UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E BIOLÓGICAS
DEPARTAMENTO DE COMPUTAÇÃO

GABRIEL BICALHO FERREIRA
Orientador: Prof. Dr. Rodrigo Cesar Pedrosa Silva

COMPRESSÃO DE REDES NEURAIS ARTIFICIAIS COM RESTRIÇÕES DE ACURÁCIA UTILIZANDO OTIMIZAÇÃO MULTIOBJETIVO BASEADA EM MODELOS SUBSTITUTOS

Ouro Preto, MG
2022

UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E BIOLÓGICAS
DEPARTAMENTO DE COMPUTAÇÃO

GABRIEL BICALHO FERREIRA

COMPRESSÃO DE REDES NEURAIS ARTIFICIAIS COM RESTRIÇÕES DE ACURÁCIA UTILIZANDO OTIMIZAÇÃO MULTIOBJETIVO BASEADA EM MODELOS SUBSTITUTOS

Monografia apresentada ao Curso de Ciência da Computação da Universidade Federal de Ouro Preto como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Rodrigo Cesar Pedrosa Silva

Ouro Preto, MG
2022

SISBIN - SISTEMA DE BIBLIOTECAS E INFORMAÇÃO

F383c Ferreira, Gabriel Bicalho.
compressão de redes neurais artificiais com restrições de acurácia utilizando otimização multiobjetivo baseada em modelos substitutos. [manuscrito] / Gabriel Bicalho Ferreira. - 2022.
38 f.: il.: color., gráf., tab..

Orientador: Prof. Dr. Rodrigo Cesar Pedrosa Silva.
Monografia (Bacharelado). Universidade Federal de Ouro Preto. Instituto de Ciências Exatas e Biológicas. Graduação em Ciência da Computação .

1. Compressão de Redes Neurais. 2. Aprendizado Profundo. 3. Redes Neurais Convolucionais. 4. Otimização Multiobjetivo. 5. Problemas Substitutos. I. Silva, Rodrigo Cesar Pedrosa. II. Universidade Federal de Ouro Preto. III. Título.

CDU 004

Bibliotecário(a) Responsável: Luciana De Oliveira - SIAPE: 1.937.800

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE OURO PRETO
REITORIA
INSTITUTO DE CIENCIAS EXATAS E BIOLOGICAS
DEPARTAMENTO DE COMPUTACAO

FOLHA DE APROVAÇÃO

Gabriel Bicalho Ferreira

Compressão de Redes Neurais Artificiais com Restrições de Acurácia utilizando Otimização Multiobjetivo baseada em modelos substitutos

Monografia apresentada ao Curso de Ciência da Computação da Universidade Federal
de Ouro Preto como requisito parcial para obtenção do título de Bacharel em Ciência da Computação

Aprovada em 27 de Outubro de 2022.

Membros da banca

Rodrigo César Pedrosa Silva (Orientador) - Doutor - Universidade Federal de Ouro Preto
Jadson Castro Gertrudes (Examinador) - Doutor - Universidade Federal de Ouro Preto
Guilherme Augusto Lopes Silva (Examinador) - Bacharel - Universidade Federal de Ouro Preto

Rodrigo César Pedrosa Silva, Orientador do trabalho, aprovou a versão final e autorizou seu depósito na Biblioteca Digital de Trabalhos de Conclusão de Curso da UFOP em 27/10/2022.

Documento assinado eletronicamente por **Rodrigo Cesar Pedrosa Silva**, **PROFESSOR DE MAGISTERIO SUPERIOR**, em 27/10/2022, às 16:45, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 , informando o código verificador **0416025** e o código CRC **F32E382B**.

R. Diogo de Vasconcelos, 122, - Bairro Pilar Ouro Preto/MG, CEP 35400-000
Telefone: 3135591692 - www.ufop.br

Agradecimentos

Em primeiro lugar, quero agradecer a Deus por ter me dado saúde, força e foco, me guiando por todos esse anos e colocando pessoas boas em minha jornada.

Agradeço infinitamente a toda minha família, por apoiarem em minhas decisões e estar ao meu lado nos momentos mais difíceis. Em especial, gostaria de agradecer meu pai José Ferreira dos Santos, minha mãe Maria Aparecida Bicalho Ferreira e meu irmão Thiago Bicalho Ferreira.

Ao professor Rodrigo, pela excelente orientação e confiança, que prontamente, compartilhou suas experiências e conhecimentos, possibilitando a realização deste trabalho.

Agradeço também, aos meu amigos, Lucas Andrade, Vinícius de Paula, Halliday Gauss, Carlos Magalhães, Willian Gomes, Emanuel Jesus, Marcos Pontes e Vinícius Targa, pela ótima convivência e companheirismo, tornando a experiência de morar longe de casa muito mais agradável.

Sou imensamente grato, a instituição UFOP, pelo meu processo de formação e a todos os professores do departamento de computação da UFOP (DECOM), que me permitiram ter uma ótima bagagem de conhecimento durante o curso.

As experiências que obtive durante a graduação, me tornaram uma pessoa melhor, capaz de correr atrás de meus objetivos.

Resumo

Dentre os diferentes métodos de aprendizado de máquina, as redes neurais vem se destacando nos últimos anos, devido a sua alta precisão na resolução de diversos tipos de problemas. O aumento de camadas e neurônios de uma rede neural é diretamente proporcional a sua eficiência, visto que permite às redes neurais artificiais (RNA) o reconhecimento de mais detalhes sobre o dado de entrada. No enatando, redes complexas resultam em grandes dificuldades na implantação em dispositivos com restrições de processamento e memória. Sendo assim, técnicas de compressão que visam a redução da complexidade da rede neural sem perda significativa na eficiência, estão ganhando cada vez mais notoriedade na literatura. Neste contexto, o presente trabalho visa parametrizar técnicas de compressão por poda e quantização, verificando os efeitos dos parâmetros escolhidos nos objetivos propostos.

Palavras-chave: Compressão de Redes Neurais. Aprendizado Profundo. Redes Neurais Convolucionais. Otimização Multiobjetivo. Problemas Substitutos.

Abstract

Among the different methods of machine learning, neural networks have been standing out in recent years, due to their high precision in solving different types of problems. The increase in layers and neurons of a neural network is directly proportional to its efficiency, since it allows artificial neural networks (ANN) to recognize more details about the input data. However, complex networks result in great difficulties in deploying them on devices with processing and memory restrictions. Thus, compression techniques that aim to reduce the neural network complexity without significant loss in efficiency are gaining more and more notoriety in the literature. In this context, the present work aims to parameterize compression techniques by pruning and quantization, verifying the effects of the chosen parameters on the proposed objectives.

Keywords: Compression of Neural Networks. Deep Learning, Convolutional Neural Networks. Multi-Objective Optimization. Surrogates Problems.

Lista de Ilustrações

Lista de Tabelas

Sumário

1 Introdução

O aprendizado de máquina é uma área da inteligência artificial que visa desenvolver sistemas computacionais capazes de adquirir conhecimento através de experiências acumuladas e tomar decisões de forma automática (MONARD; BARANAUSKAS, 2003). Estes sistemas aprendizado podem ser utilizados na resolução de diversos tipos de problemas, desde os mais simples como verificação se um e-mail é spam, e reconhecimento de letras em manuscrito, até problemas mais robustos como detecção de face em câmeras de vigilância, entre outros (TONIN, 2021).

Dentre os métodos de aprendizagem de máquina, as redes neurais vem se destacando nos últimos anos, devido a sua alta precisão na resolução de diferentes tipos de problemas, principalmente envolvendo reconhecimento de padrões e classificação de imagens. Uma rede neural, visa imitar, de certa forma, o funcionamento do sistema nervoso humano através de sistemas computacionais (TONIN, 2021). Sua representação envolve neurônios artificiais que que são unidades altamente interconectadas (MONARD; BARANAUSKAS, 2003).

Em (POUYANFAR et al., 2018) é realizado um estudo comparativo sobre a utilização do aprendizado profundo em diferentes problemas e contextos. Os autores constataram que as redes neurais convolucionais (CNNs) demonstraram resultados significativos em diferentes tarefas do mundo real para processamento de dados visuais, como detecção de objetos, processamento de imagens e vídeo. Além disso, arquiteturas convolucionais foram um sucesso quando testadas em problemas relacionados ao processamento de linguagem natural, que envolvia tarefas como análise de sentimentos e tradução de idiomas. De acordo com a revisão apresentada pelos autores, os métodos de aprendizado profundo mostrou uma enorme capacidade de serem aproveitados em diversas aplicações. Nesse contexto, as redes neurais convolucionais (RNAs) estão sendo utilizadas cada vez mais em problemas robustos, implicando no surgimento de modelos mais complexos com milhares de neurônios e camadas, que exigem um maior espaço de armazenamento e processamento computacional (TONIN, 2021).

Os últimos avanços nessas técnicas, proporcionou um aumento na precisão das RNAs e também um crescimento exponencial destas redes, aumentando sua complexidade (FERREIRA, 2022). Porém, estruturas complexas implicam em uma grande dificuldade de implantação em dispositivos edge como celulares e computadores com poucos recursos de processamento e memória (WANG et al., 2021). Nessa perspectiva, uma área de pesquisa que está ganhando cada vez mais notoriedade, são as técnicas de compressão de redes neurais. Essas técnicas visam reduzir o custo computacional de uma rede com perdas desprezíveis em sua eficácia, o que possibilita a implementação desses modelos em diferentes tipos de dispositivos com restrições de recursos. Na compressão, as técnicas mais utilizadas são a poda, que anulam os pesos mais

insignificantes de forma a remover uma conexão (GOLDBARG, 2021) e a quantização, que reduz o número de bits na representação de cada parâmetro selecionado na rede (HAN; MAO; DALLY, 2015).

CHEN; RAN (2019) fornece uma revisão abrangente sobre o estado atual da inserção do aprendizado profundo na computação de borda, discutindo algumas abordagens para essa aplicação. Os autores, relatam a inviabilidade da realização da inferência e treinamento no próprio dispositivo edge, devido ao alto custo computacional requerido. No trabalho, é constatado que a abordagem mais comum para resolução deste tipo de problema, seria mover os mover os dados desses dispositivos para um local centralizado na nuvem. No entanto, essa abordagem enfrenta inúmero desafios, como a latência que pode ser crítica para muitos dispositivos, escalabilidade e privacidade dos dados.

Neste contexto, BLALOCK et al. (2020) realizam uma análise sobre o efeito da poda na compressão de redes neurais fazendo uma investigação em 81 artigos com a finalidade de comparar diferentes estratégias de poda e seus efeitos na rede neural. Os autores relataram, que as técnicas avaliadas, são baseadas no tamanho do modelo, acurácia, velocidade de inferência da rede e a escolha de um dataset.

TONIN (2021) realiza um estudo sobre a compressão de redes neurais, mais especificamente sobre a utilização da técnica de quantização em uma rede previamente treinada. A análise é feita com base em 3 formas de quantização (uniforme, não uniforme ou deadzone) aplicada a 10 diferentes tipos de modelos RNA. Com os testes, os autores relataram ser possível comprimir uma rede em até 8 vezes com o prejuízo menor que 0.8% em sua eficácia.

Por sua vez, no trabalho de (FERNANDES; KUNG, 2021) é utilizado as duas técnicas de compressão de dados (Poda e Quantização). Os autores adotam duas estratégias de treino para cada época, a primeira sendo a realização de uma poda seguida pela quantização e a segunda a quantização seguida pela poda. Com a análise dos resultados, os autores constataram que a segunda abordagem é mais eficiente para o problema de comprimir a rede com perda mínima de eficácia.

Devido a inúmeras possibilidades de poda e quantização, WANG et al. (2021) acrescenta a otimização multiobjetivo para escolha dos parâmetros gerando as possíveis combinações das técnicas. A abordagem proposta é dividida em duas etapas com a finalidade de separar os processos de poda e quantização. Na primeira etapa, é montada uma biblioteca de redes podadas com intensidades variando de 0 a 95%, realizando o retreino em 1 época a cada nível. Na segunda etapa, consiste na aplicação da otimização multiobjetivo gerando combinações entre as redes criadas na etapa anterior com a aplicação da quantização, considerando o par de objetivos, acurácia e consumo energético ou acurácia e tamanho do modelo. Quanto aos resultados, os autores relataram que é possível reduzir em até 80% o consumo de energia, se for considerada uma perda de precisão de 2%.

Nessa mesma perspectiva, FERREIRA (2022) realiza um estudo sobre a utilização da otimização multiobjetivo na compressão de redes neurais com o retreino. Os autores afirmaram que o retreino de uma rede após a compressão é uma alternativa aconselhável, possibilitando a recuperação da rede em detrimento a algumas perdas. No entanto, treinar uma rede é algo extremamente custoso, tendo em vista que demanda tempo e recursos computacionais. Para contornar esse problema, foi proposto a realização da otimização com base em modelos substitutos, que possuem comportamento similar ao problema original com uma complexidade de tempo bem inferior. Os resultados do trabalho demonstraram que os parâmetros selecionados pelo algoritmo otimizador eram diferentes para arquiteturas distintas. Sendo assim, os autores concluíram que o conjunto ótimo depende da arquitetura selecionada, visto que os parâmetros de compressão estão fortemente ligados à estrutura da rede neural comprimida.

Diante da revisão apresentada, é notável que as técnicas de compressão de dados por poda e quantização são uma área de estudo recente e que podem ser utilizadas em diferentes contextos e estratégias.

Técnicas de compressão atuam diretamente sobre a estrutura de uma rede neural. Sendo assim, a compressão pode gerar impactos negativos em uma RNA. A escolha dos parâmetros de poda e quantização de forma automática, ainda é pouco abordada na literatura. Os trabalhos que abordam essa temática, utilizam a otimização multiobjetivo porém, não levam em consideração quaisquer tipos de restrições, podendo gerar como solução final um conjunto de soluções que possua redes com grande taxa de compressão mas ineficientes. Além disso, a falta de restrições faz com que o algoritmo otimizador avalie muitas soluções ruins, desperdiçando recursos computacionais. Como exemplo, é possível citar o trabalho FERREIRA (2022), em que os autores não utilizam restrições, ocasionado na avaliação de diversos modelos inviáveis com acurácia inferior a 20%, como demonstrado nos resultados.

Nesse contexto, o presente trabalho visa o desenvolver que um algoritmo para escolha dos parâmetros de poda e quantização de forma automática, por meio de um algoritmo otimizador baseado em restrições de acurácia, considerando que o conjunto de parâmetros ótimos depende da arquitetura.

1.1 Objetivos

O objetivo principal deste trabalho é apresentar um algoritmo eficiente para otimizar os parâmetros dos algoritmos de compressão que leve em consideração restrições de acurácia.

Para cumprir o objetivo principal, é necessário dividi-lo nos seguintes objetivos específicos.

1. Parametrizar os algoritmos de compressão;
2. Verificar se as variáveis escolhidas de fato têm efeito nos objetivos;

3. Propor o algoritmo:

    - Seleção de modelos substitutos;
    - Estratégia de seleção de modelos para as restrições;
    - Estratégia de seleção de modelos para os objetivos.

2 Compressão de Redes Neurais Artificiais e Otimização multi-objetivo baseada em substitutos

2.1 Redes Neurais e Deep Learning

Redes neurais artificiais (RNAs) são sistemas massivos e paralelos, constituídos por várias células computacionais simples que executam uma determinada função matemática (GORGENS et al., 2009). De forma geral, uma RNA é um algoritmo de aprendizado de máquina, inspirado na forma como o cérebro resolve uma determinada tarefa (FLECK et al., 2016).

As células computacionais simples, comumente denominadas como neurônios, são compostas por 5 elementos básicos, primeiro o vetor de entrada $\mathbf{x_m}$, o conjunto de pesos $W = w_{11}$, w_{12}, ... w_{km} onde k é o número de neurônios em cada camada e m o tamanho da entrada. O termo de bias é representado por b_k, a função de ativação por $\varphi(.)$ e por último a saída do neurônio y_k. A Figura 2.1, representa a estrutura de um neurônio artificial.

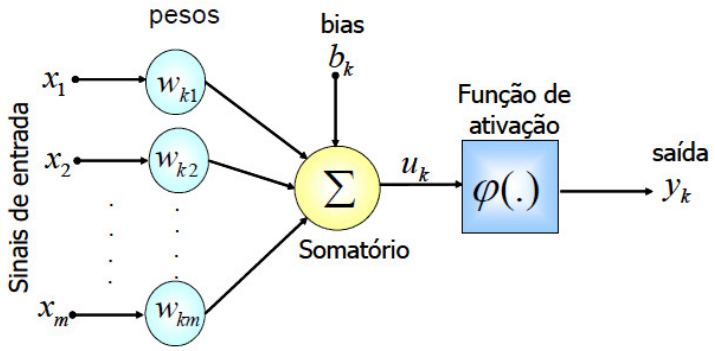

Figura 2.1 – Neurônio Artificial (<https://bit.ly/neuronio_artificial>)

Outra forma de representação, é utilizando apenas conceitos matemáticos, a equação 2.2 adaptada de (Data Science Academy, 2022) define de forma geral a saída de um neurônio.

$$(\sum_{i=1}^{m} x_i w_{k_i}) + b_k = S_k \tag{2.1}$$

$$\varphi(S_k) = y_k \tag{2.2}$$

.

Existem diferentes tipos RNAs, como por exemplo as redes neurais recorrentes (RNN), Autoencoders (AE), Perceptron Multicamadas (MLP) e as redes Neurais convolucionais (CNNs).

Todas essas arquiteturas, utilizam inúmeros neurônios que geralmente estão distribuídos em várias camadas. De forma geral, podemos classificar as camadas de uma RNA como sendo, camada de entrada, camadas ocultas (intermediárias), e camada de saída. Atrelado a uma RNA, existe o conceito de Deep learnig, que segundo o Data Science Academy (2022) é uma sub-área do aprendizado de máquina, cuja a finalidade é empregar algoritmos que tomem decisões de forma semelhante a um cérebro humano. As redes neurais profundas possuem no mínimo duas camadas ocultas e são capazes de extrair níveis mais altos de características da entrada de dados (WANG et al., 2021).

A Figura 2.2 é um exemplo básico de uma RNA perceptron multicamadas. A primeira camada, é responsável por receber os dados de entrada e repassar para as próximas camadas, onde serão realizadas dois tipos de transformações: uma linear representada pelo somatório da multiplicação do vetor de pesos pelo vetor de entrada, e outra não linear por meio da função de ativação (Data Science Academy, 2022), como demonstrado na equação 2.5. Essas transformações são realizadas nas camadas ocultas com a finalidade de detectar características úteis sobre o conjunto de dados de entrada. As informações provenientes da saída dos neurônios, são propagadas até a última camada, que determina a classe em que o vetor de dados de entrada pertence.

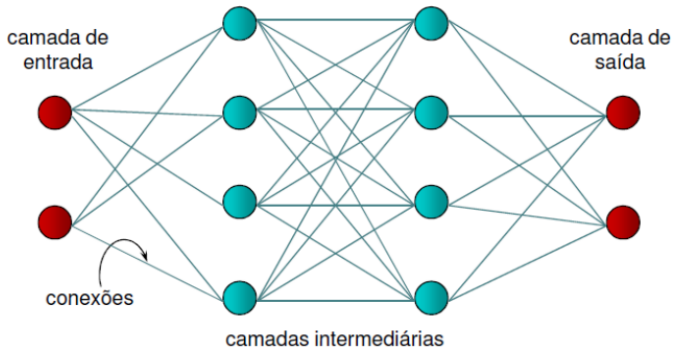

Figura 2.2 – Perceptron Multicamadas (adaptado de SOARES; SILVA (2011))

A principal divergência entre os tipos de arquiteturas de uma RNA, é como os dados fluem pela rede, mais especificamente, como as conexões entre as camadas são feitas. Na rede de perceptrons, como demonstrado na Figura 2.2, todos os neurônios de uma camada se conectam a todos neurônios da próxima camada.

Na próxima seção será apresentado sobre as redes neurais convolucionais, que é o foco desse trabalho, devido à sua eficácia em problemas de classificação de imagens.

2.1.1 Redes Neurais Convolucionais (CNNs)

As redes neurais convolucionais, são redes profundas que podem ser utilizadas para classificar imagens com base em características semelhantes, realizando operações de convolução (Data Science Academy, 2022). São arquiteturas treináveis com múltiplos estágios, em que a

entrada de cada estágio é um mapa de características (matriz) representando um recurso específico do dado de entrada (LECUN; KAVUKCUOGLU; FARABET, 2010).

Nessa arquitetura, além de camadas totalmente conectadas, existem camadas de convolução e *pooling*. A função de uma camada convolucional, é aplicar máscaras nas matrizes de entrada considerando uma determinada vizinhança pré-definida, produzindo mapas de características que armazenam os pesos das conexões entre os neurônios (SANTOS et al., 2017). A camada de *pooling*, atua separadamente sobre cada saída de uma convolução, provocando uma redução de dimensionalidade, através de uma função matemática, gerando um mapa de características robusto a pequenas varações na localização de padrões (LECUN; KAVUKCUOGLU; FARABET, 2010).

A Figura 2.3, apresenta uma rede convolucional. Na camada de entrada, temos uma imagem, que é interpretada como um tensor de pixels e repassada para as próximas camadas. A partir do tensor de entrada, inicia-se o processo de convolução seguido pela aplicação de uma função de ativação não linear. Logo após, é realizado uma subamostragem na camada de pooling, reduzindo o tamanho dos mapas. Com a realização dessas três etapas, tem-se um estágio completo de uma rede neural convolucional, que pode possuir diversos estágios (JURASZEK et al., 2014). Por fim, uma CNN possui camadas densas totalmente conectadas igualmente a um perceptron multicamadas, como descrito na seção anterior.

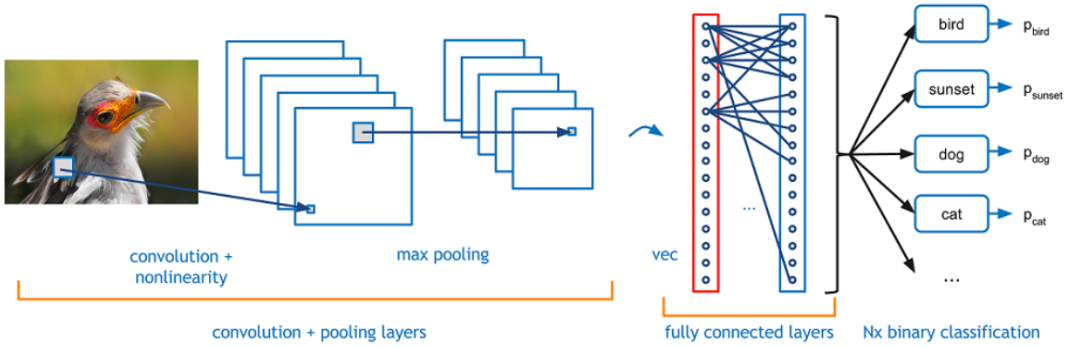

Figura 2.3 – Rede Neural Convolucional ((Data Science Academy, 2022))

O processo de convolução representado pela Figura 2.4, considera a entrada como uma matiz 6x6 e a máscara 3x3. Além desses dois elementos, uma convolução possui um parâmetro de *stride*, que indica quantos passos a máscara é movida ao longo da entrada e o preenchimento (*padding*) que indica o número a ser colocado caso a entrada não tenha elementos suficientes (POINTER, 2019). No exemplo, o *stride* é equivale a 1 e não é necessário valor de *padding*.

O processo de *pooling* exemplificado pela Figura 2.5, utiliza uma mascara de 2x2. De forma idêntica a uma convolução, a máscara é arrastada sobre a matriz de características. No entanto, nessa fase temos a aplicação de uma função matemática utilizada para destacar os pontos mais importantes da entrada. No exemplo foi utilizado uma função *maxpool*, que pega o maior

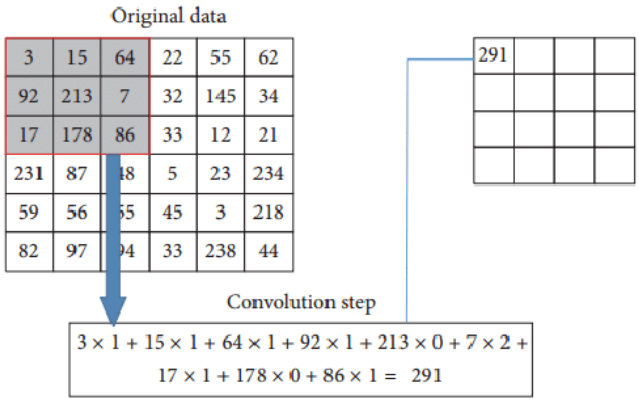

Figura 2.4 – Processo de convolução (MARIED et al. (2017))

elemento dentro da região delimitada pela máscara.

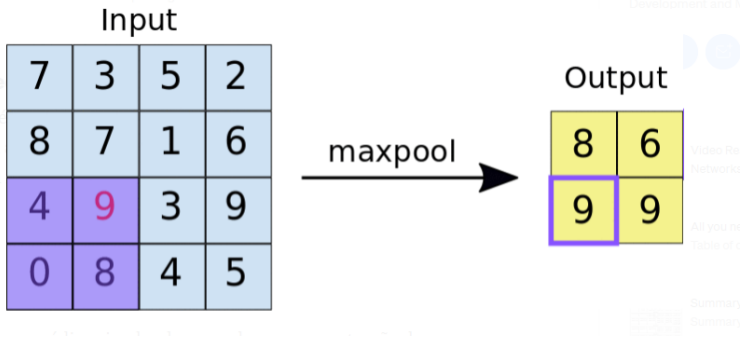

Figura 2.5 – Processo de pooling (<https://bit.ly/medium_pooling>)

Como demonstrado nesta Seção, por meio de processos de convolução e *pooling*, as CNNs conseguem ao longo de suas camadas detectar padrões relevantes sobre os dados de entrada. Mostrando eficiência na resolução de diversos tipos de problemas como, detecção de pedestres (VARGAS; PAES; VASCONCELOS, 2016), detecção de lesões na pele (BAPTISTA, 2021), auxilio no diagnóstico da COVID-19 (LUZ et al., 2022) e diversos outros tipos de problemas encontrados na literatura.

2.2 Compressão de uma Rede Neural

Como descrito na seção anterior, as redes neurais artificiais profundas são estruturas robustas, complexas e eficazes para resolução de diversos tipos de problemas. Apesar dessa eficácia, uma rede neural exige um alto custo computacional, dificultando a implantação em diversos tipos de dispositivos (WANG et al., 2021). Nessa perspectiva, existem técnicas de compreensão que visam reduzir o custo computacional dessas redes. Nesta seção, serão abordadas sobre as duas técnicas de compressão mais comuns na literatura.

2.2.1 Poda

A poda de uma rede neural, consiste na técnica de remover parâmetros insignificantes da rede, definindo seus valores como zero, a partir de um limiar que indica o quão intensa será a compreensão (GOLDBARG, 2021). De acordo com WANG et al. (2021), em uma rede bem treinada, existem inúmeros parâmetros entre pesos e bias, cujos valores são pequenos e que na maioria dos casos, não são importantes para realização da inferência do modelo. Sendo assim, uma boa técnica de remoção durante o processo de poda, seria remover os parâmetros mais insignificantes, visando manter a eficácia da rede.

A Figura 2.6, representa o processo de poda em que 50% das conexões são removidas.

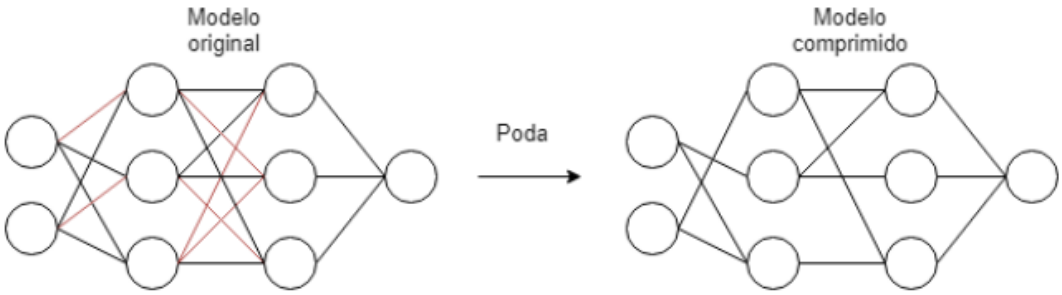

Figura 2.6 – Técnica de poda (GOLDBARG (2021)

Como demonstrado anteriormente, é possível expressar a conexão entre neurônios através da equação 2.3. Com a técnica de poda, essa conexão é removida anulando os valores dos pesos e número de bias.

Neste projeto, será utilizado o *framework pytorch* em que as técnicas de poda são implementadas pelo modulo *nn.tils* da biblioteca *Torch*. Com base na implementação desta função, existem alguns aspectos relevantes a serem considerados, são eles:

- **Frequência da Poda:** Determina o percentual de conexões que serão removidas de uma determinada camada;
- **Módulo:** Camada específica a ser podada, que em uma CNN pode ser uma camada convolucional ou uma camada totalmente conectada;
- **Parâmetro de Remoção:** Indica o parâmetro a ser removido da conexão, podendo ser o valor de bias ou pesos.

Como a poda remove conexões entre os neurônios de um respectivo modelo treinado, a rede comprimida pode sofrer alterações negativas em relação a sua eficácia. Nessa perspectiva, é indicado treinamento em paralelo com o processo de poda, ou um treinamento após a compressão do modelo.

2.2.2 Quantização

A quantização de uma rede neural artificial, é uma técnica de compressão de modelos. Esse método, reduz o número de bits necessários para representar cada parâmetro de uma RNA (HAN; MAO; DALLY, 2015). Na maioria dos casos, os parâmetros de uma RNA são representados por pontos flutuantes de 32 bits, o que aumenta o custo de armazenamento e complexidade dos cálculos (TONIN, 2021). Além dos benefícios de tornar o modelo mais rápido e diminuir o custo de armazenamento, os parâmetros de baixa precisão ainda podem armazenar informações suficientes para inferência do modelo, mantendo sua precisão (WANG et al., 2021).

A Figura 2.7, exemplifica o processo de quantização, em que uma matriz 3x3 possui elementos do tipo float 32 bits é transformada em uma matriz do tipo inteiro 8 bits. Esse é o processo que ocorre nos parâmetros de uma RNA.

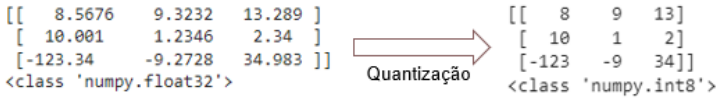

Figura 2.7 – Quantização

Neste trabalho, a técnica de quantização utilizada está implementada no módulo quantization da biblioteca Torch. De acordo com a documentação (QUANTIZATION, 2022), a função implementada comprime a rede em até quatro vezes, quantizando os parâmetros do tipo float de 32 bits, para inteiros de 8 bits.

2.3 Otimização Multi-objetivo

Como demonstrado na Seção 2.2, a compressão é baseada na escolha de diversos parâmetros, como por exemplo, a frequência da poda, camada a ser podada, parâmetro de remoção, entre outros. Uma forma de automatizar a escolha desses parâmetros, é por meio de técnicas de otimização.

Na área de otimização, existem problemas com mais de um objetivo a ser considerado. De forma geral, esses objetivos são conflitantes entre si, de tal modo, que melhorar um objetivo pode prejudicar outro (FERREIRA, 2022). Essas são as caraterísticas de um problema multiobjetivo, em que raramente uma solução atenderá todos os critérios simultaneamente (SAMPAIO, 2011).

Um problema de otimização multiobjetivo pode ser definido de forma generalizada por:

$$minimizar f(\mathbf{x}) \in Y \subset R^m \tag{2.3}$$

$$sujeito : x \in R \tag{2.4}$$

Neste caso, Y representa o conjunto de n objetivos do problema, [$f_1(\mathbf{x})$, $f_2(\mathbf{x})$, ..., $f_n(\mathbf{x})$], contido no espaço R^m destinado aos objetivos, que estão sujeitos a um conjunto de restrições R, podendo ser de desigualdade ou igualdade.

A Figura 2.8, apresenta as possíveis soluções para um problema de minimização entre duas funções, $f_1(x) = (x-4) * (x-4)$ e $f_2(x) = (x-1) * (x-1)$. Para esse problema, foi considerado como a região factível x $\in [-5, 5] \subset Z$. Este, é um problema clássico de otimização multiobjetivo, em que não existe apenas uma solução correta, visto que melhorar um objetivo pode prejudicar outro.

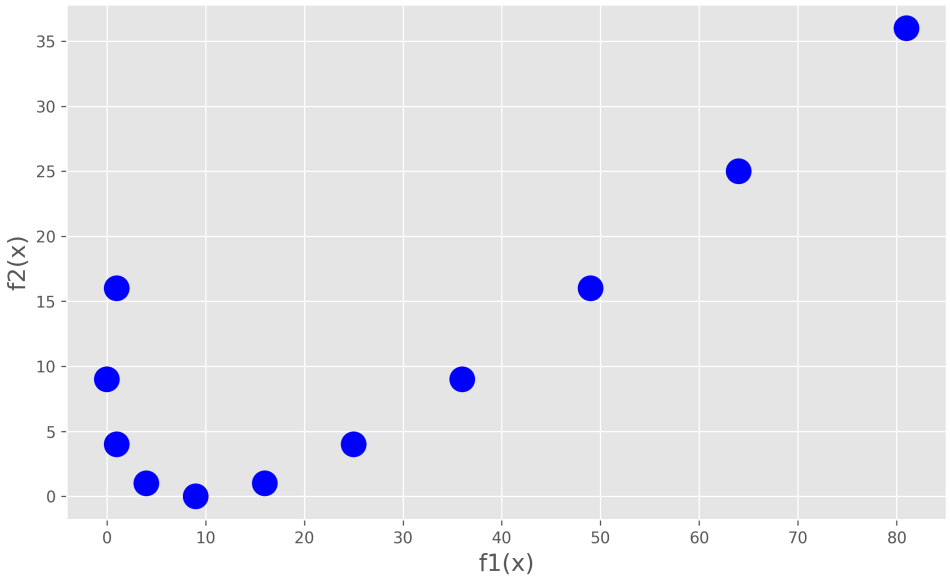

Figura 2.8 – Problema Multiobjetivo

Nesta perspectiva, a meta em otimização multiobjetivo é encontrar um conjunto de soluções que representa bons conflitos de escolhas *(trade-offs)* (SILVA, 2018). Problemas desta natureza, são encontrados em grande quantidade na vida real e podem ser modelados facilmente em funções matemáticas, um exemplo, seria o problema de custo x benefício.

Segundo Silva (2018), a saída de um algoritmo multiobjetivo é um conjunto de soluções não dominadas, em que cada uma dessas soluções seguem a definição:

Dado as soluções S_1 e S_2 que possuem n objetivos:

$$S_1 = [U_1, U_2, ..., U_n]$$

$$S_2 = [V_1, V_2, ..., V_n]$$

S_1 domina S_2 se:

$$U \leq V \; \forall \; i \in [1, 2, ..., n] \; and \; \exists \; j \in [1, 2, ..., n] : U_j < V_j \tag{2.5}$$

Ou seja, uma solução domina outra, quando esta possui valores melhores para todos objetivos.

Com essa definição, é possível dividir o exemplo anterior em dois grupos, de soluções dominadas e soluções não dominadas. A Figura 2.9 apresenta essa divisão, em que as soluções não dominadas estão separadas por um retângulo.

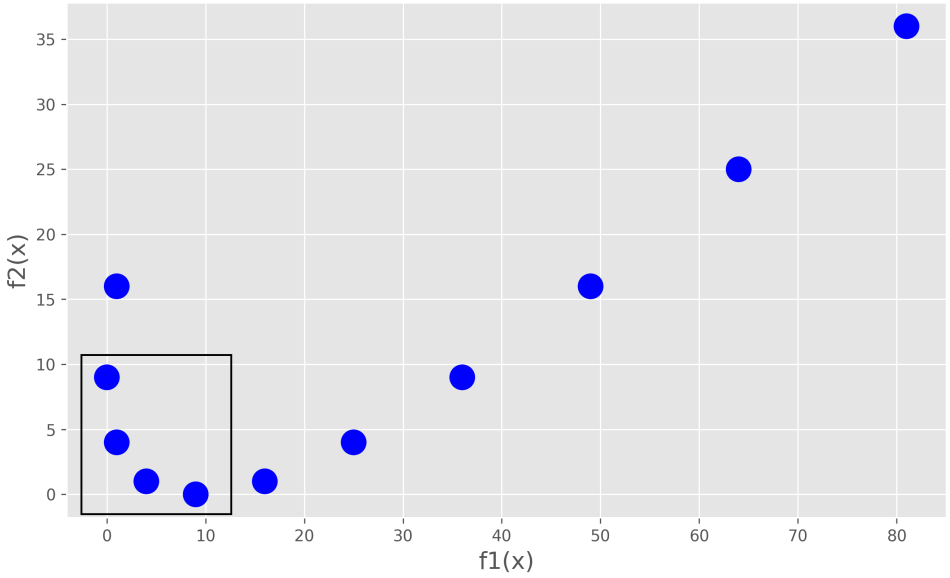

Figura 2.9 – Relação de Dominância

Analisando o problema de compressão na literatura, é possível relatar, um conflito existente entre a quantidade que uma rede neural é podada e o valor da acurácia. Como pode ser visto no trabalho de Ferreira (2022).

2.4 Otimização baseada em substitutos

Como definido na seção anterior, problemas de otimização multiobjetivo consideram a avaliação de diferentes tipos de funções. O problema de compressão de redes seguido da validação do modelo, pode ser facilmente modelado em funções objetivo. Porém, avaliações de modelos desta natureza, tende a ser extremamente custosa em um nível computacional. Sendo assim, uma alternativa viável é utilizar a otimização baseada em substitutos, que são métodos que visam substituir o problema original por um modelo substituto semelhante, porém, com um custo computacional bem inferior (SILVA, 2018).

O fluxograma da Figura 2.10, apresentado por SILVA (2018), representa o processo de otimização baseada em substitutos. O primeiro passo, é construir um modelo alternativo com

base no original. Se esse modelo for considerado satisfatório, ele será repassado ao algoritmo otimizador. Logo após a otimização, será realizado a validação de acordo com a proposta original. Caso as soluções atendam aos requisitos, o método é finalizado. Caso contrário, o procedimento é reiniciado, passando as informações já obtidas para a construção de um modelo substituto aprimorado. O processo continua até que os recursos de tempo e processamento sejam esgotados.

De acordo com SILVA (2018), existem algumas formas de construir um problema substituto, são elas:

- **Reduzindo o número de variáveis de decisão:** Nesta abordagem a premissa é que existem variáveis redundantes e a ideia é resolver um problema em que as variáveis de decisão sejam um subconjunto do problema original;
- **Reduzir o número de objetivos:** Eliminar objetivos que sejam redundantes ou irrelevantes, de forma semelhante ao que ocorre na redução de variáveis;
- **Modelo Substituto:** qualquer construção que seja eficiente em produzir respostas análogas ao modelo original.

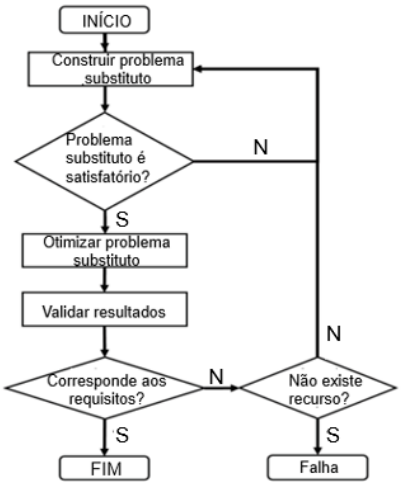

Figura 2.10 – Otimização baseada em substitutos (adaptado SILVA (2018))

Nas próximas Seções, será abordada a formulação do problema multiobjetivo de compressão, que posteriormente será otimizado através de modelos substitutos.

3 Compressão de redes neurais utilizando PyTorch

3.1 PyTorch

O Pytorch é uma oferta de código aberto do facebook que facilita a construção de código de aprendizado profundo na linguagem de programação python (POINTER, 2019). Lançada em 2016, o framework PyTorch permite suporte para acelaração em GPU, integração com a biblioteca numpy e gerações automáticas de gráficos computacionais (PYTORCH, 2018). Ainda, o Pytorch permite uma alta flexibilidade na utilização dos tensores e na criação de diferentes tipos de estruturas de redes neurais. Tal flexibilidade, é de grande importância para a comunidade científica, para fins de testes e pesquisas.

3.2 Módulo de compressão

Neste trabalho estudamos 4 objetivos relacionados ao problema de compressão, sendo a minimização do tempo de inferência, minimização do tamanho do modelo, maximização da acurácia e minimização da taxa de parâmetros não nulos, como apresentado pelas equações abaixo.

$$minimizar f_1(\mathbf{x}) : Tempo(Inferência) \tag{3.1}$$

$$minimizar f_2(\mathbf{x}) : Tamanho(MB) \tag{3.2}$$

$$maximizar f_3(\mathbf{x}) : acurácia \tag{3.3}$$

$$minimizarr f_4(\mathbf{x}) : taxa\ não\ nulos \tag{3.4}$$

O quarto objetivo está relacionado somente com a compressão por poda. Tal objetivo foi adicionado pois está relacionado à esparsidade dos tensores de pesos e bias. Existem estruturas de dados e algoritmos capazes de representar matrizes esparsas armazenando somente os números não nulos, gerando uma economia de armazenamento (RODRIGUES, 2011). Logo, mesmo o pytorch não realizando esses procedimentos para economia de memória, diminuir a taxa de não nulos é algo interessante.

O PyTorch disponibiliza funções para compressão das redes neurais através de duas técnicas distintas, poda e quantização. Na quantização, é possível reduzir os parâmetros da rede

para inteiros 8 bits em camadas convolucionais e lineares. Em relação a poda, a biblioteca possui maior flexibilidade, possibilitando a escolha da camada, dos parâmetros a serem podados e da intensidade da poda. As próximas seções irão detalhar sobre a implementação destas técnicas neste trabalho.

3.2.1 PyTorch Poda

Implementada no módulo `torch.nn.utils.prune`, a poda no PyTorch permite anular valores de pesos ou bias em uma determinada camada da rede. Existem 3 formas para escolha dos parâmetros a serem podados. A primeira escolhendo valores totalmente aleatórios, a segunda baseada na norma L_1 e a terceira com base na norma L_2. A norma L_1, realiza um somatório sobre o valor absoluto de todos pesos ou bias de uma camada. Logo após, o valor de cada parâmetro é dividido pelo valor da norma, os parâmetros que resultam nos menores valores da divisão, são anulados pela poda. Semelhante à norma L_1, a L_2 realiza um procedimento parecido, no entanto, os valores no somatório são elevados ao quadrado e é calculado a raiz quadrada da soma final. Após isso, é realizado o mesmo procedimento de divisão e anulação dos parâmetros de pesos ou bias. Neste trabalho, foi adotado a estratégia de escolha com base em L_1.

```
1         def evalCompression(x, Net):
2
3             for name, module in Net.named_modules():
4                 if isinstance(module, torch.nn.Conv2d):
5                     prune.l1_unstructured(module, name='weight', amount=x)
6                     prune.remove(module, 'weight')
7
8                  elif isinstance(module, torch.nn.Linear):
9                     prune.l1_unstructured(module, name='weight', amount=x)
10                    prune.remove(module, 'weight')
```

O código acima demonstra como foi realizada a poda em relação aos valores dos pesos. A função `evalCompression` recebe como parâmetro o valor da intensidade da poda e a rede a ser comprimida. As redes neurais selecionadas neste trabalho, permitem comprimir dois módulos distintos, as camadas convolucionais e lineares. A estrutura de repetição (linha 3) passa por cada módulo da rede neural. Nas linhas 4 e 8, existem condicionais que verificam o tipo do módulo, permitindo aplicações diferentes de poda para as camadas lineares e convolucionais. Na linha 5 é realizado a poda em um módulo convolucional da rede neural. A quantidade de conexões removidas é representado pela variável x, que expressa a intensidade da poda. Por último, na linha 6 é removido a reparametrização tornando a poda permanente (PYTORCH, 2022). O processo descrito nas linhas 5 e 6, se repetem nas linhas 9 e 10, porém, para um módulo linear.

O código acima, refere-se ao processo de poda para os pesos de uma rede neural. Para podar o valor de bias, basta substituir o name 'weight' das linhas 5 e 9 pelo name 'bias'.

Como apresentado na revisão de literatura presente na Seção 1, inúmeros autores utilizam a poda seguido por retreino. No entanto, uma limitação devido a utilização do Pytorch, uma vez que a biblioteca não realiza o congelamento dos parâmetros já previamente podados. Com isso, esses campos podem assumir outros valores diferentes de nulo em caso de retreino, podendo desfazer o processo de poda. Nessa perspectiva, neste trabalho só será abordado estratégias de compressão em redes previamente treinadas, prontas para o uso.

3.2.2 PyTorch Quantização

O Pytorch oferece algumas abordagens para a quantização de um modelo, permitindo a diminuição do custo de armazenamento. Essas abordagens estão implementadas no módulo torch.quantization e podem ser do tipo dinâmica ou estática. A quantização estática reduz todos os parâmetros das camadas lineares e convolucionais da rede para inteiro 8 bits, e a quantização dinâmica reduz apenas os parâmetros das camadas lineares.

Atualmente, a quantização está implementada em modo Beta, o que implica em algumas limitações em relação ao seu uso. Como limitação principal, é possível destacar que modelos quantizados só podem ser alocados em CPU, não sendo possível o casting em GPU.

Redes neurais PyTorch, apresentam por padrão, os valores dos pesos e bias como sendo do tipo ponto flutuante 32 bits. Por sua vez, a quantização dinâmica atua sobre as camadas lineares reduzindo a representação do valor de pesos e bias para inteiros 8 bits. Segundo algumas diretrizes desse tipo de quantização, o modelo quantizado dinamicamente não pode ser colocado em modo de treino, apenas de validação (QUANTIZATION, 2022). Dessa forma, a quantização será utilizada logo após o processo de poda, em redes previamente treinadas.

O código abaixo, apresenta a implementação da quantização.

```
1     model_int8 = torch.quantization.quantize_dynamic(
2                     Net,  # the original model
3                     {torch.nn.Linear},  # a set of layers to dynamically quantize
4                     dtype=torch.qint8)  # the target dtype for quantized weights
```

Para a quantização dinâmica são necessários 3 parâmetros. O primeiro se trata da rede a ser comprimida, que no exemplo está representado por (`Net`). Por sua vez, o segundo parâmetro é o modulo da rede a ser quantizado, que neste caso seria o módulo linear. Por último, existe o parâmetro de quantização, que segundo a documentação, só pode ser do tipo `torch.qint8`. A quantização dinâmica gera um novo modelo a partir da rede neural inicial.

3.2.3 Variáveis de projeto

Como descrito na Seção 3.2.1, a compressão por poda necessita de uma configuração especifica, que pode variar de acordo com a estratégia adotada. Tipo da poda, camada a ser comprimida e intensidade, são os parâmetros a serem definidos que podem alterar o valor dos objetivos. Além disso, é possível combinar as duas técnicas de compressão. Nesse contexto, foram definidos variáveis de projeto com o intuito de gerar diversas combinações para análise de seus efeitos perante as funções objetivo. A Tabela 3.1 apresenta um resumo das variáveis de projeto.

Tabela 3.1 – Variáveis de Projeto

Variável	**Definição**	**Domínio**
x_1	Poda para a camada linear	$x_1 \in \{0, 1, 2\}$
x_2	Poda para a camada convolucional	$x_2 \in \{0, 1, 2\}$
x_3	Intensidade da poda linear	$0 < x_3 < 1$
x_4	Intensidade da poda convolucional	$0 < x_4 < 1$
x_5	Tipo de poda linear	$x_5 \in \{0, 1, 2\}$
x_6	Tipo de poda convolucional	$x_6 \in \{0, 1, 2\}$
x_7	Quantização	$x_7 \in \{0, 1\}$

Cada variável apresentada na Tabela 3.1 possui uma restrição de domínio, em que cada valor representa uma estratégia para o modelo de compressão. As variáveis de um a seis referem-se à compressão por poda, sendo x_7 a única variável que representa a quantização, em que o valor 0 indica ausência de quantização e o valor 1 com a aplicação do processo de quantização dinâmica.

Na Tabela presente na Figura 3.1 é apresentado um resumo com a descrição das possibilidades para as variáveis x_1, x_2, x_5 e x_6.

variável/ possibilidade	0	1	2
x_1	Não realiza a poda na camada linear	Escolhe camadas lineares de forma aleatória para aplicação da poda	Aplica a poda em todas camadas lineares
x_2	Não realiza a poda na camada convolucional	Escolhe camadas convolucionais de forma aleatória para a aplicação da poda	Aplica a poda em todas camadas convolucionais
x_5	Poda somente o valor de bias na camada linear	Poda somente o valor dos pesos na camada linear	Poda o valor de bias e pesos na camada linear
x_6	Poda somente o valor de bias na camada convolucional	Poda somente o valor dos pesos na camada convolucional	Poda os valores dos pesos e bias na camada convolucional

Figura 3.1 – Variáveis x_1, x_2, x_5 e x_6

Por fim, as variáveis x_3 e x_4 deve assumir um valor contínuo entre 0 e 1. Esse valor indica a intensidade da poda, sendo x_3 para camadas lineares e x_4 para camadas convolucionais.

3.2.4 Problema de Compressão

Com as variáveis de projeto definidas na seção 3.2.3 é perceptível que o problema a ser tratado é combinatório e multiobjetivo. Desta forma, foi utilizado o `pymoo` (Blank; Deb, 2020) para definir o problema a ser utilizado por um algoritmo multiobjetivo, como demonstrado abaixo:

```
class MyProblem(ElementwiseProblem):

def __init__(self, limite):
    super().__init__(n_var=7,
                     n_obj=2,
                     n_constr=1,
                     xl=np.array([0, 0, 0, 0, 0, 2 , 0 ]), #limite inferiror
                     xu=np.array([2, 2, 1, 1, 2, 2, 1])) # limite superior

def _evaluate(self, x, out, op = 'vgg16', rest, **kwargs):
    # x representa uma única solução a ser avaliada
    # out é um dicionario que representa a saída
    # rest é acurácia miníma
    # op é a rede a ser comprimida

    objs = compression.comprime(x, op)
    f1 = -1*(objes[0]) # acurácia negativada
    f2 = objs[1]

    g1 = rest + f1  # restrição


    out["F"] = [f1, f2] # saida para os objetivos
    out["G"] = [g1] # saida para as restrições
```

Com a formulação do problema, é possível definir um algoritmo para geração de modelos substitutos com base no problema original. Tal algoritmo será apresentado na próxima Seção.

4 Seleção de modelos substitutos para o problema de compressão

4.1 Otimização multiobjetivo restrita baseada em modelos substitutos

Como descrito na Seção 2.4, aplicar algoritmos de otimização para problemas de compressão, pode ser inviável, devido ao custo de tempo requerido para avaliar as funções objetivos. Nessa perspectiva, uma solução encontrada na literatura, é otimizar em modelos substitutos e avaliar as soluções encontradas no problema original. O algoritmo proposto nesse trabalho foi adaptado de Silva et al. (2017) e segue os seguintes passos descritos no fluxograma da figura 4.1.

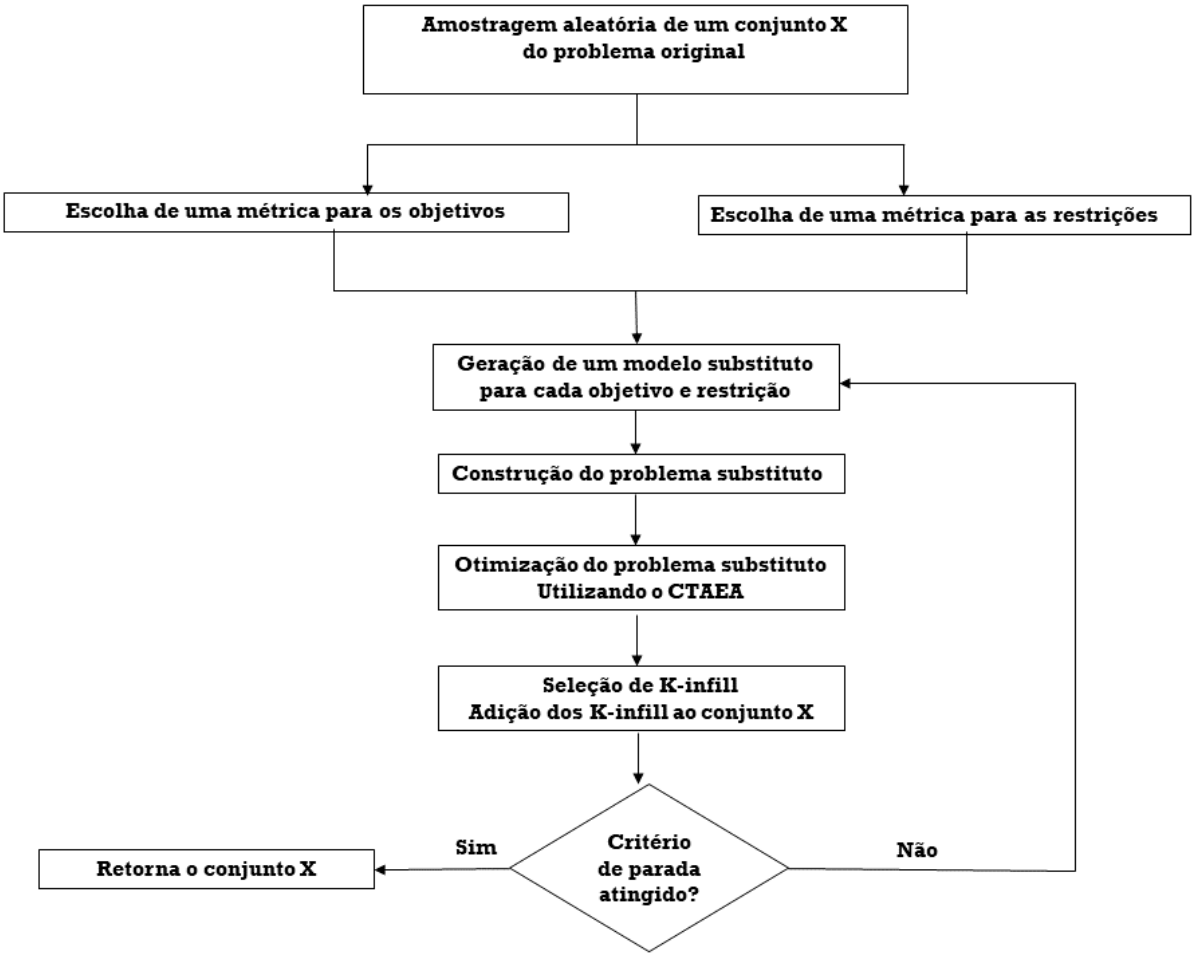

Figura 4.1 – Otimização por substitutos

Primeiramente, é realizado a amostragem de um conjunto aleatório do problema original.

Logo após, métricas serão selecionadas para a avaliação de modelos de regressão sobre o conjunto de dados $\mathbf{X}$, com a finalidade de escolher modelos para cada função objetivo e restrições. Com a escolha dos modelos, é possível construir um problema substituto similar ao original, que será resolvido por um algoritmo de otimização multiobjetivo restrito. Cada processo de otimização em n épocas, retorna um conjunto de soluções factíveis e não dominadas. Dado esse conjunto, são selecionados k infill points, que são soluções a serem avaliadas no problema original e adicionadas ao conjunto $\mathbf{X}$.

O processo de gerar modelos substitutos, escolher e otimizar, é repetido até que o número de amostras contidas no conjunto $\mathbf{X}$ seja satisfeito. Na Seção 4.2, será detalhado a respeito das métricas utilizadas para a seleção de modelos para os objetivos e restrições.

4.2 Métricas para seleção de modelos

De forma geral, os modelos substitutos para objetivos e restrições, são modelos de regressão implementados pela biblioteca Scikit-learn (PEDREGOSA et al., 2011). Após gerar modelos de regressão, para escolha dos modelos a serem utilizados, foram selecionados quatro métricas distintas, que serão descritas nas próximas subseções.

4.2.1 Erro Quadrático Médio (MSE)

O erro quadrático médio (MSE), é uma métrica de risco, que calcula o quadrado da diferença entre o valor real e o valor predito. Pode ser definido pela equação 4.1

$$\frac{1}{n}(\sum_{i=0}^{n-1}(Y_i - Y_{pi})^2) \tag{4.1}$$

Onde n representa a quantidade de amostras, y_i o valor real de cada amostra e Y_{pi} o valor da predição.

4.2.2 Erro Percentual Absoluto Médio (MAPE)

O erro percentual médio absoluto, é uma métrica de avaliação para problemas de regressão. É uma medida independente de escala, que pode ser útil para comparar modelos com diferentes conjuntos de dados Silva et al. (2017). Essa métrica é definida pela equação 4.2.

$$\frac{1}{n}(\sum_{i=0}^{n-1}\frac{(|Y_i - Y_{pi})|}{MAX(e, |y_i|)}) \tag{4.2}$$

Onde n representa a quantidade de amostras, y_i o valor real da amostra, Y_{pi} é o valor da predição e e um valor positivo muito pequeno, evitando inconsistências na divisão quando y_i for zero.

4.2.3 Rand

Rand é uma seleção aleatória. Dado uma lista de diferentes modelos, o rand escolhe um modelo totalmente aleatório para os objetivos e outro totalmente aleatório para as restrições.

4.2.4 Correlação de Spearman

O coeficiente de correlação Spearman, é uma medida não paramétrica que descreve a relação entre conjuntos de dados através de funções monotônicas (WEIR, 2017). Diferente das métricas apresentadas nas seções anteriores, a correlação de Spearman exige cálculos mais complexos, com análises de ranqueamento. Em Silva et al. (2017), a correlação de Spearman e definido através das seguintes equações:

Seja $f_{m,i}(.)$ o valor da função objetivo definido sobre o modelo $m_,$ de tal modo que:

$$f_{m,1} < f_{m,2} < f_{m,3} < f_{m,4} < f_{m,n} \tag{4.3}$$

O rank r_m de uma solução candidata x_i de um dado modelo m_s é

$$r_m(x_i) = j \; de \; tal \; modo \; que \; f_{m,j} = f_m(x_i) \tag{4.4}$$

Após a computação dos rank, é possível definir a correlação de Spearman através da equação 4.5.

$$r_{rs,rf} = \frac{cov(r_s,r_f)}{\sigma_j \sigma_{rf}} = \frac{\sum_{i=0}^{n-1}(r_{s,i} - \tilde{r_s})(r_{f,i} - \tilde{r_f})}{\sqrt{\sum_{i=0}^{n-1}(r_{s,i} - \tilde{r_s})^2}\sqrt{\sum_{i=0}^{n-1}(r_{f,i} - \tilde{r_f})^2}} \tag{4.5}$$

Onde Cov(r_s,r_f) é a covariância das variáveis de rank. σ_{rs} e σ_{rf} representam o desvio padrão das variáveis de classificação.

Definindo esses cálculos de correlação, é possível realizar um comparativo entre os resultados dos modelos decidindo qual modelo mais adequado para a construção do problema.

4.3 Otimizador CTAEA

Problemas multiobjetivos restritos, requerem um balanceamento entre a convergência, diversidade e viabilidade das soluções com base nas restrições adotadas (LI et al., 2019) . Nessa perspectiva, o algoritmo adotado para otimizar os modelos substitutos foi o CTAEA, que é o algoritmo atual do estado da arte para resolver problemas multiobjetivos restrito. Tal algoritmo, foi proposto em Li et al. (2019) e é baseado nos seguintes arquivos:

- **Convergence Archive (CA):** Arquivo orientado a convergência, responsável por direcionar a população para a região factível e aproximar da fronteira de pareto;

- **Diversity Archive (DA):** Arquivo complemento, que visa manter a diversidade das soluções através da exploração de áreas subexploradas, incluindo regiões não factíveis.

Após gerar os dois arquivos, o algoritmo realiza um procedimento denominado seleção de acasalamento restrito, no qual os arquivos são combinados para a seleção dos pais da próxima geração. O primeiro progenitor, é selecionado do arquivo com maior proporção de soluções não dominadas, já o segundo progenitor é selecionado de um arquivo aleatório. O pseudo código contido na Figura 4.2 exemplifica esse processo.

```
Input: CA, DA
Output: Mating parents p1, p2
H_m ← CA ⋃ DA;
ρ_c ←proportion of nondominated solution of CA in H_m;
ρ_d ←proportion of nondominated solution of DA in H_m;
if ρ_c > ρ_d then
    p1 ←TournamentSelection(CA);
else
    p1 ←TournamentSelection(DA);
if rand < ρ_c then
    p2 ←TournamentSelection(CA);
else
    p2 ←TournamentSelection(DA);
return p1, p2
```

Figura 4.2 – Restricted Mating Selection (LI et al., 2019)

Após escolher os progenitores, é realizado um cruzamento binário e uma mutação polinomial. No final da otimização, será retornado as soluções factíveis e não dominadas. Assim como o NSGAIII (DEB; JAIN, 2014), o CTAEA (LI et al., 2019), é um algoritmo evolutivo baseado em direções de referências. Dessa forma, é necessário definir os seguintes parâmetros:

1. **reference direction:** Direção de referência que será utilizada no processo de otimização. Uma matriz, onde cada linha representa uma referência e cada coluna uma variável;
2. **Sampling:** Conjunto inicial de amostras do problema a ser otimizado. Podendo ser aleatório ou um conjunto de soluções já avaliadas;
3. **Selection:** Forma que os progenitores são escolhidos para produzir os descendentes;
4. **Crossover:** Forma relacionada a geração dos descendentes a partir dos progenitores;
5. **Mutation:** Probabilidade de mutação dos indivíduos gerados no cruzamento.

O CTAEA pode ser utilizado facilmente através da API fornecido pelo pymoo (Blank; Deb, 2020).

5 Metodologia experimental

Como descrito na Seção 1.1, este trabalho visa parametrizar os algoritmos de compressão verificando se as variáveis escolhidas de fato têm efeito nos objetivos e aplicar uma otimização baseada em substitutos. Sendo assim, nesta seção será descrito todos equipamentos e estruturas utilizadas para a realização dos experimentos.

5.1 Hardware

A implementação do projeto foi realizada em duas fases distintas, ambas utilizando o Google Colab. A primeira implementação tem por objetivo pegar uma rede base e treiná-la usando uma base de dados. Como o processo de treinamento é demorado e custoso, foi utilizado um acelerador de hardware disponibilizado pelo *Google colab*. A segunda implementação consiste na aplicação das técnicas de compressão sobre a rede treinada na fase 1. Como a quantização é limitada a respeito da utilização de aceleradores de hardware, na fase 2 foi utilizado apenas o CPU do *Google colab*. A Tabela 5.1 apresenta a configuração do hardware utilizado.

Tabela 5.1 – Configuração Hardware

Hardware	Configuração
GPU	NVIDIA Tesla T4
CPU	Intel(R) Xeon(R) 2.200 GHz
RAM do sistema	12.68 GB
Memória da GPU	15109 MiB
Disco	107.72 GB
Frequência CPU	2199.998 MHz

5.2 Base de Dados

A base de dados escolhida foi a CIFAR-10, devido à sua ampla utilização na literatura, facilitando a comparação de resultados com outros trabalhos. A CIFAR-10 consiste em uma base de dados com 60 mil imagens coloridas de tamanho 32x32, que são divididas em 10 classes distintas, com cada uma contendo exatamente 6000 imagens (CSTORONTO, s.d). A nível de implementação, é possível fazer o download das imagens da seguinte forma:

```
1          import torch
2          from torchvision import datasets
3          from torchvision import transforms
4          dados_transformados = transforms.Compose([
```

```
5                               transforms.Resize(32),
6                               transforms.ToTensor(),])
7
8           dados_treino = datasets.CIFAR10('.',
9                                           train=True,
10                                          transform=dados_transformados,
11                                          download=True)
12
13          dados_teste = datasets.CIFAR10('.',
14                                         train=False,
15                                         transform=dados_transformados,
16                                         download=True)
```

As redes neurais esperam os dados de entrada em formato de tensor, para isso é utilizado a função transforms com a finalidade de transformar os dados de entrada no formato desejado. A própria base de dados CIFAR-10 permite a separação dos dados de treino e teste, sendo 50 mil imagens para treino e 10 mil para testes (CSTORONTO, s.d). Nas linhas de 8 a 11 é realizado o download e conversão das imagens para treino e da linha 13 a 16 o armazenamento das imagens próprias para validação.

A Figura 5.1 apresenta 10 exemplos de imagens de cada uma das classes presentes na base de dados escolhida.

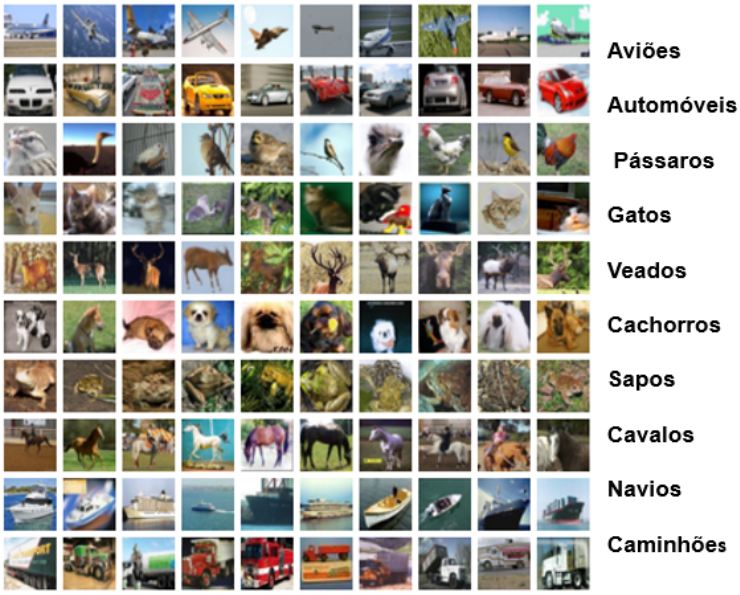

Figura 5.1 – Classes CIFAR-10(adaptado(<https://www.cs.toronto.edu/~kriz/cifar.html>))

5.2.1 Arquiteturas selecionadas

De mesma forma que a base de dados, as arquiteturas foram escolhidas de acordo com sua utilização na literatura, o que facilita a comparação entre diversos trabalhos. As arquiteturas selecionadas foram a VGG16 (SIMONYAN; ZISSERMAN, 2014) e a Resnet50 (HE et al., 2016), que serão exploradas nesta seção.

5.2.2 VGG16

A VGG16 é um modelo de rede neural convolucional profundo, que possui 13 camadas convolucionais e 3 camadas totalmente conectadas com mais de 138 milhões de parâmetros (SUGATA; YANG, 2017). A Figura 5.2 exemplifica a disposição de camadas de uma rede VGG16 com base em um tensor de entrada de dimensões 224x224.

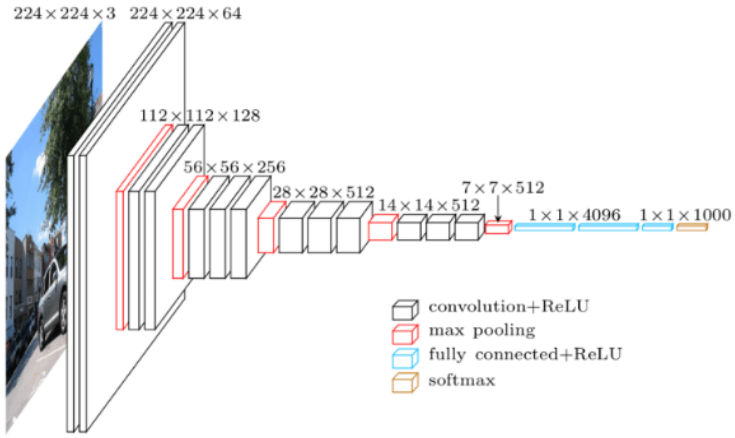

Figura 5.2 – VGG16 ((SUGATA; YANG, 2017))

Como demonstrado na Figura 5.2, a VGG16 utiliza a função de ativação ReLu que basicamente processa o valor de entrada da função retornando 0 para valores negativos e o próprio valor para entradas positivas. Além disso, no processo de convolução é utilizado máscaras convolucionais de tamanho 3x3, que são arrastadas em janelas com espaçamento 1x1. Se for necessário o preenchimento das bordas, o valor definido de *padding* é 1x1. Na saída final da VGG16 é realizado uma distribuição de probabilidades definida pela função *softmax*.

5.2.3 Resnet50

Assim como a VGG16, a resnet50 é uma rede neural convolucional profunda que utiliza a função de ativação ReLu. No entanto, existem grandes divergências entre as arquiteturas. A Resnet50 possui 50 camadas, sendo 49 convolucionais e apenas uma totalmente conectada. Os filtros convolucinais seguem o tamanho padrão de 3x3 (BENDJILLALI et al., 2020) e após o processo de convolução é realizado uma normalização de dados.

Apesar de possuir uma quantidade maior de camadas que a arquitetura VGG16, segundo Bendjillali et al. (2020) a Resnet50 possui uma quantidade inferior de parâmetros o que a torna menos complexa. É estimado um valor inferior a 26 milhões de parâmetros para essa arquitetura.

5.3 Treinamento

A Tabela 5.2 apresenta as configurações utilizadas para o treinamento das redes neurais adotadas neste trabalho.

Tabela 5.2 – configurações para o treinamento

Hiperparâmetros	Configuração
Épocas	40
Taxa de aprendizado	1e-3
Penalidade de pesos (Regularização L2)	1e-3
tamanho do Batch	50
Base de dados	CIFAR10
Função de perda	Cross Entropy Loss (Entropia cruzada)
Otimizador	Adam

Na próxima seção serão apresentados os resultados do experimentos definidos nesta seção.

6 Resultados

A demonstração dos resultados, será feito em duas seções distintas. Primeiramente, em 6.1 será analisado de uma forma geral, a relação entre as variáveis de projeto definido na Seção 3.2.3 com os objetivos propostos em 3.2. Em um segundo momento, na seção 6.2, será demonstrado os resultados com base na otimização por modelos substitutos.

6.1 Efeito das variáveis definidas na qualidade dos modelos comprimidos

Para analisar os efeitos das escolhas das variáveis de projeto em relação ao objetivos, foram selecionados um conjunto de valores para a realização dos testes. A Tabela 6.1 representa essa seleção.

Tabela 6.1 – Valores variáveis de projeto

$x_1 \in \{0, 1, 2\}$
$x_2 \in \{0,1,2\}$
$x_3 \in \{0.1, 0.25, 0.5, 0.75, 0.95\}$, $x_4 = x_3$
$x_5 = x_6 = 2$
$x_7 \in \{0, 1\}$

Foram consideradas todas as combinações dos níveis descritos na Tabela 6.1.

Para uma análise adequada dos resultados preliminares, serão demonstrados alguns gráficos comparando os objetivos dois a dois para as duas redes convolucionais. Os rótulos contidos nos gráficos, denotam os seguintes significados:

N: Rede original sem compressão;

Q: Compressão apenas por quantização;

PQ: Compressão por poda e quantização;

P: Compressão apenas por poda.

O gráfico da Figura 6.1, demonstra a comparação entre a acurácia e a taxa de não nulos. Analisando a parte inferior esquerda dos gráficos, é possível observar, que podas agressivas que resultam em valores baixos para a taxa de não nulos, pode gerar soluções com acurácias baixas. No entanto, em alguns casos temos soluções com valores pequenos para f4 com acurácia similar ao modelo original. Por sua vez, a quantização não gera grandes efeitos para a acurácia, visto que

a solução comprimida apenas pela quantização está sobreposta ao problema original. Além disso, é possível observar, que para cada solução comprimida apenas por poda, existe uma solução próxima comprimida por poda e quantização, ressaltando que os maiores efeitos em relação a acurácia é ocasionada pela poda. A análise é pertinente para as duas arquiteturas selecionadas.

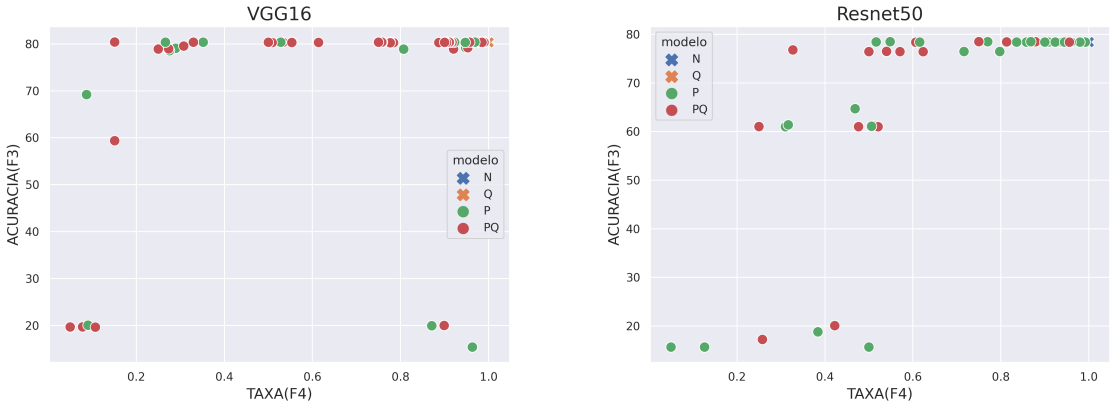

Figura 6.1 – Acurácia X Taxa

A partir dos gráficos da Figura 6.2, é possível perceber a divisão das soluções em dois grupos distintos referente ao tamanho. Um grupo possuí soluções quantizadas e o outro grupo, soluções sem quantização. Dessa forma, é possível concluir, que dos métodos de compressão adotados nesse trabalho, somente a quantização diminui o valor do tamanho em megabytes (MB). A poda, não interfere no objetivo F2 e existem apenas dois valores possíveis para este objetivo.

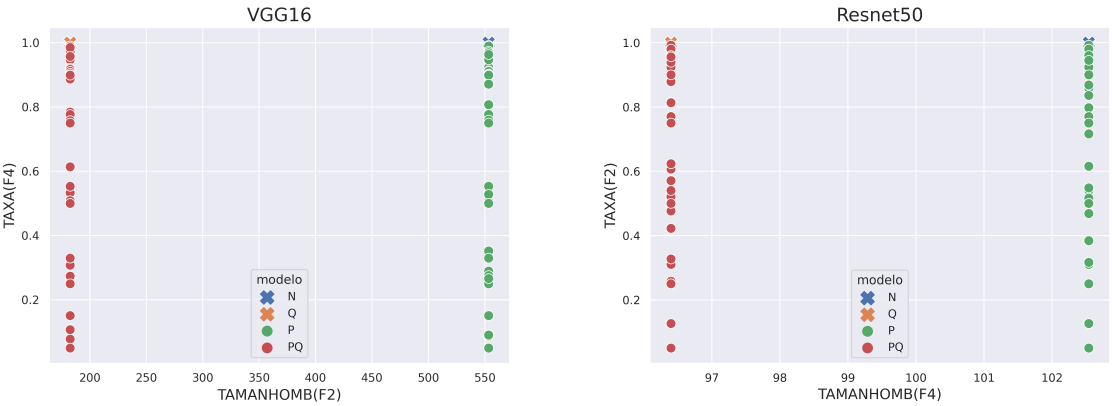

Figura 6.2 – Taxa X TamanhoMB

O tempo, foi o objetivo com maior variância de acordo com os testes realizados e demonstrados na Figura 6.3. A poda tende a diminuir o tempo de inferência para as duas redes. No entanto, a quantização tem efeitos distintos para as arquiteturas selecionadas. Para a VGG16, a quantização diminui consideravelmente o tempo. Por outro lado, a quantização na Resnet50 ocasiona o aumento de alguns segundos se comparado com soluções que são comprimidas apenas por poda. Existem fatores externos que podem influenciar no valor do tempo, como lentidão no

servidor, consumo excessivo de recursos computacionais por outros processos etc. No entanto, analisando os resultados para as duas arquiteturas, é possível observar a tendência de tempos menores para valores menores da taxa de não nulos, uma relação diretamente proporcional.

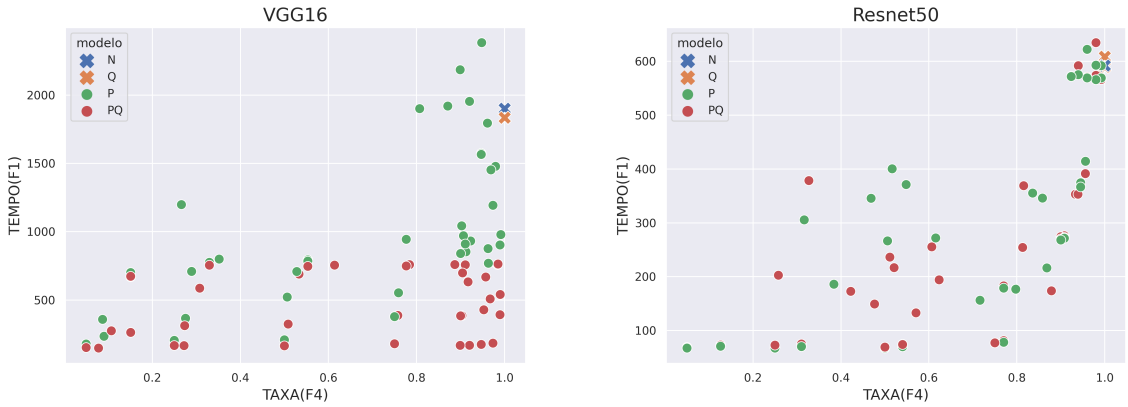

Figura 6.3 – Tempo(s) X Taxa

A comparação, demonstra que existem objetivos conflitantes, o que caracteriza um problema mutlti-objetivo. Como exemplo de conflito, é possível destacar que a solução com o menor tempo de inferência, possui uma acurácia muito ruim, em ambas as redes neurais selecionadas. O mesmo se aplica na relação acurácia x taxa de não nulos, os resultados com as menores taxas possuem baixa acurácia. No entanto, existem soluções promissoras com acurácia similar e um tempo de inferência muito menor em relação a rede original. Tais soluções, estão destacadas nos gráficos da Figura 6.4.

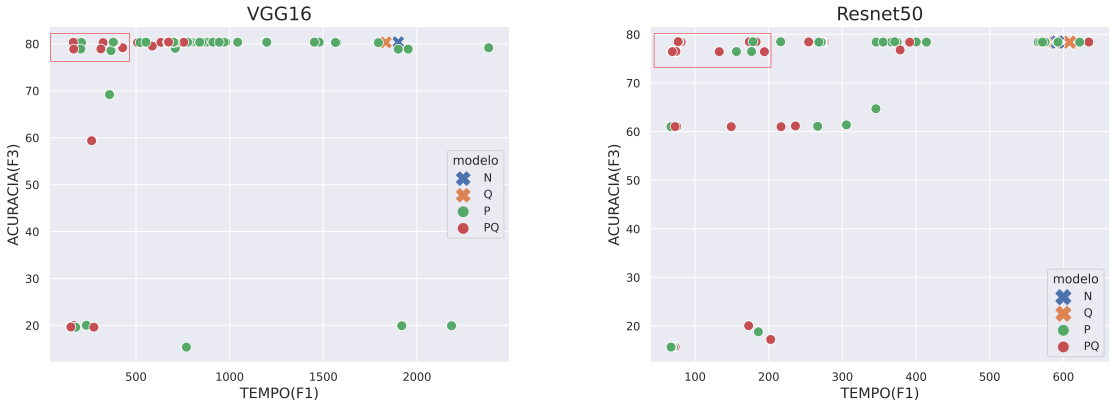

Figura 6.4 – Tempo X Acurácia

6.2 Otimização em Modelos substitutos

Com os resultados demonstrados na seção anterior, fica explicito a necessidade da utilização de algoritmos otimizadores para escolha dos parâmetros de compressão. No entanto, otimizar

sobre o modelo original, pode ser inviável devido ao alto tempo de inferência na avaliação de cada solução.

Com a finalidade de diminuir a complexidade do problema, foi realizada uma redução nos números de objetivos. Nessa fase, estão sendo considerados apenas dois objetivos, a acurácia e a taxa de não nulos. A redução foi pertinente, devido às análise realizada em 6.1 no qual foi constatado, que existem apenas dois valores para o TamanhoMB, que é influenciado apenas pela quantização, que quando aplicada, gera efeitos pequenos para a acurácia e tempo. Por sua vez, a maior relação para a diminuição do tempo de inferência, é proveniente da poda, que é representada pela taxa de não nulos, logo, minimizando essa taxa é equivalente a minimizar o tempo de inferência.

Dessa forma, esta seção será dedicada a análise dos resultados obtidos na aplicação da otimização em modelos substitutos, descrito na Seção 4.1. O algoritmo CTAEA (LI et al., 2019) foi utilizado com uma população inicial de 50 indivíduos e 1000 gerações. Além disso, foram selecionados 2 infills-points de cada processo de otimização. Como critério de parada, foi adotado o tamanho do conjunto X como sendo 100 avaliações do problema original.

As quatro métricas adotadas neste trabalho, descrito em 4.2, foram combinadas, gerando um total de 16 configurações distintas de escolha de modelos para construção do problema substituto. Cada configuração foi testada 5 vezes, para uma maior análise sobre o comportamento de escolha das métricas. A Tabela presente na Figura 6.5 representa cada uma das combinações possíveis.

Objetivos	Restrições			
MSE	MSE	MAPE	SPEARMAN	RAND
MAPE	MSE	MAPE	SPEARMAN	RAND
SPEARMAN	MSE	MAPE	SPEARMAN	RAND
RAND	MSE	MAPE	SPEARMAN	RAND

Figura 6.5 – Combinações de métricas

Visando um comparativo sobre a eficácia da otimização por substitutos, os seguintes experimentos da Tabela 6.2 foram realizados levando em consideração apenas o problema original. Cada experimento leva em consideração 100 avaliações, mesma proporção adotada nos testes com modelos substituto, em que o conjunto final possui tamanho 100.

Tabela 6.2 – Testes Problema Original

Teste	**População**	**Gerações**
CTAEAT1	5	20
CTAEAT2	10	10
Substitutos	50	1000

Cada um desses experimentos foram realizados para as duas redes neurais convolucionais. Os resultados serão demonstrados nas próximas seções. Para comparar os conjuntos de soluções dos diversos testes realizados, foram utilizadas medidas de hipervolume, que são expressos pelos diagramas de caixas nas Figuras 6.6 e 6.7.

Analisando a figura 6.6 é perceptível que todas as combinações de métricas para os substitutos obtiveram vantagens em relação a utilização do método CTAEA puro. Os melhores resultados, como podem ser vistos no gráfico, foram provenientes da escolha da métrica mape para restrições e objetivos. Como demonstrado na Seção 4.2.2, na equação do mape as variáveis do problema são dimensionadas para unidades percentuais em uma aplicação de somatório. Desta forma, com a realização dos testes, foi possível constatar que essa média absoluta (MAPE) se adaptou a muito bem tanto para os objetivo quanto para a restrição.

Realizando uma análise sobre a variância, é possível perceber que a métrica spearmean para objetivos e restrições resultou em valores bem próximos para cada um dos testes, tendo resultados consistentes.

Outra análise interessante, é sobre a primeira configuração do CTAEA puro, como pode ser observado nos gráfico, todos os 5 testes resultou em valores de hipervolume muito abaixo dos demais testes. tal configuração, não conseguiu convergir para um conjunto de Pareto eficiente.

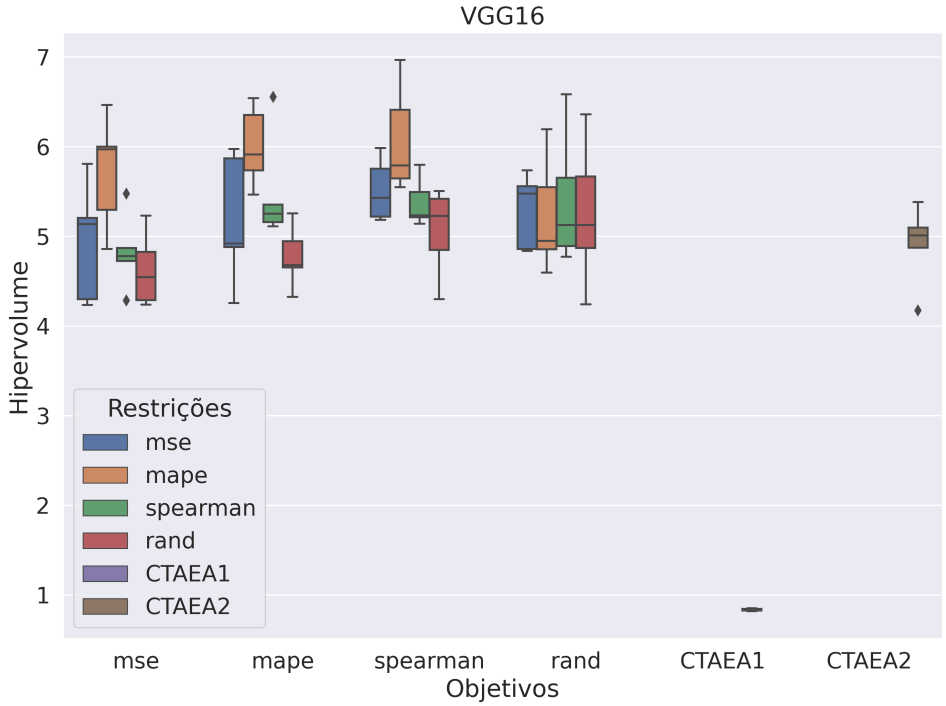

Figura 6.6 – Diagrama de caixas hiper-volumes VGG16

A partir da Figura 6.7, é perceptível que os padrões analisados para a a rede neural VGG16 são semelhantes para a resnet50. A escolha do mape para objetivos e restrições traz resultados

mais promissores, e a escolha do spearaman resulta em uma variância menor. Analisando as duas configurações do CTAEA puro, é possível destacar uma vantagem otimizando por substitutos em todas as combinações.

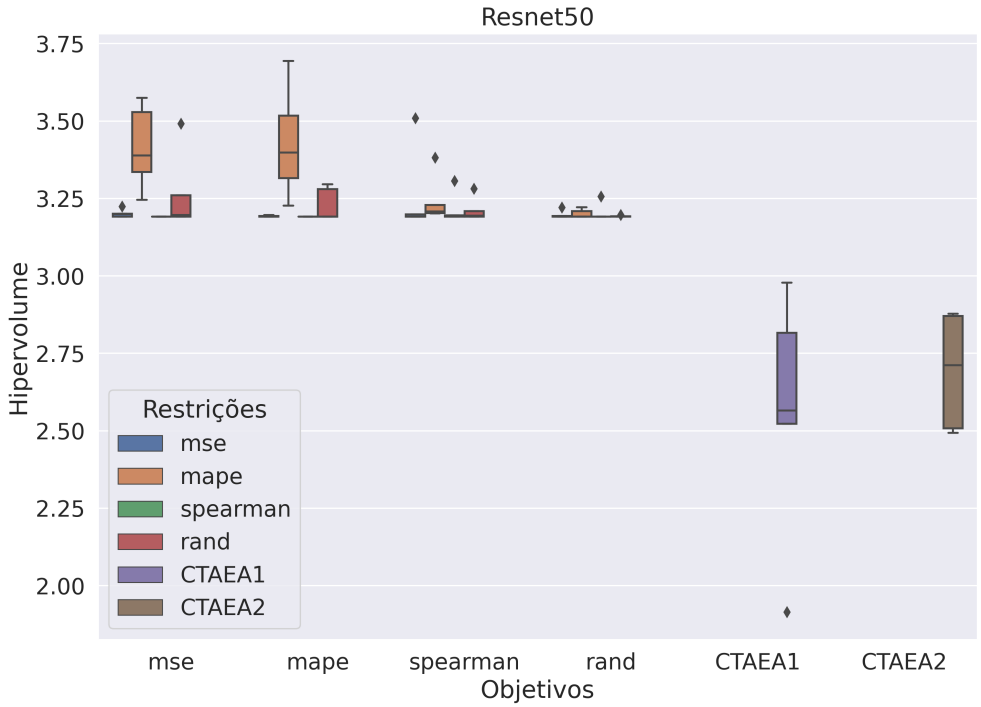

Figura 6.7 – Diagrama de caixas hiper-volumes Resnet50

Para demonstrar de forma geral, as tabelas presentes nas Figuras 6.8 e 6.9 contém informações a respeito da média (Me) e desvio padrão (DP) para valores de hipervolume nos 5 testes de cada uma das combinações. A partir das tabelas é possível perceber os padrões analisados anteriormente, a respeito da melhor combinação para hipervolume e variância.

VGG16												
Metricas	MSE		MAPE		SPEARMAN		RAND		CTAEA T1		CTAEA T2	
obj/restr	DP	Me	DP	Me	DP	Me	DP	Me	DP	Me	DP	Me
MSE	0,5950	4,9353	0,5688	5,7181	0,3816	4,8264	0,3685	4,6262	0,0119	0,8335	0,4034	4,9071
MAPE	0,6505	5,1082	0,3957	6,0016	0,5412	5,4859	0,3119	4,7719				
SPEARMAN	0,3103	5,5137	0,5386	6,0726	0,2415	5,3756	0,4425	5,0593				
RAND	0,3727	5,2932	0,5756	5,2280	0,6630	5,4051	0,7195	5,2573				

Figura 6.8 – Hipervolume VGG16

Os gráficos das Figuras 6.10 e 6.11, demonstram as soluções não dominadas da escolha da métrica mape para objetivos e restrições. O tempo de inferência foi calculado de forma externa no processo de infill-points.

Resnet50												
Métricas	MSE		MAPE		SPEARMAN		RAND		CTAEA T1		CTAEA T2	
obj/restr	DP	Me	DP	Me	DP	Me	DP	Me	DP	Me	DP	Me
MSE	0,0116	3,2013	0,1216	3,4147	0,0034	3,1916	0,1154	3,2661	0,6066	2,7192	0,1675	2,6921
MAPE	0,0017	3,1930	0,1628	3,4304	0,0026	3,1917	0,0473	3,2301				
SPEARMAN	0,1263	3,2566	0,0691	3,2444	0,0453	3,2159	0,0342	3,2134				
RAND	0,0112	3,1976	0,0123	3,2011	0,0256	3,2044	0,0017	3,1928				

Figura 6.9 – Hipervolume Resnet50

Em um primeiro momento, analisando a Figura 6.10, é possível perceber que o conjunto final possui apenas duas soluções não dominadas. As duas soluções encontradas, realizam uma poda superior a 99% das conexões, e ainda apresentam uma acurácia superior ao modelo original. As soluções comprimidas, apresentam um tempo de inferência cerca de 8 vezes menor que o modelo original. Além disso, uma solução possui quantização, o que promove uma redução no tamanho em megabytes(MB).

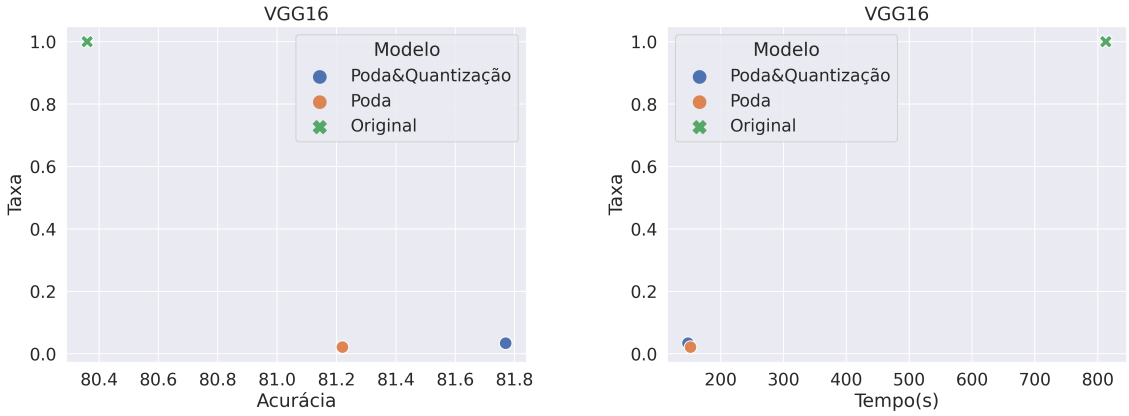

Figura 6.10 – Objetivos VGG16

Analisando para a Resnet50, na Figura 6.11, é possível perceber um conjunto mais amplos de soluções não dominadas. Diferente da VGG16, o processo de otimização não foi tão agressivo, removendo menos de 70% das conexões. No entanto, é possível perceber que todas soluções comprimidas apresentam um tempo de inferência menor que a rede neural original, em alguns casos, uma redução de quase 5 vezes no tempo. Além disso, algumas soluções comprimidas apresentaram acurácia maior que o modelo original. A maioria das soluções do conjunto final, possuem quantização, logo, existe uma redução do tamanho em megabytes nesses casos.

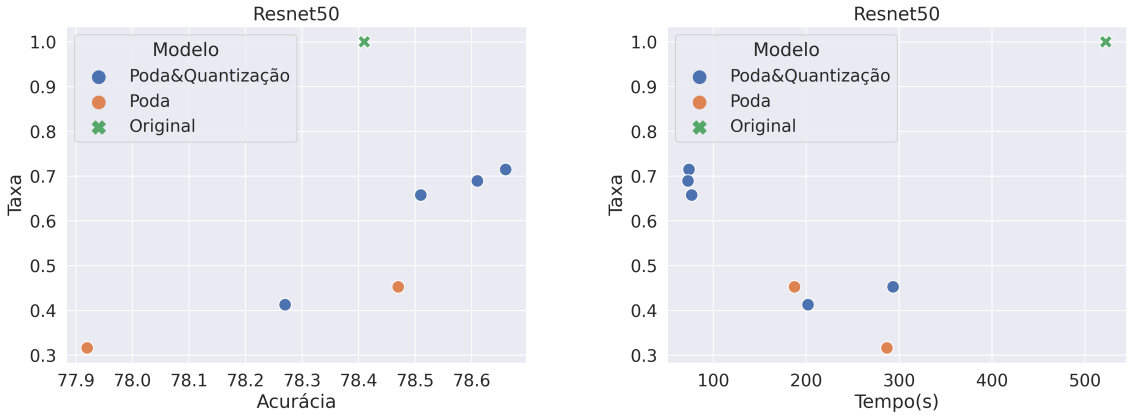

Figura 6.11 – Objetivos Resnet50

Apresentado os resultados, é perceptível que a otimização por modelos substitutos produz resultados promissores para o problema de compressão. Além disso, foi constatado a possibilidade de comprimir de forma significativa as duas redes neurais sem perda expressiva de acurácia.

Voltando a revisão de literatura, WANG et al. (2021), afirma que a compressão é mais eficaz para redes mais complexas, o mesmo pode ser observado neste trabalho, em que foi possível comprimir de forma mais agressiva a VGG16 que possui mais parâmetros que a Resnet50.

7 Considerações Finais

7.1 Conclusão

Modelos complexos, dificilmente podem ser implementados em dispositivos com restrições de memória e processamento. Sendo assim, o presente trabalho visa parametrizar técnicas de compressão de redes neurais, avaliando os efeitos no modelo comprimido. Através dos experimentos realizados, foi possível concluir que a mudança na escolha dos parâmetros de compressão como a intensidade da poda, altera aspectos importantes na rede neural, principalmente o tempo de inferência e a acurácia. É possível escolher os parâmetros de forma automatizada através da utilização de algorítimos de otimização multiobjetivo. No entanto, a avaliação das funções objetivo é um processo custoso, devido à necessidade de realizar inferências na rede. Nesse contexto, otimizar com modelos substitutos se tornou uma opção viável, apresentando resultados extremamente promissores. Restringir o problema multiobjetivo em relação a acurácia, tornou-se necessário, evitando o desperdício de tempo e recursos computacionais provenientes da avaliação de modelos ineficientes.

7.2 Trabalhos Futuros

A partir deste trabalho, novas pesquisas podem ser realizadas através de testes com diferentes base de dados e arquiteturas neurais, reforçando a eficiência estatística. Além disso, uma possível melhoria do algoritmo proposto, seria permitir a seleção de diferentes métricas para cada um dos objetivos, no processo de construção e otimização do problema substituto.

Referências

BAPTISTA, J. V. R. Aplicação de técnicas de xai em redes neurais convolucionais na classificação de lesões de pele. 2021.

BENDJILLALI, R. I.; BELADGHAM, M.; MERIT, K.; TALEB-AHMED, A. Illumination-robust face recognition based on deep convolutional neural networks architectures. *Indonesian Journal of Electrical Engineering and Computer Science*, v. 18, n. 2, p. 1015–1027, 2020.

BLALOCK, D.; ORTIZ, J. J. G.; FRANKLE, J.; GUTTAG, J. What is the state of neural network pruning? *Proceedings of machine learning and systems*, v. 2, p. 129–146, 2020.

Blank, J.; Deb, K. pymoo: Multi-objective optimization in python. *IEEE Access*, v. 8, p. 89497–89509, 2020.

CHEN, J.; RAN, X. Deep learning with edge computing: A review. *Proceedings of the IEEE*, IEEE, v. 107, n. 8, p. 1655–1674, 2019.

CSTORONTO. *The CIFAR-10 dataset*. s.d. Accessed: 2022-05-05. Disponível em: <https://www.cs.toronto.edu/~kriz/cifar.html>.

Data Science Academy. Data Science Academy, 2022. Disponível em: <https://www.deeplearningbook.com.br/>.

DEB, K.; JAIN, H. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints. *IEEE Transactions on Evolutionary Computation*, v. 18, n. 4, p. 577–601, 2014.

FERNANDES, M. A.; KUNG, H. A novel training strategy for deep learning model compression applied to viral classifications. In: IEEE. *2021 International Joint Conference on Neural Networks (IJCNN)*. [S.l.], 2021. p. 1–9.

FERREIRA, A. d. B. N. M. Otimização multiobjetivo baseada em modelos substitutos para compressão de redes neurais artificiais. 2022.

FLECK, L.; TAVARES, M. H. F.; EYNG, E.; HELMANN, A.; ANDRADE, M. d. M. Redes neurais artificiais: Princípios básicos. *Revista Eletrônica Científica Inovação e Tecnologia*, v. 1, n. 13, p. 47–57, 2016.

GOLDBARG, M. A. S. d. S. *Análise de técnicas de compressão em redes neurais profundas por poda em dataset de imagens*. Dissertação (B.S. thesis) — Universidade Federal do Rio Grande do Norte, 2021.

GORGENS, E. B.; LEITE, H. G.; SANTOS, H. d. N.; GLERIANI, J. M. Estimação do volume de árvores utilizando redes neurais artificiais. *Revista Árvore*, SciELO Brasil, v. 33, n. 6, p. 1141–1147, 2009.

HAN, S.; MAO, H.; DALLY, W. J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

HE, K.; ZHANG, X.; REN, S.; SUN, J. Deep residual learning for image recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2016. p. 770–778.

JURASZEK, G. D. et al. Reconhecimento de produtos por imagem utilizando palavras visuais e redes neurais convolucionais. Universidade do Estado de Santa Catarina, 2014.

LECUN, Y.; KAVUKCUOGLU, K.; FARABET, C. Convolutional networks and applications in vision. In: IEEE. *Proceedings of 2010 IEEE international symposium on circuits and systems*. [S.l.], 2010. p. 253–256.

LI, K.; CHEN, R.; FU, G.; YAO, X. Two-archive evolutionary algorithm for constrained multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, v. 23, n. 2, p. 303–315, 2019.

LUZ, E.; SILVA, P.; SILVA, R.; SILVA, L.; GUIMARÃES, J.; MIOZZO, G.; MOREIRA, G.; MENOTTI, D. Towards an effective and efficient deep learning model for covid-19 patterns detection in x-ray images. *Research on Biomedical Engineering*, Springer, v. 38, n. 1, p. 149–162, 2022.

MARIED, E.; ELDALI, M.; ZIADA, O.; BABA, A. *A Literature Study of Deep learning and its application in Digital Image Processing*. 2017.

MONARD, M. C.; BARANAUSKAS, J. A. Conceitos sobre aprendizado de máquina. *Sistemas inteligentes-Fundamentos e aplicações*, Manole, v. 1, n. 1, p. 32, 2003.

PEDREGOSA, F.; VAROQUAUX, G.; GRAMFORT, A.; MICHEL, V.; THIRION, B.; GRISEL, O.; BLONDEL, M.; PRETTENHOFER, P.; WEISS, R.; DUBOURG, V. et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, JMLR. org, v. 12, p. 2825–2830, 2011.

POINTER, I. *Programming PyTorch for Deep Learning: Creating and Deploying Deep Learning Applications*. [S.l.]: O'Reilly Media, 2019.

POUYANFAR, S.; SADIQ, S.; YAN, Y.; TIAN, H.; TAO, Y.; REYES, M. P.; SHYU, M.-L.; CHEN, S.-C.; IYENGAR, S. S. A survey on deep learning: Algorithms, techniques, and applications. *ACM Computing Surveys (CSUR)*, ACM New York, NY, USA, v. 51, n. 5, p. 1–36, 2018.

PYTORCH, A. D. I. Pytorch. 2018.

PYTORCH, P. *PRUNING TUTORIAL*. 2022. Accessed: 2022-05-01. Disponível em: <https://pytorch.org/tutorials/intermediate/pruning_tutorial.html>.

QUANTIZATION, P. *QUANTIZATION*. 2022. Accessed: 2022-05-05. Disponível em: <https://pytorch.org/docs/stable/quantization.html>.

RODRIGUES, C. F. Análise comparativa entre os métodos decomposição em valores singulares e análise de componentes principais envolvendo matrizes esparsas de grande porte. Universidade Federal de Minas Gerais, 2011.

SAMPAIO, P. R. *Teoria, métodos e aplicações de otimização multiobjetivo*. Tese (Doutorado) — Universidade de São Paulo, 2011.

SANTOS, A.; AIRES, K.; VERAS, R.; UCHÔA, V.; SANTOS, L. Uma abordagem de classificação de imagens dermatoscópicas utilizando aprendizado profundo com redes neurais convolucionais. In: SBC. *Anais do XVII Workshop de Informática Médica*. [S.l.], 2017.

SILVA, R. C.; LI, M.; RAHMAN, T.; LOWTHER, D. A. Surrogate-based moea/d for electric motor design with scarce function evaluations. *IEEE Transactions on Magnetics*, IEEE, v. 53, n. 6, p. 1–4, 2017.

SILVA, R. C. P. *Surrogate problem evaluation and selection for optimization with expensive function evaluations*. [S.l.]: McGill University (Canada), 2018.

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

SOARES, P. L. B.; SILVA, J. P. da. Aplicação de redes neurais artificiais em conjunto com o método vetorial da propagação de feixes na análise de um acoplador direcional baseado em fibra ótica. *Revista Brasileira de Computação Aplicada*, v. 3, n. 2, p. 58–72, 2011.

SUGATA, T.; YANG, C. Leaf app: Leaf recognition with deep convolutional neural networks. In: IOP PUBLISHING. *IOP Conference Series: Materials Science and Engineering*. [S.l.], 2017. v. 273, n. 1, p. 012004.

TONIN, M. V. P. Análise de quantização para codificação de redes neurais sem retreino. 2021.

VARGAS, A. C. G.; PAES, A.; VASCONCELOS, C. N. Um estudo sobre redes neurais convolucionais e sua aplicação em detecção de pedestres. In: SN. *Proceedings of the xxix conference on graphics, patterns and images*. [S.l.], 2016. v. 1, n. 4.

WANG, Z.; LUO, T.; LI, M.; ZHOU, J. T.; GOH, R. S. M.; ZHEN, L. Evolutionary multi-objective model compression for deep neural networks. *IEEE Computational Intelligence Magazine*, IEEE, v. 16, n. 3, p. 10–21, 2021.

WEIR, I. Spearman's correlation. Statstutor. *Mathematics Education Centre Loughborough University Available at:*, v. 17, 2017. Disponível em: <https://www.statstutor.ac.uk/resources/uploaded/spearmans.pdf>.