.

YAM, J. Y. F.; CHOW, T. W. S. A weight initialization method for improving training speed in feedforward neural network. **Neurocomputing**, v. 30, n. 1–4, p. 219–232, 2000. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/S0925231299001277>.

YAMAMOTO, K.; MAENO, K. PCAS: Pruning channels with attention statistics for deep network compression. **30th British Machine Vision Conference 2019, BMVC 2019. a**rXiv preprint arXiv:1806.05382, 2020.

YANG, C.; AN, Z.; ZHU, H.; et al. Gated Convolutional Networks with Hybrid Connectivity for Image Classification. , 2019. Disponível em: <http://arxiv.org/abs/1908.09699>.

YANG, G.; PENNINGTON, J.; RAO, V.; SOHL-DICKSTEIN, J.; SCHOENHOLZ, S. S. A Mean Field Theory of Batch Normalization. , 2019. Disponível em: <http://arxiv.org/abs/1902.08129>.

YE, J.; LU, X.; LIN, Z.; WANG, J. Z. Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers. **6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings**, , n. 2017, p. 1–11, 2018. Disponível em: <http://arxiv.org/abs/1802.00124>.

YEOM, S. K.; SEEGERER, P.; LAPUSCHKIN, S.; et al. Pruning by explaining: A novel criterion for deep neural network pruning. **Pattern Recognition**, v. 115, p. 107899, 2021. Elsevier Ltd. Disponível em: <https://doi.org/10.1016/j.patcog.2021.107899>.

YOU, Z.; YAN, K.; YE, J.; MA, M.; WANG, P. Gate decorator: Global filter pruning method for accelerating deep convolutional neural networks. **Advances in Neural Information Processing Systems**, v. 32, n. NeurIPS, p. 1–12, 2019.

YU, D.; WANG, H.; CHEN, P.; WEI, Z. Mixed Pooling for Convolutional Neural

Networks. **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**. v. 8818, p.364–375, 2014. Disponível em: <http://link.springer.com/10.1007/978-3-319-11740-9_34>.

YU, R.; LI, A.; CHEN, C.-F. F.; et al. NISP: Pruning Networks Using Neuron Importance Score Propagation. **Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition**, p. 9194–9203, 2018. IEEE. Disponível em: <https://ieeexplore.ieee.org/document/8579056/>.

ZAINELDIN, H.; GAMEL, S. A.; TALAAT, F. M.; et al. Silent no more: a comprehensive review of artificial intelligence, deep learning, and machine learning in facilitating deaf and mute communication. **Artificial Intelligence Review**, v. 57, n. 7, p. 188, 2024. Disponível em: <https://link.springer.com/10.1007/s10462-024-10816-0>.

ZEILER, M. D.; FERGUS, R. Visualizing and Understanding Convolutional Networks. **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**, v. 8689 LNCS, n. PART 1, p. 818–833, 2013. Disponível em: <http://arxiv.org/abs/1311.2901>.

ZHANG, E. Fruit Recognition. Disponível em: <https://www.kaggle.com/sshikamaru/fruit-recognition>.

ZHANG, H.; WU, C.; ZHANG, Z. Z.; et al. ResNeSt: Split-Attention Networks. , 2020. Disponível em: <http://arxiv.org/abs/2004.08955>.

ZHANG, P.; TIAN, C.; ZHAO, L.; DUAN, Z. A multi-granularity CNN pruning framework via deformable soft mask with joint training. **Neurocomputing**, v. 572, n. June 2023, p. 127189, 2023. Elsevier B.V. Disponível em: <https://doi.org/10.1016/j.neucom.2023.127189>.

ZHANG, Q.; ZHANG, M.; CHEN, T.; et al. Recent advances in convolutional neural network acceleration. **Neurocomputing**, v. 323, p. 37–51, 2019. Elsevier B.V. Disponível em: <https://doi.org/10.1016/j.neucom.2018.09.038>.

ZHANG, S.; WU, G.; GU, J.; HAN, J. Pruning convolutional neural networks with an attention mechanism for remote sensing image classification. **Electronics (Switzerland)**, v. 9, n. 8, p. 1–19, 2020.

ZHANG, T.; YE, S.; ZHANG, K.; et al. A Systematic DNN Weight Pruning Framework using Alternating Direction Method of Multipliers. **6th International Conference on Learning Representations, ICLR 2018 - Workshop Track Proceedings**, v. 11212 LNCS, p. 191–207, 2018. Disponível em: <http://arxiv.org/abs/1802.05747>.

ZHENG, Y.; SUN, P.; REN, Q.; XU, W.; ZHU, D. A novel and efficient model pruning method for deep convolutional neural networks by evaluating the direct and indirect effects of filters. **Neurocomputing**, v. 569, n. December 2023, p. 127124, 2024. Elsevier B.V. Disponível em: <https://doi.org/10.1016/j.neucom.2023.127124>.

ZHONG, Z.; ZHENG, L.; KANG, G.; LI, S.; YANG, Y. Random Erasing Data Augmentation. **Proceedings of the AAAI Conference on Artificial Intelligence**, v. 34, n. 07, p. 13001–13008, 2020. Disponível em: <https://ojs.aaai.org/index.php/AAAI/article/view/7000>.

ZHOU, H.; LAN, J.; LIU, R.; YOSINSKI, J. Deconstructing Lottery Tickets: Zeros, Signs, and the Supermask. Conference on Neural Information Processing Systems (NeurIPS 2019), 2019. Disponível em: <http://arxiv.org/abs/1905.01067>.

ZHOU, Z.-H.; CHAWLA, N. V.; JIN, Y.; WILLIAMS, G. J. Big Data Opportunities and Challenges: Discussions from Data Analytics Perspectives. **IEEE Computational Intelligence Magazine**, v. 9, n. 4, p. 62–74, 2014. Disponível em: <http://www.journalofbigdata.com/content/2/1/1>.

ZOPH, B.; VASUDEVAN, V.; SHLENS, J.; LE, Q. V. Learning Transferable Architectures for Scalable Image Recognition. IEEE conference on computer vision and pattern recognition. p.8697–8710, 2018.

ZUNINO, A.; BARGAL, S. A.; MORERIO, P.; et al. Excitation Dropout: Encouraging Plasticity in Deep Neural Networks. **International Journal of Computer Vision**, v. 129, n. 4, p. 1139–1152, 2021. Springer US. Disponível em: <https://doi.org/10.1007/s11263-020-01422-y>.

APÊNDICE 1 – RESULTADOS COMPLEMENTARES DA CAMADA NULA

As TABELA 15, TABELA 16 e TABELA 17 apresentam o desvio padrão dos resultados numéricos do método CN. Os resultados são de 5 ensaios realizados com as mesmas especificações de treinamento, variando os pesos iniciais.

TABELA 15 – DESVIO PADRÃO PARA OS TESTES DA CAMADA NULA COM ARQUITETURA CNN3

			Tanh	Tanh + CN	CReLU	CReLU + CN	ReLU	ReLU + CN	ELU	ELU + CN	LReLU	LReLU + CN
CNN3	Mnist	Glorot	0,0171	0,0120	0,1043	0,0990	0,1894	0,1476	0,3567	0,0021	0,0375	0,0039
		He	0,0396	0,0089	0,2550	0,0446	0,0000	0,0678	0,0001	0,0132	0,0319	0,0349
		Narrow-normal	0,0428	0,0294	0,0907	0,0472	0,1162	0,1530	0,0038	0,0040	0,1608	0,1506
	Cifar-10	Glorot	0,0099	0,0062	0,0540	0,0159	0,0020	0,0158	0,0325	0,0045	0,0234	0,0117
		He	0,0112	0,0020	0,0339	0,0316	0,0000	0,0339	0,0000	0,0090	0,0040	0,0278
		Narrow-normal	0,0105	0,0087	0,0255	0,0113	0,0213	0,0225	0,0062	0,0061	0,0103	0,0157
	Flower	Glorot	0,0347	0,0133	0,0062	0,0166	0,0000	0,0000	0,0000	0,0241	0,0000	0,0534
		He	0,0290	0,0361	0,0311	0,0311	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000
		Narrow-normal	0,0243	0,0211	0,1305	0,0538	0,0021	0,0623	0,0912	0,0727	0,0265	0,0558
	Fruit	Glorot	0,0291	0,0170	0,2201	0,0975	0,1558	0,2049	0,0128	0,0294	0,5409	0,0007
		He	0,0355	0,0260	0,2447	0,2470	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000
		Narrow-normal	0,0426	0,0331	0,0625	0,1290	0,0484	0,0952	0,0117	0,0122	0,0005	0,0007

FONTE: O autor (2025).

TABELA 16 - DESVIO PADRÃO PARA OS TESTES DA CAMADA NULA COM ARQUITETURA ALEXNET

			Tanh	Tanh + CN	CReLU	CReLU + CN	ReLU	ReLU + CN	ELU	ELU + CN	LReLU	LReLU + CN
AlexNet	Mnist	Glorot	0,0046	0,0062	0,0071	0,0070	0,0016	0,0015	0,0011	0,0027	0,0040	0,0050
		He	0,0056	0,0035	0,0035	0,0034	0,0141	0,0031	0,0189	0,0021	0,0799	0,0032
		Narrow-normal	0,0596	0,0746	0,0914	0,1094	0,0654	0,1510	0,0025	0,0034	0,0193	0,0687
	Cifar-10	Glorot	0,0052	0,0063	0,0108	0,0077	0,0076	0,0148	0,0106	0,0071	0,0116	0,0144
		He	0,0720	0,0047	0,0073	0,0099	0,0596	0,0126	0,0139	0,0116	0,0472	0,0072
		Narrow-normal	0,0210	0,0136	0,0230	0,0211	0,0218	0,0267	0,0056	0,0051	0,0349	0,0274
	Flower	Glorot	0,0259	0,0284	0,0259	0,0231	0,0157	0,0203	0,0201	0,0162	0,0198	0,0127
		He	0,0107	0,0193	0,0183	0,0314	0,0712	0,0135	0,1575	0,0200	0,1252	0,0374
		Narrow-normal	0,0120	0,0158	0,0608	0,0651	0,0294	0,0840	0,0118	0,0093	0,0259	0,0663
	Fruit	Glorot	0,0082	0,0022	0,0065	0,0055	0,0000	0,0011	0,0000	0,0011	0,0000	0,0011
		He	0,0010	0,0016	0,0035	0,0034	0,0148	0,0007	0,0015	0,0000	0,0071	0,0000
		Narrow-normal	0,0071	0,0104	0,0000	0,0005	0,0131	0,0340	0,0052	0,0118	0,3427	0,0005

FONTE: O autor (2025).

TABELA 17 - DESVIO PADRÃO PARA OS TESTES DA CAMADA NULA COM ARQUITETURA VGG19

			Tanh	Tanh + CN	CReLU	CReLU + CN	ReLU	ReLU + CN	ELU	ELU + CN	LReLU	LReLU + CN
VGG19	Mnist	Glorot	0,0011	0,0007	0,0573	0,0081	0,0031	0,0326	0,0021	0,0024	0,0088	0,0582
		He	0,0024	0,0022	0,0028	0,0020	0,2203	0,0015	0,0580	0,0015	0,2664	0,0030
		Narrow-normal	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000
	Cifar-10	Glorot	0,0072	0,0053	0,0365	0,0191	0,0089	0,0312	0,0031	0,0073	0,0618	0,0310
		He	0,0023	0,0095	0,0064	0,0152	0,0589	0,0101	0,1484	0,0075	0,0874	0,0223
		Narrow-normal	0,0024	0,0000	0,0000	0,0000	0,0000	0,0000	0,0007	0,0000	0,0000	0,0000
	Flower	Glorot	0,0151	0,0166	0,0149	0,0000	0,0369	0,0000	0,0095	0,0193	0,0406	0,0083
		He	0,0269	0,0140	0,1945	0,0180	0,2012	0,0219	0,1189	0,0555	0,1884	0,0149
		Narrow-normal	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000
	Fruit	Glorot	0,0018	0,0085	0,0321	0,0000	0,0017	0,0000	0,0000	0,0000	0,0030	0,0000
		He	0,0000	0,0005	0,0000	0,0005	0,2685	0,0005	0,0133	0,0000	0,0780	0,0036
		Narrow-normal	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000

FONTE: O autor (2025).

As TABELA 18, TABELA 19 e TABELA 20 apresentam os valores máximos (Max) e mínimos (Min) entre os 5 ensaios realizados para os testes do método CN.

TABELA 18 – VALORES MÁXIMOS E MÍNIMOS PARA O ALGORITMO CN COM A ARQUITETURA CNN3

			Tanh		Tanh + CN		CReLU		CReLU + CN		ReLU		ReLU + CN		ELU		ELU + CN		LReLU		LReLU + CN	
			Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min
CNN3	Mnist	Glorot	0,7488	0,7099	0,8146	0,7851	0,7594	0,5069	0,8815	0,6491	0,7190	0,2854	0,9129	0,5372	0,9263	0,1865	0,9217	0,9160	0,9257	0,8325	0,9204	0,9100
		He	0,7354	0,6411	0,8569	0,8348	0,7481	0,0975	0,9090	0,8164	0,0980	0,0980	0,9161	0,7428	0,0981	0,0979	0,9316	0,8975	0,8411	0,7511	0,9221	0,8349
		Narrow-normal	0,5153	0,4091	0,6453	0,5733	0,5179	0,2925	0,5485	0,4196	0,8334	0,5478	0,8279	0,4108	0,8479	0,8377	0,8381	0,8296	0,8387	0,4485	0,8357	0,4801
	Cifar-10	Glorot	0,2602	0,2376	0,2894	0,2729	0,2929	0,1510	0,3604	0,3227	0,1017	0,0969	0,3817	0,3386	0,2449	0,1778	0,3838	0,3729	0,2162	0,1662	0,3770	0,3466
		He	0,2498	0,2229	0,2894	0,2850	0,1842	0,1044	0,3134	0,2275	0,1000	0,1000	0,3214	0,2347	0,1000	0,1000	0,3394	0,3186	0,2521	0,2417	0,3251	0,25600
		Narrow-normal	0,2213	0,1932	0,2177	0,1982	0,2222	0,1504	0,2231	0,2015	0,3034	0,2438	0,2888	0,2395	0,3259	0,3129	0,3117	0,2960	0,3073	0,2824	0,2831	0,2453
	Flower	Glorot	0,5370	0,4537	0,5185	0,4907	0,1898	0,1759	0,2130	0,1759	0,1759	0,1759	0,1759	0,1759	0,1759	0,1759	0,2315	0,1759	0,0000	0,0000	0,5093	0,3888
		He	0,4954	0,4259	0,4676	0,3889	0,2454	0,1759	0,2454	0,1759	0,1759	0,1759	0,1759	0,1759	0,1759	0,1759	0,1759	0,1759	0,0000	0,0000	0,0000	0,0000
		Narrow-normal	0,5185	0,4583	0,5231	0,4769	0,4306	0,1759	0,4769	0,3380	0,1806	0,1759	0,3519	0,1759	0,4074	0,1806	0,4954	0,2963	0,4769	0,4167	0,4491	0,3101
	Fruit	Glorot	0,9810	0,9084	0,9881	0,9465	0,8216	0,2913	0,8537	0,6243	0,3841	0,0297	0,6885	0,2366	0,0583	0,0297	1,000	0,9334	0,9929	0,0000	1,000	0,9988
		He	0,9738	0,8775	0,9941	0,9394	0,7658	0,1403	0,7562	0,0892	0,0297	0,0297	0,0297	0,0297	0,0297	0,0297	0,0297	0,0297	0,0000	0,0000	0,0000	0,0000
		Narrow-normal	0,9227	0,8098	0,9298	0,8502	0,7860	0,6278	0,8561	0,5054	0,7515	0,6576	0,9417	0,7170	1,000	0,9738	1,000	0,9727	1,000	0,9988	1,000	0,9988

FONTE: O autor (2025).

TABELA 19 – VALORES MÁXIMOS E MÍNIMOS PARA O ALGORITMO CN COM A ARQUITETURA ALEXNET

			Tanh		Tanh + CN		CReLU		CReLU + CN		ReLU		ReLU + CN		ELU		ELU + CN		LReLU		LReLU + CN	
			Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min
AlexNet	Mnist	Glorot	0,8699	0,8591	0,8809	0,8662	0,8962	0,8770	0,9019	0,8862	0,9449	0,9416	0,9343	0,9305	0,9555	0,9527	0,9542	0,9471	0,9458	0,9347	0,9373	0,9248
		He	0,8649	0,8516	0,8777	0,8692	0,9147	0,9060	0,9211	0,9121	0,8652	0,8307	0,9464	0,9384	0,9464	0,8989	0,9585	0,9526	0,9163	0,7214	0,9449	0,9369
		Narrow-normal	0,5445	0,3987	0,5963	0,4303	0,2974	0,1010	0,3599	0,1012	0,7554	0,5927	0,4689	0,1028	0,9108	0,9045	0,8998	0,8920	0,7647	0,7193	0,5567	0,3828
	Cifar-10	Glorot	0,3580	0,3455	0,3775	0,3627	0,3387	0,3104	0,3284	0,3092	0,3553	0,3361	0,3496	0,3115	0,4714	0,4451	0,4300	0,4134	0,3747	0,3450	0,3440	0,3058
		He	0,3054	0,1462	0,3554	0,3436	0,3483	0,3296	0,3357	0,3097	0,3176	0,1675	0,3122	0,2803	0,4080	0,3751	0,4016	0,3692	0,3319	0,2266	0,3115	0,2969
		Narrow-normal	0,2040	0,1522	0,2478	0,2164	0,1643	0,1019	0,2011	0,1507	0,1939	0,1367	0,1602	0,1001	0,3211	0,3063	0,2966	0,2835	0,1810	0,1057	0,2102	0,1434
	Flower	Glorot	0,6065	0,5509	0,6343	0,5556	0,5972	0,5417	0,6481	0,5833	0,6713	0,6296	0,6759	0,6296	0,7315	0,6852	0,7269	0,6852	0,6898	0,6389	0,6944	0,6620
		He	0,5926	0,5694	0,6065	0,5556	0,5741	0,5324	0,6157	0,5324	0,4167	0,2407	0,5833	0,5509	0,6111	0,1806	0,6898	0,6343	0,5880	0,2917	0,6389	0,5416
		Narrow-normal	0,3981	0,3657	0,4444	0,4028	0,3611	0,2269	0,3750	0,2361	0,5278	0,4491	0,4444	0,2269	0,6759	0,6481	0,6250	0,6019	0,5417	0,4861	0,4676	0,3055
	Fruit	Glorot	0,9679	0,9465	0,9631	0,9572	0,9715	0,9548	0,9750	0,9631	1,000	1,000	1,000	0,9976	1,000	1,000	1,000	0,9976	1,000	1,000	1,000	0,9976
		He	0,9881	0,9857	0,9905	0,9869	0,9929	0,9845	0,9893	0,9810	1,000	0,9655	1,000	0,9988	1,000	0,9964	1,000	1,000	1,000	0,9834	1,000	1,000
		Narrow-normal	0,0749	0,0583	0,1130	0,0844	0,0583	0,0583	0,0595	0,0583	0,0844	0,0583	0,1344	0,0583	0,9477	0,9334	0,9382	0,9084	0,8395	0,0583	0,0595	0,0582

FONTE: O autor (2025).

TABELA 20 – VALORES MÁXIMOS E MÍNIMOS PARA O ALGORITMO CN COM A ARQUITETURA VGG19

			Tanh		Tanh + CN		CReLU		CReLU + CN		ReLU		ReLU + CN		ELU		ELU + CN		LReLU		LReLU + CN	
			Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min
VGG19	Mnist	Glorot	0,9126	0,9099	0,9100	0,9083	0,2279	0,1028	0,1320	0,1135	0,9632	0,9548	0,1881	0,1135	0,9694	0,9639	0,9607	0,9549	0,9462	0,9227	0,2402	0,1009
		He	0,9415	0,9359	0,9435	0,9386	0,9456	0,9389	0,9476	0,9423	0,4865	0,0000	0,9633	0,9600	0,1135	0,0000	0,9725	0,9684	0,6324	0,0000	0,9577	0,9509
		Narrow-normal	0,1135	0,1135	0,1135	0,1135	0,1135	0,1135	0,1135	0,1135	0,1135	0,1135	0,1135	0,1135	0,1135	0,1135	0,1135	0,1135	0,1135	0,1135	0,1135	0,1135
	Cifar-10	Glorot	0,4115	0,3928	0,4322	0,4186	0,2111	0,1144	0,1441	0,1000	0,3072	0,2866	0,1938	0,1156	0,4961	0,4876	0,4718	0,4517	0,2462	0,1029	0,2062	0,122
		He	0,4359	0,4301	0,4375	0,4149	0,3612	0,3463	0,3684	0,3292	0,2348	0,0988	0,3517	0,3302	0,4510	0,1000	0,4561	0,4371	0,3116	0,1367	0,3603	0,3026
		Narrow-normal	0,1008	0,0950	0,1000	0,1000	0,1000	0,1000	0,1000	0,1000	0,1000	0,1000	0,1000	0,1000	0,1015	0,1000	0,1000	0,1000	0,1000	0,1000	0,1000	0,1000
	Flower	Glorot	0,6250	0,5880	0,6620	0,6204	0,5231	0,4861	0,2454	0,2454	0,5509	0,4537	0,2454	0,2454	0,6806	0,6620	0,7176	0,6713	0,5231	0,4259	0,2454	0,2268
		He	0,6620	0,5972	0,6620	0,6250	0,4398	0,0000	0,6389	0,5926	0,4444	0,0000	0,6620	0,6111	0,2454	0,0000	0,7083	0,5602	0,4213	0,0000	0,6343	0,5972
		Narrow-normal	0,2454	0,2454	0,2454	0,2454	0,2454	0,2454	0,2454	0,2454	0,2454	0,2454	0,2454	0,2454	0,2454	0,2454	0,2454	0,2454	0,2454	0,2454	0,2454	0,2453
	Fruit	Glorot	1,000	0,9952	0,9988	0,9798	0,1356	0,0583	0,0583	0,0583	0,9988	0,9952	0,0583	0,0583	1,000	1,000	1,000	1,000	1,000	0,9929	0,0583	0,0582
		He	1,000	1,000	1,000	0,9988	1,000	1,000	1,000	0,9988	0,9976	0,3876	1,000	0,9988	1,000	0,9703	1,000	1,000	1,000	0,8276	1,000	0,9916
		Narrow-normal	0,0583	0,0583	0,0583	0,0583	0,0583	0,0583	0,0583	0,0583	0,0583	0,0583	0,0583	0,0583	0,0583	0,0583	0,0583	0,0583	0,0583	0,0583	0,0583	0,0582

FONTE: O autor (2025).