Abordagens Evolutivas para Otimização de Redes Neurais Convolucionais baseadas em Algoritmos Genéticos

Raphael de Lima Mendes

Universidade Federal de Uberlândia
Faculdade de Computação
Programa de Pós-Graduação em Ciência da Computação

Uberlândia
2021

Raphael de Lima Mendes

Abordagens Evolutivas para Otimização de Redes Neurais Convolucionais baseadas em Algoritmos Genéticos

Dissertação de mestrado apresentada ao Programa de Pós-graduação da Faculdade de Computação da Universidade Federal de Uberlândia como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Área de concentração: Ciência da Computação

Orientador: Laurence Rodrigues do Amaral

Uberlândia
2021

Ficha Catalográfica Online do Sistema de Bibliotecas da UFU
com dados informados pelo(a) próprio(a) autor(a).

M538
2021

Mendes, Raphael de Lima, 1993-
Abordagens Evolutivas para Otimização de Redes Neurais Convolucionais baseadas em Algoritmos Genéticos [recurso eletrônico] / Raphael de Lima Mendes. - 2021.

Orientador: Laurence Rodrigues do Amaral.
Dissertação (Mestrado) - Universidade Federal de Uberlândia, Pós-graduação em Ciência da Computação.
Modo de acesso: Internet.
Disponível em: http://doi.org/10.14393/ufu.di.2021.694
Inclui bibliografia.
Inclui ilustrações.

1. Computação. I. Amaral, Laurence Rodrigues do,1978-, (Orient.). II. Universidade Federal de Uberlândia. Pós-graduação em Ciência da Computação. III. Título.

CDU: 681.3

Bibliotecários responsáveis pela estrutura de acordo com o AACR2:

Gizele Cristine Nunes do Couto - CRB6/2091

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
Coordenação do Programa de Pós-Graduação em Ciência da Computação
Av. João Naves de Ávila, nº 2121, Bloco 1A, Sala 243 - Bairro Santa Mônica, Uberlândia-MG, CEP 38400-902
Telefone: (34) 3239-4470 - www.ppgco.facom.ufu.br - cpgfacom@ufu.br

ATA DE DEFESA - PÓS-GRADUAÇÃO

Programa de Pós-Graduação em:	Ciência da Computação				
Defesa de:	Mestrado Acadêmico, 28/2021, PPGCO				
Data:	17 de dezembro de 2021	Hora de início:	09:00	Hora de encerramento:	11:49
Matrícula do Discente:	12012CCP010				
Nome do Discente	Raphael de Lima Mendes				
Título do Trabalho:	Abordagens evolutivas para otimização de redes neurais convolucionais baseadas em Algoritmos Genéticos				
Área de concentração:	Ciência da Computação				
Linha de pesquisa:	Inteligência Artificial				
Projeto de Pesquisa de vinculação:	-				

Reuniu-se, por videoconferência, a Banca Examinadora, designada pelo Colegiado do Programa de Pós-graduação em Ciência da Computação, assim composta: Professores Doutores: Murillo Guimarães Carneiro - FACOM/UFU; Edimilson Batista dos Santos - DCOMP/UFSJ e Laurence Rodrigues do Amaral - FACOM/UFU, orientador do candidato.

Os examinadores participaram desde as seguintes localidades: Edimilson Batista dos Santos - São João Del Rei/MG; Murillo Guimarães Carneiro - Uberlândia/MG e Laurence Rodrigues do Amaral - Patos de Minas/MG. O discente participou da cidade de Uberlândia/MG.

Iniciando os trabalhos o presidente da mesa, Prof. Dr. Laurence Rodrigues do Amaral, apresentou a Comissão Examinadora e o candidato, agradeceu a presença do público, e concedeu ao Discente a palavra para a exposição do seu trabalho. A duração da apresentação do Discente e o tempo de arguição e resposta foram conforme as normas do Programa.

A seguir o senhor presidente concedeu a palavra, pela ordem sucessivamente, aos examinadores, que passaram a arguir o candidato. Ultimada a arguição, que se desenvolveu dentro dos termos regimentais, a Banca, em sessão secreta, atribuiu o resultado final, considerando o candidato:

Aprovado.

Esta defesa faz parte dos requisitos necessários à obtenção do título de Mestre.

Este trabalho é dedicado à minha noiva Tássia, aos meus familiares e amigos.

Agradecimentos

Só posso me sentir grato de estar aqui e poder apresentar este trabalho durante um período tão difícil para a nossa ciência e para a sociedade, com todo o contexto de pandemia. Gostaria de agradecer especialmente à minha noiva Tássia Melo, por todo o apoio em todas as frentes da minha vida. Ao meu *dog* filho Sherlock, por toda a alegria e amor. À minha família, Astride, Ingrid e Paulo, por sempre acreditarem em mim e me sinto privilegiado de poder dar esse orgulho para eles. Aos meus sogros, Aurileide e Luiz, que se tornaram meus segundos pais durante o período de pandemia em que moramos em Chã-Grande/PE e também ao meu cunhado Diego e sua esposa Lívia, que hoje são irmãos pra mim. Ao meu orientador, por todos os ensinamentos, paciência e exemplo de profissionalismo. À Bayer por permitir e apoiar a minha pós-graduação. Por fim, gostaria de agradecer a todos da PPGCO, especialmente Erisvaldo e aos professores que apresentaram flexibilidade e resiliência durante o período de aulas virtuais e à banca examinadora, pelas valiosas ponderações e contribuições para melhorar este trabalho.

"If you wish to succeed, you must brave the risk of failure."
(Garry Kasparov)

Resumo

As Redes Neurais Convolucionais (CNN) são consideradas o estado da arte em aplicações de visão computacional. Entretanto, a construção de modelos de aprendizagem de máquina utilizando essa tecnologia ainda requer a otimização de parâmetros, hiperparâmetros e desafios como a quantidade de dados para treinamento. Algoritmos Genéticos (AG) apresentam-se como uma técnica promissora para otimizar diversos aspectos das CNN. Este trabalho apresenta duas abordagens para otimização de CNNs utilizando AGs: gaCNN, para a otimização arquitetural e MLTLGA, para a otimização de aprendizagem por transferência. No método gaCNN, foi proposta uma codificação de cromossomo para incorporar funções de ativação e novos operadores de mutação, estas mudanças permitiram que o método superasse o desempenho de 9 dos 13 métodos avaliados em acurácia de classificação. No método MLTLGA, foi proposto um novo processo de inicialização que apresentou resultados superiores em pelo menos 2% de acurácia de classificação a todos os outros métodos de aprendizagem por transferência avaliados no trabalho. Dessa forma, apresentando métodos promissores no contexto de otimização de CNNs.

Palavras-chave: Aprendizagem de Máquina. Redes Neurais Convolucionais. Algoritmos Genéticos. Otimização. Neuroevolução.

Abstract

Convolutional Neural Networks (CNN) are considered the state-of-the-art in computer vision applications. However, building machine learning models such method still requires the parameter and hyperparameter tuning, and considerably large training datasets. Genetic Algorithms (GA) can be a promising technique to optimize various aspects of CNNs. In this work, two approaches to optimize CNN using GA are proposed: gaCNN, for architectural optimization and MLTLGA, for transfer learning optimization. In the gaCNN, a new individual codification strategy for activation functions is proposed alongside new mutation operators. The proposed method outperformed 9 out 13 methods evaluated in classification accuracy. In the MLTLGA, a new initialization operation is presented that outperformed by at least 2% the other transfer learning methods evaluated. Therefore, the two methods are promising in the study of optimization of CNN.

Keywords: Machine Learning. Convolutional Neural Networks. Genetic Algorithm. Optimization. Neuroevolution.

Lista de ilustrações

Lista de tabelas

Lista de algoritmos

Lista de siglas

AG Algoritmo Genéticos

AM Aprendizagem de Máquina

AT Aprendizagem por Transferência

CNN Redes Neurais Convolucionais

CE Computação Evolutiva

GPU Unidades de Processamento Gráfico

IA Inteligência Artificial

MLTLGA Algoritmo Genético para Aprendizagem de Transferência de Múltiplas Camadas

PSO Otimização por Enxame de Partículas

RNN Redes Neurais Recorrentes

ReLU Unidade Linear Retificadora

Tanh Tangente Hiperbólica

CPU Unidade de Processamento Central

UA Alinhamento de Unidade

UR Restauração de Unidade

UC Coleção de Unidades

WWO Otimização por Ondas

Sumário

Capítulo 1

Introdução

Algoritmos de Aprendizagem de Máquina (AM) têm sido utilizados de forma extensiva em uma série de áreas do conhecimento, como agricultura, logística, automobilística e saúde. Apesar da evolução do poder computacional e dos algoritmos de Inteligência Artificial (IA), a rede neural artificial (ROSENBLATT, 1958) ainda se mostra uma técnica relevante e considerada estado da arte em diversas aplicações, especialmente com o advento das redes neurais profundas (GOODFELLOW; BENGIO; COURVILLE, 2016), que são redes neurais artificiais com um maior número e tipos de camadas e conexões. Alguns exemplos de arquiteturas de redes neurais artificiais profundas são: Redes Neurais Convolucionais CNN (LECUN; BENGIO et al., 1995), Redes Neurais Recorrentes (RNN) (HOCHREITER; SCHMIDHUBER, 1997), Autoencoders (KRAMER, 1991) e Máquinas de Boltzmann (HINTON; SEJNOWSKI et al., 1986). Dentre estes tipos, as CNN se destacam como o estado da arte para aplicações em visão computacional, como classificação de imagens e identificação de objetos, sendo capazes muitas vezes de superar o desempenho de especialistas humanos em tarefas de detecção de objetos (JUNIOR; YEN, 2019). Entretanto, arquitetar e desenvolver CNN de alto desempenho ainda é considerado um problema desafiador e não resolvido devido ao número elevado de configurações de arquitetura, hiperparâmetros, custo computacional e especificidades dos problemas a serem resolvidos. Portanto, o desenvolvimento de CNN de bom desempenho requer a busca de diversas configurações, hiperparâmetros e funções de ativação. Dessa maneira, algoritmos bio-inspirados podem ser uma alternativa viável de identificar boas configurações, visto que esses algoritmos são capazes de explorar grandes espaços de busca, inspirados em elementos encontrados na natureza. Neste grupo, há diversos algoritmos, dentre estes, os Algoritmos Genéticos AG. O uso de IA na otimização de redes neurais artificiais é definido como Neuroevolução (MIIKKULAINEN et al., 2019).

1.1 Motivação

Um modelo de aprendizagem profunda pode ser definido como um otimizador em que alguns parâmetros podem ser manualmente selecionados, de modo que este modelo se comporte de maneira distinta. Desta maneira, dependendo do conjunto de parâmetros a serem ajustados, cada um destes conjuntos pode ser definido como uma tarefa distinta de otimização (MARTINEZ et al., 2021). As alterações destes parâmetros podem resultar em ganhos significativos de desempenho, entretanto, a tarefa de testar e identificar tais parâmetros é uma tarefa de tentativa e erro demorada e com alto custo computacional (WISTUBA; RAWAT; PEDAPATI, 2019).

O estudo de otimização de topologias e hiperparâmetros tem gerado interesse nos pesquisadores. Os desafios desse tipo de otimização estão relacionados principalmente à utilização de uma estratégia efetiva de codificação dos indivíduos para utilização de algoritmos bio-inspirados de forma eficiente (CUI; SHABASH; WIESE, 2019), além de operadores de cruzamento e mutação como no caso dos AG. Este é um fator importante, pois o custo de avaliação de cada indivíduo requer o treinamento de uma rede neural artificial, portanto, um número elevado de indivíduos na população ou um número elevado de gerações podem tornar o método proibitivo.

Outro grande desafio desta área de pesquisa é o desenvolvimento de um método mais eficiente de avaliação de indivíduos, o que permitiria um número maior de avaliações durante a execução dos algoritmos bio-inspirados.

Além disso, os tipos de topologias otimizadas restrigem-se a arquiteturas de CNN mais simples como a família de arquiteturas de redes VGG (SIMONYAN; ZISSERMAN, 2015) e LeNet (LECUN; BENGIO et al., 1995), porém o estado da arte de topologias de CNN inclui a família de redes como a *InceptionV3* (SZEGEDY et al., 2016).

Além dos desafios supracitados, os trabalhos desenvolvidos apresentam comparações com outros métodos bio-inspirados em relação ao desempenho dos modelos obtidos, utilizando métricas de comparação, tal como a acurácia de classificação. Entretanto, outros aspectos das CNN, como o número de parâmetros, custo computacional e tempo de execução podem ser explorados. Porém, o uso de AG na otimização de CNN não se restringe à busca por arquiteturas e hiperparâmetros apenas, mas também no processo de busca por parâmetros de treinamento de uma CNN, por exemplo (MARTINEZ et al., 2021).

Uma abordagem apresentada em trabalhos recentes propõe o uso de AG no processo de otimização da Aprendizagem por Transferência AT, ou paradigma Professor-Aluno (NAGAE; KAWAI; NOBUHARA, 2020). Esta abordagem é interessante porque ainda que CNN otimizadas sejam capazes de superar o desempenho do especialista humano em diversas aplicações, o treinamento destas redes normalmente requer uma quantidade elevada de dados de treinamento (WILLIAMS, 2019). Uma técnica comumente utilizada para atenuar este problema é o uso da AT (TORREY; SHAVLIK, 2010), que consiste em utilizar uma CNN treinada em um conjunto de dados original e utilizar os pesos e

a sua topologia em um conjunto de dados de interesse. A vantagem dessa abordagem para conjuntos de dados pequenos é que o desempenho do modelo gerado é normalmente melhor que treinar uma CNN a partir de pesos inicializados aleatoriamente.

A AT utiliza-se do conceito de que CNN tendem a aprender características mais gerais, como contornos, curvas e linhas nas camadas intermediárias e se tornam mais especializadas em formas mais complexas nas camadas mais profundas (LIU et al., 2016). Desta forma, pesquisadores normalmente adicionam algumas poucas camadas ao fim da CNN que passarão pelo processo de treinamento no conjunto de dados de interesse, enquanto as demais camadas possuem seus pesos fixos. Este processo reduz consideravelmente o tempo e custo computacional para criação de modelos, além de ser possível utilizar arquiteturas eficientes e distribuídas pela comunidade de AM.

Entretanto, estudos recentes demonstraram que o treinamento de algumas camadas intermediárias da CNN na AT podem melhorar o desempenho do modelo (SIMONYAN; ZISSERMAN, 2015). Desta maneira, selecionar estas camadas coloca-se como um novo desafio considerando que as CNN tendem a crescer em número de camadas e em tipos de conexão (NAGAE; KAWAI; NOBUHARA, 2020).

1.2 Objetivos e Desafios da Pesquisa

Considerando o impacto que as diferentes configurações de CNN podem ter no desempenho final de um modelo, métodos sistemáticos de otimização apresentam-se como uma alternativa promissora no desenvolvimento de modelos de AM utilizando CNN.

Métodos de otimização bio-inspirados têm demonstrado resultados promissores neste estudo, porém entende-se que seja necessário o aprimoramento constante destas técnicas devido, principalmente, ao elevado custo computacional.

O objetivo geral deste trabalho é desenvolver dois AG, sendo um algoritmo para Otimização de CNN, que seja capaz de alterar aspectos estruturais e hiperparâmetros para um determinado problema de classificação e outro algoritmo capaz de otimizar redes através da seleção das camadas do AT. Estes algoritmos devem ser competitivos em relação a outros métodos bio-inspirados e estruturas conhecidas, em relação ao tamanho das redes geradas, desempenho na classificação e custo computacional.

Os objetivos específicos são:

1. Desenvolver um AG para otimização do processo de AT;

2. Estender a técnica evoCNN com a introdução de otimização de funções de ativação e novos operadores genéticos;

3. Propor melhorias na representação cromossômica e dos operadores do evoCNN (SUN et al., 2020);

4. Comparar o método desenvolvido com métodos bio-inspirados semelhantes.

1.3 Hipótese

O algoritmo proposto por Sun et al. (2020) possui uma representação cromossômica robusta, e produz bons resultados na classificação de diversos *datasets*. Contudo, esta técnica pode ser estendida a outros aspectos importantes das CNN, como diferentes funções de ativação, e hiperparâmetros como o uso do *Dropout* e Normalização de Lotes otimizados através do AG. Além disso, entende-se que a otimização topológica e de hiperparâmetros é computacionalmente custosa, desta maneira, o uso de AG em outros aspectos de otimização, como no AT, pode tornar a técnica mais competitiva em aplicações do mundo real devido ao seu custo reduzido.

1. A adição da otimização das funções de ativação, normalização por lotes e *dropout* melhoram a técnica EvoCNN (SUN et al., 2020) em termos de acurácia de classificação?

2. A técnica proposta para otimização topológica e de hiperparâmetros é capaz de atingir um bom nível de generalização, ou seja, que pode ser aplicado em diferentes tipos de *datasets* comparado a outros métodos bio-inspirados e não bio-inspirados?

3. A técnica proposta para otimização de AT traz ganhos de desempenho de classificação em relação aos métodos convencionais?

4. A técnica proposta para otimização de AT apresenta melhor desempenho em relação a outros métodos bio-inspirados semelhantes?

1.4 Contribuições

Com o desenvolvimento deste trabalho foram realizadas as seguintes contribuições:

1. Aprimoramento da evoCNN (SUN et al., 2020) através da sua extensão à outros parâmetros de otimização;

2. Melhores taxas de classificação;

3. Aprimoramento das técnicas bio-inspiradas no âmbito do Aprendizado por Transferência.

1.5 Organização da Dissertação

No Capítulo 2 são apresentados os principais conceitos teóricos utilizados para o desenvolvimento do trabalho. No Capítulo 3 os principais trabalhos correlatos são abordados e o Capítulo 4 apresenta com detalhes os dois métodos propostos. No Capítulo 5 são apresentadas as configurações de cada experimento e como os mesmos foram conduzidos, assim como os resultados e uma análise para cada método. Por fim, o Capítulo 6 conclui o trabalho apresentando as principais contribuições e os trabalhos futuros.

Capítulo 2

Fundamentação Teórica

Neste capítulo serão apresentados os conceitos utilizados para sustentar o desenvolvimento deste trabalho. Este capítulo está dividido em 4 subseções. Na Subseção 2.1 serão abordados os conceitos teóricos sobre AG. Na Subseção 2.2 serão abordados conceitos teóricos sobre as CNN e, por fim, na Subseção 2.3 serão abordados conceitos teóricos sobre AT.

2.1 Algoritmos Genéticos

Entende-se que a dificuldade de resolução de um problema está diretamente relacionada à complexidade do seu espaço de busca. Dessa maneira, algoritmos exatos nem sempre são alternativas viáveis para instâncias consideráveis desses problemas devido ao seu elevado custo computacional. Desta forma, estratégias bio-inspiradas são amplamente utilizadas quando não há algoritmos exatos eficientes.

Os AG são métodos de otimização que pertencem à sub-área de IA denominada Computação Evolutiva (CE), que se caracteriza pela utilização de mecanismos inspirados na teoria da evolução das espécies. Inicialmente propostos por Holland (1968), e posteriormente, aprimorados por Goldberg (1989), os AG são definidos como métodos computacionais estocásticos de busca e otimização direcionados a partir de mecanismos de seleção natural. Seu aspecto estocástico se dá pela natureza probabilística da busca, como na estratégia de inicialização aleatória de indivíduos e na probabilidade de que soluções sofram alterações durante a execução do algoritmo devido ao seus diversos operadores (CARNEIRO et al., 2012).

Nos AG, uma população inicial de soluções evolui ao longo de gerações a partir de representações e operadores inspirados na biologia e genética. Uma característica fundamental desse processo evolutivo é que há, em média, uma tendência de convergência em direção às melhores soluções a cada geração, de modo que, ao atingir um critério de parada do processo evolutivo seja possível extrair um conjunto de soluções boas ou ótimas. Dentre os elementos bio-inspirados nos AG, podemos destacar:

- Gene: elemento constituinte de um cromossomo;
- Cromossomo: representação de um conjunto de genes;
- Indivíduo: estrutura de um ou mais cromossomos que representam uma potencial solução para o problema;
- População: conjunto de indivíduos a serem manipulados durante a execução do AG;
- Aptidão: métrica utilizada para avaliar o indivíduo dentro do espaço de busca;
- Função de Aptidão: função utilizada para calcular a aptidão do indivíduo;
- Seleção: processo de definição dos indivíduos que irão combinar-se para gerar novos filhos, ou progênie;
- Cruzamento: processo utilizado para combinar indivíduos;
- Reinserção: processo de formação da população da geração seguinte;
- Mutação: processo de alteração de indivíduos de modo a adicionar diversidade na população para evitar ótimos locais;
- Geração: iteração do AG.

O processo de execução do AG inicia-se a partir da geração de uma população aleatória, sendo possível em alguns casos aplicar heurísticas para criação de bons indíviduos inicialmente. Entretanto, essa estratégia pode limitar a capacidade do AG de explorar o espaço de busca. O número de indivíduos na população é controlado pelo parâmetro $Tpop$. A população será, então, submetida ao processo de avaliação, onde para cada indivíduo a sua aptidão é calculada. Utilizando-se da aptidão dos indíviduos, a seleção é realizada para a definição dos pais, que gerarão novos filhos ou progênie. A quantidade de pais e filhos é controlada a partir da Taxa de Cruzamento (Tc) e pode variar de acordo com a técnica utilizada para o cruzamento dos pais. O cruzamento é o processo de combinação de cromossomos de modo a gerar novos indíviduos, combinando genes dos pais. A mutação pode ocorrer nos filhos de acordo com a Taxa de Mutação ($Tmut$) de modo a adicionar diversidade na população. Os novos indivíduos são submetidos ao cálculo da aptidão e a nova população é formada de acordo com os critérios de reinserção. Esse processo se repete até que um critério de parada seja atingido, tais como: convergência à solução ótima, pouca variabilidade dos indivíduos ou numéro máximo de gerações. O melhor indivíduo da geração final é considerado a solução final do AG, porém, em alguns casos, como na otimização de CNN ou em AG multi-objetivo, um conjunto dos melhores indivíduos podem ser utilizados (AMARAL; OLIVEIRA, 2010). O fluxograma do ciclo de execução de um AG é ilustrado na Figura 1.

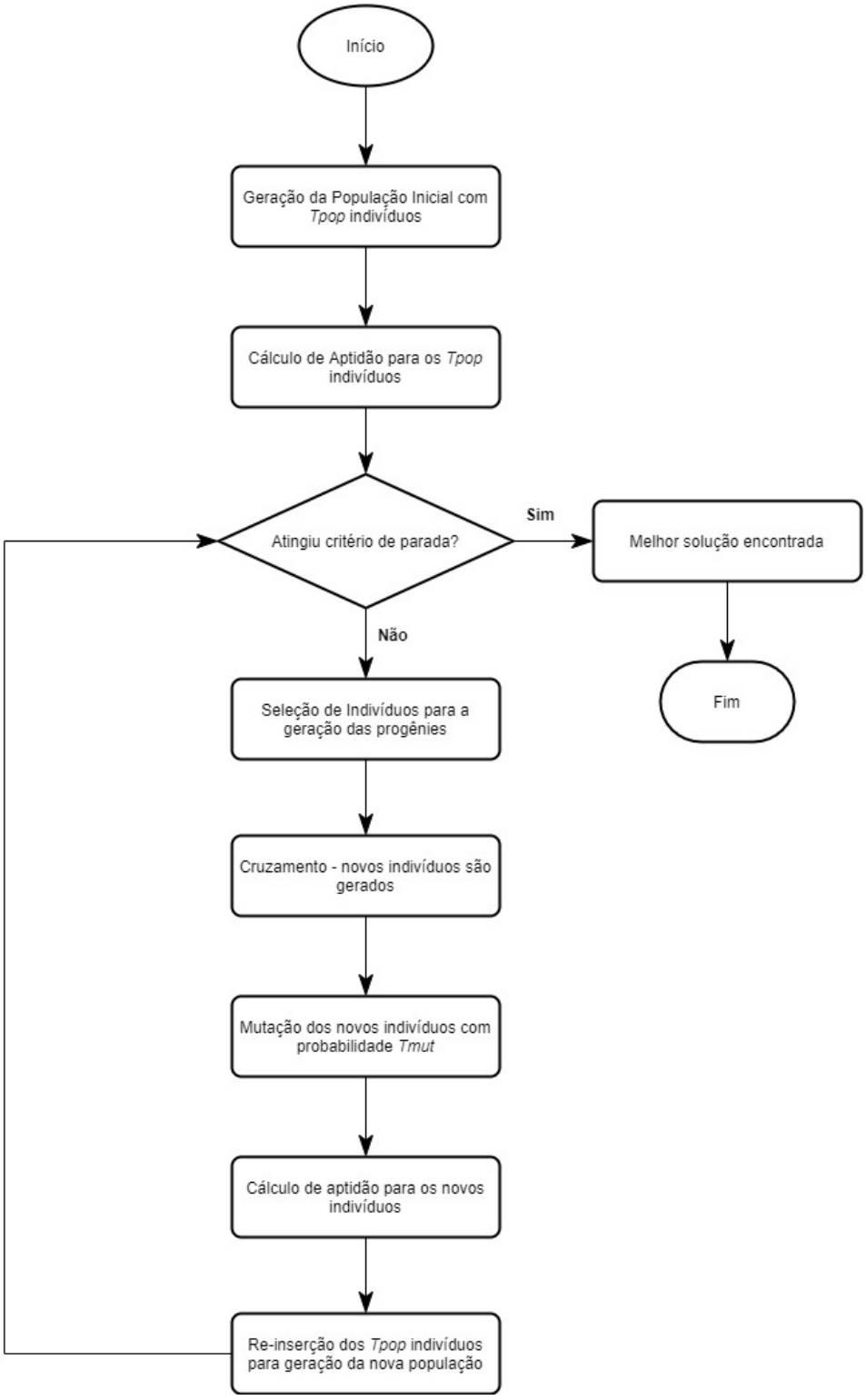

Figura 1 – Fluxograma do ciclo de execução de um AG. A cada geração novos indíviduos são gerados e avaliados de acordo com a função de aptidão, ao fim do AG a melhor solução final encontrada é o que possui maior aptidão.

2.1.1 Representação dos Indivíduos

A representação dos indivíduos é parte importante no desenvolvimento de um AG, já que esta deve ser uma solução válida para o problema investigado e deve ser contruída de modo que os operadores do AG possam ser utilizados. A representação clássica nos AG é a binária. Neste caso, o cromossomo é formado por um conjunto de genes que podem assumir valores binários, como pode ser observado na Figura 2. Além da representação binária, podemos destacar a representação do cromossomo a partir de genes com valores reais ou até mesmo inteiros, que pode ser utilizada para problemas de permutação, como o problema do Caixeiro Viajante. As representações de valores reais e inteiros podem ser observadas nas Figuras 3 e 4, respectivamente. As propostas de representação de indivíduos para otimização de CNN serão exploradas nos capítulos seguintes.

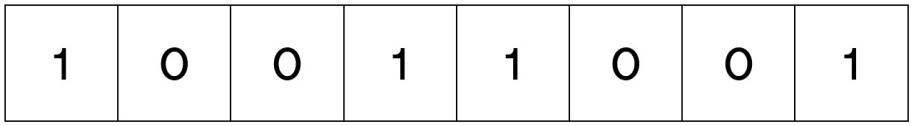

Figura 2 – Representação Binária de Indivíduo, onde cada espaço no vetor indica um elemento booleano.

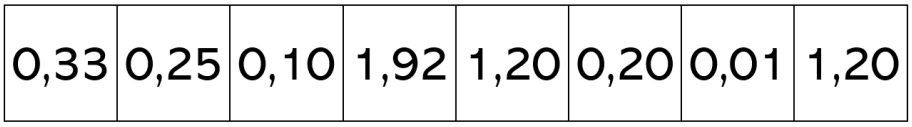

Figura 3 – Representação Real de Indivíduo, onde cada espaço no vetor indica um elemento do tipo real.

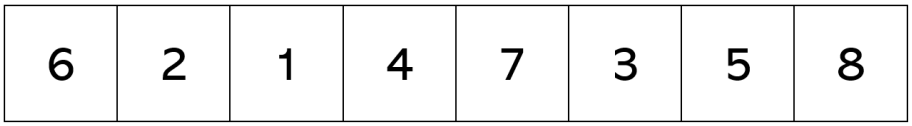

Figura 4 – Representação Inteira de Indivíduo, onde cada espaço no vetor indica um elemento do tipo inteiro.

2.1.2 Função de Aptidão

A função de aptidão é utilizada para calcular uma métrica que defina o valor de um indivíduo no espaço de busca. Esta métrica é utilizada pelos operadores genéticos para definição das chances de um indivíduo gerar uma progênie, através da seleção. Uma função de aptidão bem desenvolvida garante que indíviduos melhores possuam valores de aptidão consideravelmente maiores que os indivíduos piores da população, mas que ainda permita o AG fazer uma busca global sem acarretar numa convergência prematura. Além

disso, realizar o cálculo de maneira eficiente é importante visto que a função de aptidão pode representar boa parte do tempo de execução do AG.

2.1.3 Seleção

O operador de seleção é responsável por selecionar os indivíduos que gerarão as progênies. Este processo é uma metáfora que representa a principal característica da seleção natural: a sobrevivência dos indivíduos mais aptos. Neste caso, a aptidão é calculada a partir da Função de Aptidão. Na seleção, os indivíduos mais bem avaliados devem ser os mais propensos a perpetuarem sua informação genética, porém ainda com possibilidade de seleção de indivíduos menos aptos com o intuito de garantir uma variabilidade genética na população.

Um dos mais tradicionais métodos de seleção é o torneio, sendo uma estratégia simples e eficiente. Neste método, t indivíduos da população são selecionados aleatoriamente ou através de uma roleta (torneio estocástico) e, dentre eles, o que possuir maior aptidão ou for selecionado na roleta, respectivamente, será utilizado no processo de cruzamento. Um parâmetro importante a ser definido antes da execução do AG é o $Tour$, que indica o número de indivíduos no torneio. O torneio é realizado até que se obtenha a quantidade de indivíduos necessários para o cruzamento.

2.1.4 Cruzamento

O operador de cruzamento é responsável por combinar soluções (pais) e criar filhos, ou progênie. O cruzamento, ou *crossover* ocorre, normalmente, através da reprodução sexuada, com a combinação de genes dos dois pais para formação do(s) novo(s) indivíduo(s). Dentre os métodos mais utilizados para o cruzamento, pode-se destacar o cruzamento de 1 ponto, 2 pontos, k pontos e uniforme (HOLLAND, 1992). Entretanto, dependendo da representação do indivíduo pode ser necessária a elaboração de um método específico. A Figura 5 ilustra como é realizado o cruzamento de 1 ponto, utilizando dois indivíduos escolhidos a partir do operador seleção. Para efetuar o cruzamento um ponto de corte é sorteado e há a troca de genes a partir deste ponto. Neste caso dois pais irão gerar dois filhos.

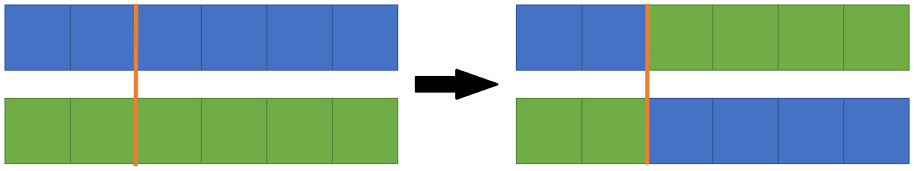

Figura 5 – Cruzamento de 1 ponto. Neste processo, uma posição do vetor de pais é escolhida aleatoriamente como ponto inicial para a troca de genes dos pais (esquerda), gerando os filhos (direita).

2.1.5 Mutação

Como supracitado, o operador de seleção é responsável por escolher indíviduos com boa aptidão para a perpetuação de bons genes no processo de busca por melhores soluções. Entretanto, um processo muito elitista não é desejável pois a variabilidade na população permite que o AG faça uma busca mais ampla, varrendo uma maior parte do espaço de busca. De modo a atenuar esse efeito da seleção, utiliza-se o operador de mutação. O operador de mutação é responsável por fazer mudanças aleatórias nos filhos gerados pelo cruzamento. A probabilidade de mutação é definida pelo parâmetro $Pmut$. Dentre os métodos de mutação mais comuns estão o de alteração de 1 bit no AG com representação binária, ilustrado na Figura 6 e a mutação através da troca de genes do cromossomo em problemas de permutação, ilustrado na Figura 7.

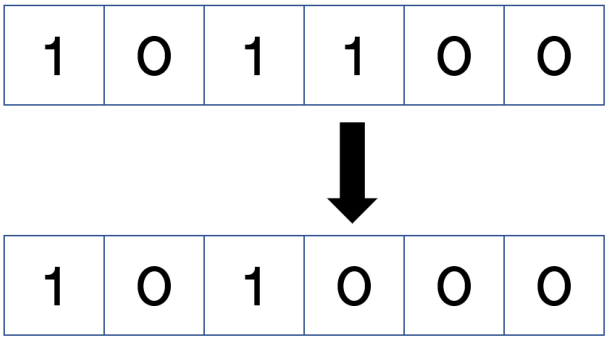

Figura 6 – Mutação através da alteração de 1 bit do cromossomo.

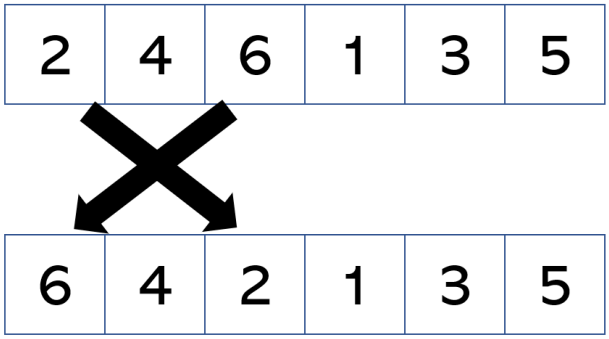

Figura 7 – Mutação através da troca de genes.

2.1.6 Reinserção

O operador de reinserção é responsável por gerar a população da nova geração através de alguma estratégia de seleção dos indivíduos da geração atual e dos seus filhos. Em alguns casos, é possível que uma porcentagem dos indivíduos seja formada por indivíduos aleatórios (NAGAE; KAWAI; NOBUHARA, 2020). Uma estratégia comum, denominada elitista, é a seleção dos $Tpop$ melhores indivíduos do conjunto população anterior + pais +

filhos para formar a nova população. Dessa maneira, os melhores indivíduos de gerações anteriores não são perdidos. O processo elitista é ilustrado na Figura 8.

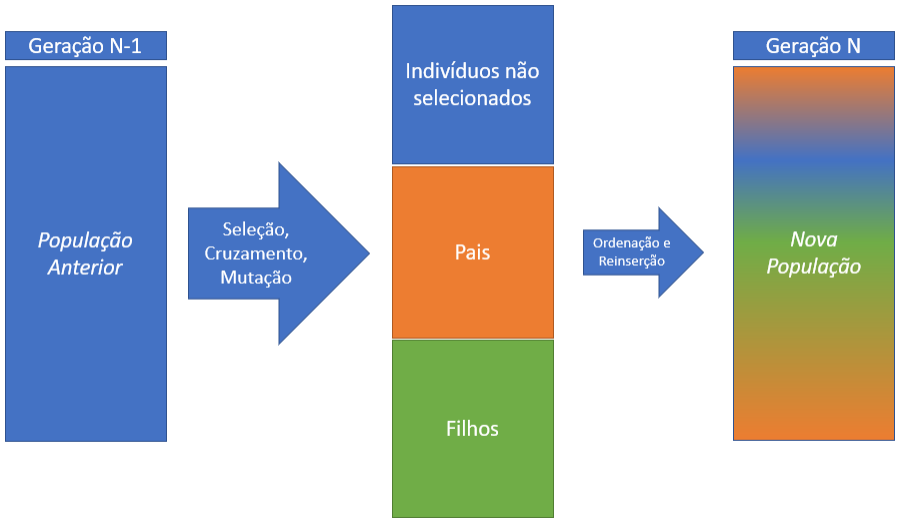

Figura 8 – Processo elitista de reinserção.

2.2 Redes Neurais Convolucionais

Uma CNN é um tipo de rede neural artificial com três tipos de camadas: convolucional, *pooling* e completamente conectadas. Basicamente, a camada convolucional é responsável por aplicar filtros através de convoluções na informação de entrada. Este tipo de operação é normalmente aplicada em problemas de visão computacional de larga escala devido a característica das imagens de que pixels próximos possuem uma boa correlação e tendem a formar objetos ou formas conhecidas que podem ser úteis na tarefa de classificação ou detecção de objetos (RUSSAKOVSKY et al., 2015). As camadas de *pooling* são utilizadas para reduzir a dimensionalidade de uma janela de pixels, através de técnicas como a média (WANG et al., 2012) ou valor máximo dos pixels da vizinhança (BOUREAU; PONCE; LECUN, 2010). As camadas completamente conectadas podem ser vistas como uma CNN, em que todos os neurônios da rede são conectados. Esta parte da arquitetura da CNN atua como o classificador ou estimador, enquanto o conjunto de camadas convolucionais e de *pooling* são comumente denominados extratores de informação (RECENT. . . , 2018). Uma CNN é ilustrada na Figura 9.

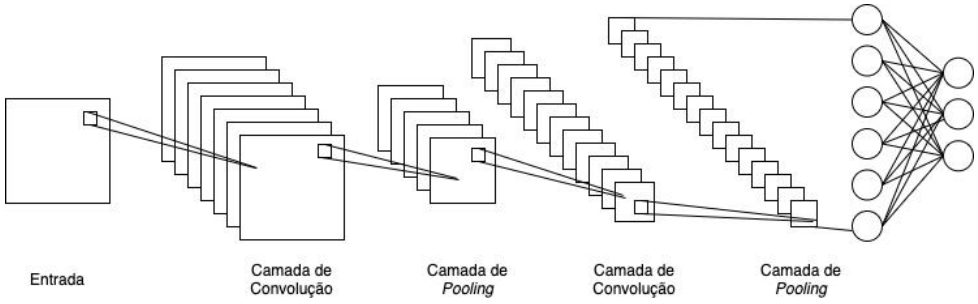

Figura 9 – Rede Neural Convolucional Convencional.

Ainda que as CNN sejam muito utilizadas em tarefas de visão computacional com imagens bidimensionais (EGMONT-PETERSEN; RIDDER; HANDELS, 2002), o processo de convolução pode ser unidimensional ou ter três ou mais dimensões, sendo o primeiro normalmente utilizado em tarefas como estudo de sequências e o tridimensional tem sido utilizado em tarefas de visão computacional com vídeos ou sequências de imagens (*frames*) (REDMON et al., 2016). Desta maneira, a operação de convolução 1D, 2D ou 3D são análogas, alterando-se apenas o número de dimensões em que as convoluções acontecem. Como citado anteriormente, a convolução 2D é normalmente utilizada em tarefas de visão computacional em imagens. A matriz ou filtro de dimensões i x j, atravessa a informação de entrada deslizando da esquerda para a direita e de cima para baixo até que toda a imagem seja coberta, gerando um mapa de características na saída. A operação é ilustrada na Figura 10. Este mapa normalmente possui uma dimensão inferior a informação de entrada a não ser que parâmetros como *padding* sejam aplicados. Os resultados no mapa de características são a soma dos produtos numa posição específica do filtro na imagem ou camada anterior. Matematicamente, o valor na posição (i, j) do k-ésimo mapa de características na camada l, $z^l_{i,j,k}$, pode ser calculado através da equação 1 (RECENT. . . , 2018).

$$z^l_{i,j,k} = \mathbf{w_k^l}^{\mathbf{T}} \mathbf{x_{i,j}^l} + b^l_k \tag{1}$$

Onde $\mathbf{w_k^l}$ e b^l_k são a matriz de pesos e de viéses, respectivamente, na camada l e no k-ésimo mapa de característica. A entrada da convolução na posição (i, j) é dada por $\mathbf{x_{i,j}^l}$ e a matriz de pesos é compartilhada na geração de um mapa de característica $z^l_{i,j,k}$. Normalmente, os parâmetros associados à convolução são: número de mapas de características, dimensão do filtro, largura e altura do passo *(stride)* além de *padding*.

Além disso, os valores passados através das camadas são conectados a partir de funções de ativação, a. O propósito de utilizar este tipo de operação é a introdução de propriedades não-lineares à rede neural artificial. Há diversos tipos de funções de ativação, como as funções Sigmoide e Tangente Hiperbólica (Tanh). Em CNNs, o tipo mais utilizado é a Unidade Linear Retificadora (ReLU). O valor de ativação, $a^l_{i,j,k}$, do elemento $z^l_{i,j,k}$ pode

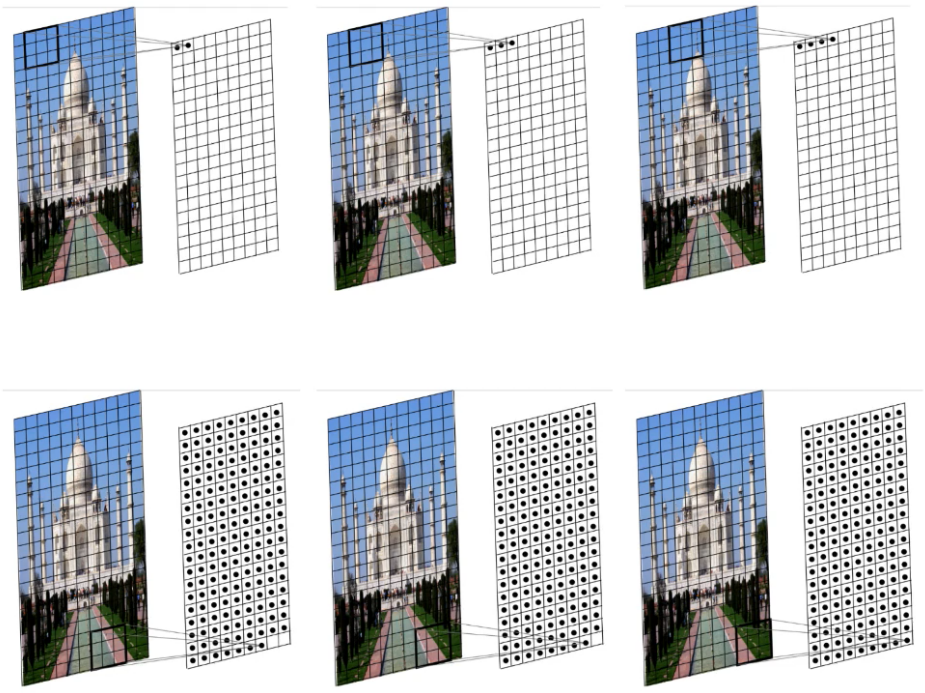

Figura 10 – Ilutração da operação de convolução sendo aplicada numa imagem, o filtro atravessa a imagem de modo a gerar um mapa de características.

Fonte: Kumar (2019).

ser obtido através da equação 2.

$$a^l_{i,j,k} = a(z^l_{i,j,k}) \tag{2}$$

Nas CNN, normalmente, cada camada de convolução é seguida de uma camada de *Pooling*. O objetivo do uso do *Pooling* é reduzir a complexidade dos relacionamentos na informação de entrada, de modo a aprender representações úteis, reduzindo o custo computacional de uma CNN. O processo é análogo à convolução em sua execução, porém neste caso, um filtro é aplicado numa janela de pixels e usualmente o valor médio ou máximo de pixels naquela janela são extraídos para a geração do mapa de características seguinte. Os parâmetros desta operação são: número de mapas de características, dimensão do filtro, largura e altura do passo *(stride)*, *padding* e tipo de *pooling*. Matematicamente, esta função de *pooling*, *pool*, pode ser descrita a partir da equação 3.

$$y^l_{i,j,k} = pool(a^l_{m,n,k}), \forall (m,n) \in N_{i,j} \tag{3}$$

Onde $N_{i,j}$ é a vizinhança de (i,j). A estrutura de uma CNN convencional é um conjunto de sequências de convoluções, *pooling* seguidas de pelo menos uma camada completamente conectada unidimensional. Além das características estruturais, uma rede neural artificial precisa ser otimizada, ou treinada. Esta otimização ocorre através da estimação dos pesos e viéses da rede neural artificial e dos filtros de convolução em CNN,

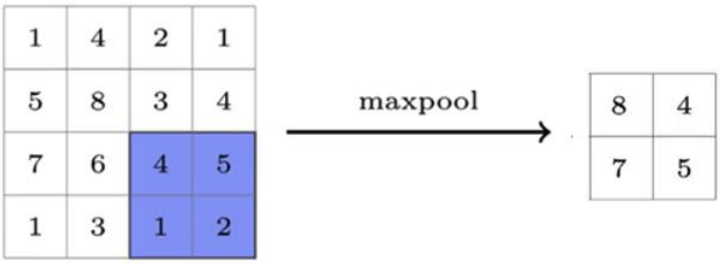

Figura 11 – Ilustração de uma operação de pooling do tipo máximo, com filtro 2x2 e passo de 2.

conhecidos como parâmetros treináveis. Esta estimativa é obtida iterativamente utilizando um algoritmo de otimização. Comumente é utilizado o Gradiente Descendente Estocástico (WIJNHOVEN; WITH, 2010), com o objetivo de minimizar uma função de perda, ajustando os parâmetros da rede através do *Backpropagation* (ROJAS, 1996). Além dos hiperparâmetros e arquitetura da rede, os parâmetros de treinamento como taxa de aprendizado, número de épocas e momento são fundamentais no desempenho final da rede.

2.2.1 Arquiteturas Conhecidas de CNN

No estudo das CNN, comumente as arquiteturas e parâmetros das redes estão associadas ao problema enfrentado. Entretanto, pesquisadores têm buscado arquiteturas que sejam capazes de obter bons níveis de generalização para diversos problemas, ou que funcionem como ponto de partida no desenvolvimento de um novo modelo de aprendizagem de máquina. Dentre as famílias de CNN, há arquiteturas mais simples como a LeNet, AlexNet (KRIZHEVSKY; SUTSKEVER; HINTON, 2012), e VGGNet (SIMONYAN; ZISSERMAN, 2015) e arquiteturas mais complexas que propõem novos tipos de conexões e número elevado de camadas, como a ResNet. O desenvolvimento dessas arquiteturas representou importantes avanços em relação ao maior número de camadas com desempenho computacional aceitável, enfrentando problemas como a explosão ou desaparecimento do gradiente que as redes CNN convencionais enfrentavam uma vez que o número de camadas crescia.

Outra importante contribuição destas arquiteturas conhecidas, é a utilização de redes pré-treinadas no AT, ou professor-aluno, em problemas em que o conjunto de dados é insuficiente para treinar uma rede neural artificial a fim de obter um bom desempenho.

2.2.1.1 LeNet e AlexNet

A AlexNet foi proposta por Krizhevsky, Sutskever e Hinton (2012) e venceu a competição de reconhecimento de imagens em larga escala, ImageNet (RUSSAKOVSKY et al., 2015). A AlexNet é uma evolução de outra CNN, a LeNet, proposta por LeCun,

Bengio et al. (1995), ilustrada na Figura 12. A arquitetura da rede AlexNet é ilustrada na Figura 13. Para o desenvolvimento desta rede foi utilizada a aceleração a partir de Unidades de Processamento Gráfico (GPU), o que possibilitou a construção de uma rede mais complexa que a LeNet e de melhor desempenho, representando um grande avanço no estudo de redes neurais artificiais, que até então possuíam custo computacional proibitivo quando treinadas em Unidade de Processamento Central (CPU).

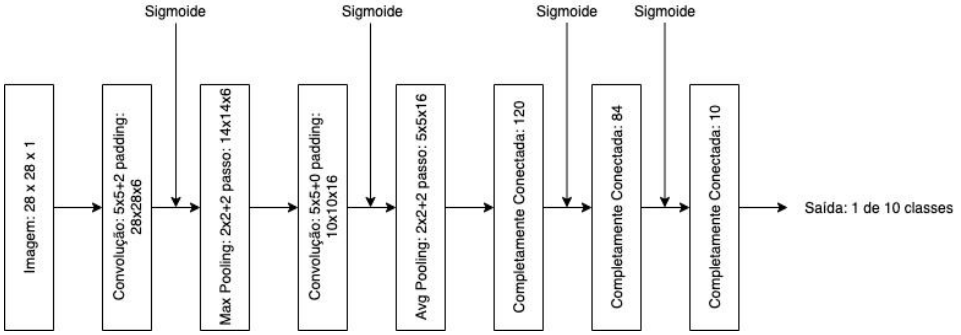

Figura 12 – Ilustração da arquitetura LeNet.

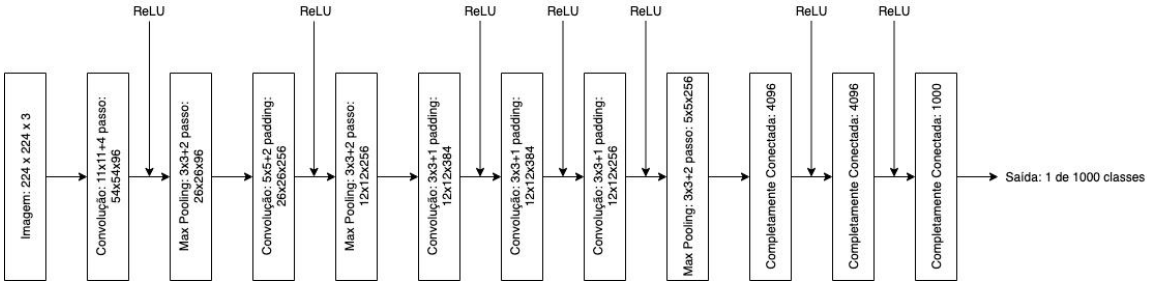

Figura 13 – Ilustração da arquitetura AlexNet.

2.2.1.2 VGGNet

A VGGNet (SIMONYAN; ZISSERMAN, 2015) foi proposta em 2014 e se assemelha a AlexNet (KRIZHEVSKY; SUTSKEVER; HINTON, 2012) em sua estrutura, porém com um número maior de camadas. O número de camadas varia de 16 (VGG-16, Figura 14) a 19 (VGG-19) camadas. O número elevado de camadas requer cuidados para evitar superajuste, ou *overfitting*.

2.2.1.3 ResNet

A rede profunda residual (ResNet) é uma das CNN mais comumente utilizadas. Sua principal característica é o bloco residual, ilustrado na Figura 15, no qual a linha curva representa uma conexão de atalho, ou *shortcut*. Com a introdução deste bloco, o problema de explosão ou desparecimento de gradiente causado pelo aumento no número de camadas

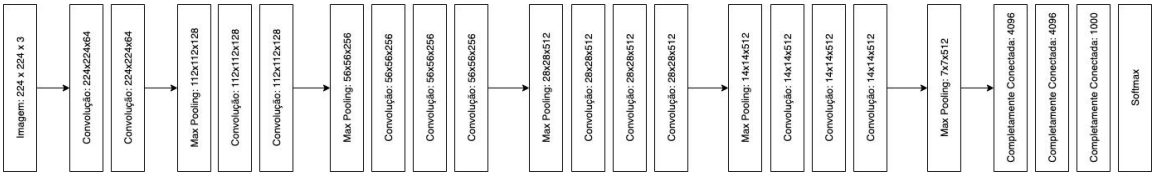

Figura 14 – Ilustração da arquitetura VGG-16. É possível notar um padrão nos blocos, constituindo-se de uma camada de *pooling* seguida de 3 camadas de convolução. Todas as camadas de convolução são seguidas da função de ativação ReLU.

foi resolvido. He et al. (2016a) mostraram que as ResNets (incluindo ResNet-18, ResNet-34, ResNet-50, ResNet-101 e ResNet-152, cujas diferenças estão no número de camadas) possuem melhor desempenho que outras CNNs no conjunto de dados *ImageNet*, indicando que as características das imagens podiam ser bem extraídas por essa família de redes neurais artificiais. A arquitetura ResNet-18 é ilustrada na Figura 16.

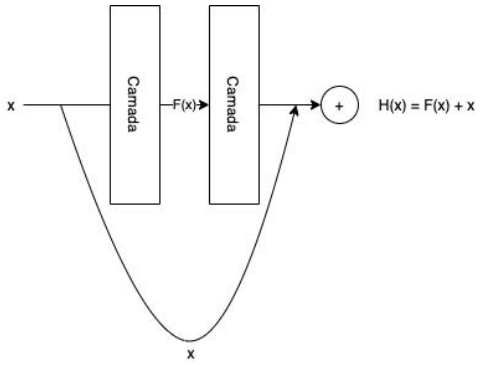

Figura 15 – O bloco residual possui uma conexão de atalho, que representa a matriz identidade.

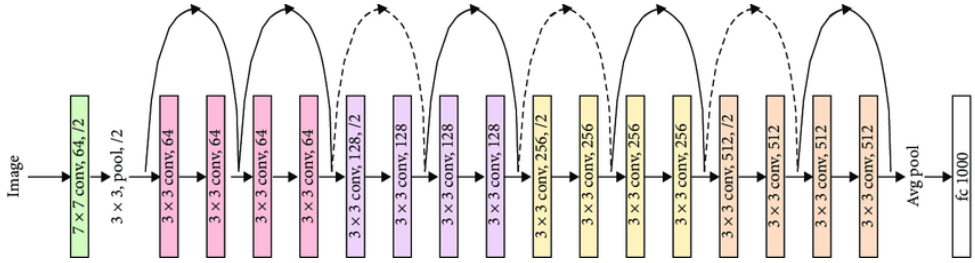

Figura 16 – Arquitetura da rede ResNet-18.

2.2.1.4 InceptionV3

A InceptionV3 é uma arquitetura de CNN que faz parte da família de CNN Inception. A principal característica das redes Inception é a repetição de blocos ao longo da rede neural artificial. Estes blocos são formados por diversos filtros de convolução de diferentes tamanhos que são concatenados para produzirem a saída do bloco. Esse tipo de

estratégia permite uma computação mais eficiente e de maior abrangência nos dados de entrada de uma determinada camada. A rede InceptionV3 é formada por 48 camadas e é comumente utilizada nas abordagens de AT (SZEGEDY et al., 2016). A InceptionV3 pode ser observada na Figura 17.

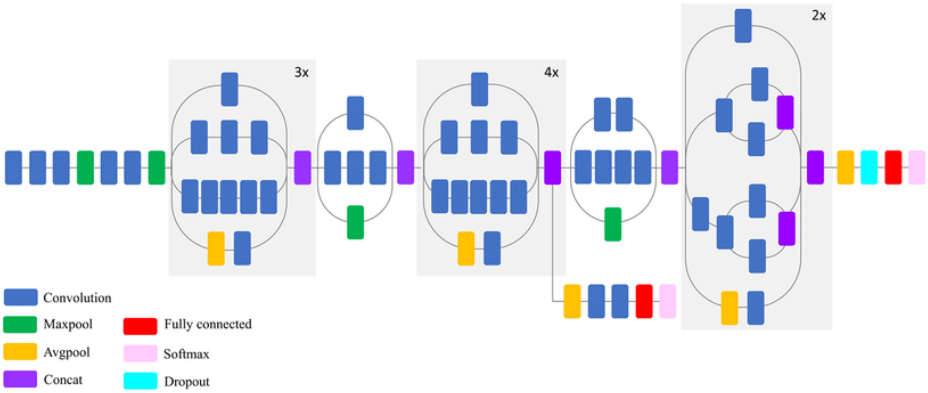

Figura 17 – Arquitetura InceptionV3, pode-se observar a repetição dos blocos inception ao longo da rede.

Fonte:Mahdianpari et al. (2018).

2.2.2 Aprendizagem por Transferência

Ainda que as CNN sejam capazes de superar até especialistas humanos em aplicações de visão computacional, o treinamento destas redes normalmente exigem uma quantidade elevada de dados (WILLIAMS, 2019). Uma técnica comumente utilizada para acelerar o processo de treinamento é o uso da AT (TORREY; SHAVLIK, 2010).

Esta técnica consiste no aproveitamento dos pesos de uma CNN treinada em um conjunto de dados fonte, habilitando o treinamento no conjunto de dados de interesse em apenas algumas camadas, normalmente as camadas finais, no conjunto de dados de interesse. A principal vantagem desta abordagem é que para conjuntos de dados reduzidos, o desempenho do modelo é superior ao de treinar uma CNN a partir de pesos aleatórios.

Esta técnica utiliza o conceito de que CNN possuem a tendência de aprender características mais básicas, como formatos simples, nas camadas iniciais e se especializam ao longo da CNN (LIU et al., 2016). Nesta abordagem, os pesquisadores adicionam algumas camadas no fim de uma CNN que será treinada e a parte inicial da CNN atua como um extrator de características, na Figura 18 o processo é ilustrado de maneira simplificada. Além disso, diminui os requisitos computacionais e reduz o tempo de treinamento.

Estudos revelaram que utilizar diferentes camadas intermediárias da CNN como treináveis pode elevar o desempenho do modelo (SIMONYAN; ZISSERMAN, 2015). Porém,

com o número elevado de camadas das CNN mais avançadas, a seleção dessas camadas se torna algo não trivial. Dentre os métodos evolucionários propostos para automaticamente selecionar as camadas de treinamento, três métodos se destacam: *Pathnet* (FERNANDO et al., 2017), *StepwisePathNet* (IMAI; KAWAI; NOBUHARA, 2020) e o método proposto por Nagae e colaboradores (NAGAE; KAWAI; NOBUHARA, 2020).

O *Pathnet* é um método de seleção de camadas que utiliza AG. Este método se restringe a apenas um tipo de estrutura de CNN, utilizando seleção por torneio. O *StepwisePathNet* é uma extensão do método PathNet para estruturas gerais de CNN. O método de Nagae e colaboradores também utiliza AG, e assim como o *StepwisePathNet*, generaliza bem para diferentes configurações e arquiteturas de CNN, apresentando quatro métodos de seleção e um novo método de cruzamento.

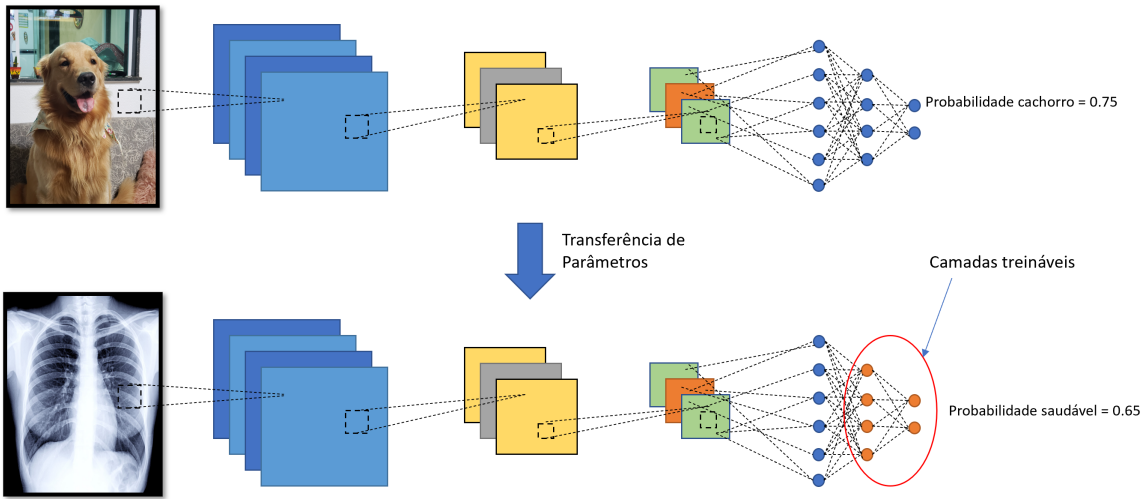

Figura 18 – Ilustração simplificada do processo de AT.

Capítulo 3

Trabalhos Correlatos

O desenvolvimento de CNN otimizadas requer uma busca profunda numa série de configurações, hiperparâmetros e funções de ativação. Portanto, algoritmos bio-inspirados podem ser uma boa alternativa para identificar CNN com bom desempenho preditivo, pois estes algoritmos são capazes de explorar grandes espaços de busca paralelamente. Neste grupo, podemos destacar os AG, Otimização por Enxame de Partículas (PSO), Otimização por Ondas (WWO), Otimização baseada em bio-geografia e Otimização de Colônia de Formigas.

Diversas propostas foram submetidas para otimizar redes neurais artificiais. Whitley, Starkweather e Bogart (1990) propôs o uso de AG para otimizar os pesos da rede neural. Zhou et al. (2019) propuseram o uso do algoritmo WWO para otimizar pesos e viéses das conexões, assim como o número de neurônios nas camadas ocultas de redes neurais rasas e profundas. Cui, Shabash e Wiese (2019) propuseram o uso de algoritmos evolucionários para evoluir redes com funções de ativação heterogêneas. Em 2002, a NeuroEvolução por Aumento de Topologias (NEAT) (STANLEY; MIIKKULAINEN, 2002) foi proposta. A NEAT utiliza AGs para evoluir redes inicialmente pequenas em redes com maior complexidade ao longo das gerações, alterando seus pesos e conexões. Sinha, Haidar e Verma (2018) propuseram um PSO para ajustar hiperparâmetros. Outros trabalhos propuseram a poda das redes, considerando que redes menores com desempenho semelhante são preferíveis a redes maiores e mais complexas, pois diminui-se o custo computacional para treinamento e inferência (WU et al., 2019).

Nas CNN, modificações arquiteturais resultam em ganhos significativos, entretanto, a tarefa de identificar estas estruturas é onerosa e sujeita a erros (WISTUBA; RAWAT; PEDAPATI, 2019). Portanto, pesquisadores da comunidade de Aprendizagem de Máquina têm estudado a automação deste processo. Chen et al. (2020) e Zoph et al. (2018) propuseram a utilização de aprendizagem por reforço como técnica para evoluir redes neurais. Recentemente, Real et al. (2017), Liu et al. (2018) e Wistuba (2018) apresentaram um método utilizando busca celular evolucionária, que consiste em encontrar repetições de estruturas fixas, ou células, que podem ser combinadas para gerar arquiteturas maiores e

mais poderosas. Entretanto, essa abordagem requer um poder computacional de milhares de horas de GPU, o que torna o método impraticável para a maior parte da comunidade de pesquisa.

Desta maneira, AG podem ser uma alternativa competitiva em relação a outros métodos de neuroevolução desde que uma boa estratégia de representação do gene seja elaborada, além de uma estratégia de busca eficaz para encontrar arquiteturas boas em um tempo não proibitivo.

Neste capítulo, as técnicas que foram utilizadas como base para este trabalho serão exploradas, o método EvoCNN (SUN et al., 2020) e o método proposto por Nagae, Kawai e Nobuhara (2020) nas Seções 3.1 e 3.2, respectivamente.

3.1 EvoCNN

3.1.1 Introdução

Neste método, um AG é proposto para a evolução de arquitetura de CNN com estruturas baseadas nas redes VGG. Além dos parâmetros de arquitetura, como o número de camadas e tipos de camadas, esta técnica também utiliza a inicialização dos pesos da rede no processo de evolução.

Algoritmo 1 Rotina Principal do EvoCNN

1: $P_0 \leftarrow$Inicializar População;
2: $t \leftarrow 0$;
3: $N \leftarrow n$;
4: **while** critério de parada não satisfeito **do**
5: Calcular Aptidão dos indivíduos da P_t;
6: $S \leftarrow$Selecionar Pais para *Crossover*;
7: $Q_t \leftarrow$Gerar Filhos;
8: $P_{t+1} \leftarrow$Criar nova População a partir da Reinserção $(P_t \cup Q_t)$;
9: $t \leftarrow t + 1$;
10: **end while**
11: $P_{best} \leftarrow$Selecionar o indivíduo com melhor aptidão;

O Algoritmo 1 apresenta a rotina principal do método EvoCNN. Inicialmente, a população é inicializada, utilizando uma codificação de gene específica e algumas restrições, que serão abordadas adiante. O processo de evolução ocorre até que um critério de parada seja satisfeito, como o número máximo de gerações. Por fim, o melhor indivíduo é selecionado e representará uma CNN válida que será treinada por um número maior de iterações (épocas). O cálculo de aptidão de cada indíviduo é feito a partir de um treinamento com número de épocas reduzido. A seleção utiliza o torneio binário proposto e os filhos são gerados a partir do operador de *crossover* proposto. A nova população

é reinserida a partir do elitismo entre a população anterior e os filhos gerados. Outros aspectos do algoritmos serão apresentados mais detalhadamente das subseções seguintes.

3.1.2 Estratégia de codificação de indivíduos

A codificação do indivíduo proposta é composta por dois vetores que representam as camadas da CNN. O primeiro vetor é composto apenas por camadas do tipo Convolucional e de *Pooling*, o segundo vetor é composto apenas por camadas completamente conectadas e na construção da CNN ele é colocado no fim do primeiro vetor. Considerando que o número de camadas de uma boa CNN é desconhecido para o problema, a codificação de tamanho variável se torna interessante.

Além dos tipos de camadas, outros atributos do gene são os parâmetros estatísticos: média e desvio-padrão. Estes serão utilizados para formar uma distribuição Gaussiana de onde os pesos iniciais serão extraídos.

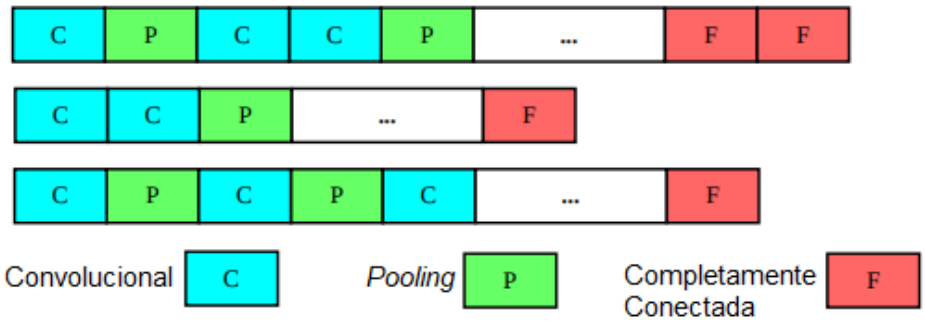

Figura 19 – Ilustração da codificação utilizada no EvoCNN

Fonte: Adaptado de Sun et al. (2020).

3.1.3 Inicialização

Devido às convenções em CNN, no primeiro vetor que representa o gene, a primeira camada sempre é do tipo convolucional e as camadas seguintes serão adicionadas com igual probabilidade, podendo ser dos tipos Convolucional ou de *Pooling*. Os hiperparâmetros de cada uma dessas camadas são definidos de maneira aleatória de acordo com as configurações disponíveis na tabela 1. Na inicialização, um tamanho de camadas é sorteado a partir de limites inferior e superiores pré-determinados. Um processo análogo acontece com o segundo vetor, que contém as camadas completamente conectadas. Os vetores são combinados como um cromossomo para representar o indivíduo. O Algoritmo 2 apresenta o processo.

Tabela 1 – Hiperparâmetros disponíveis para cada tipo de camada.

Tipo de Camada	**Hiperparâmetros disponíveis**
Convolucional	Dimensões do filtro, número de filtros, dimensões dos passos, tipo de convolução, desvio-padrão e média.
Pooling	Dimensões do filtro, número de filtros, dimensões dos passos, tipo de *pooling* (máximo ou média), desvio-padrão e média.
Completamente Conectada	Número de neurônios, desvio-padrão e média.

Fonte: Adaptado de Sun et al. (2020)

Algoritmo 2 Inicialização da População

Entrada: Tamanho da População T_{pop}, tamanho máximo do primeiro vetor N_{cp}, tamanho máximo do segundo vetor N_f.
Saída: População Inicial P_0.
$P_0 \leftarrow \emptyset$;
while $|P_0| \leq T_{pop}$ **do**
  $part_1 \leftarrow \emptyset$;
  $N_{cp} \leftarrow$ Inteiro aleatório entre $[1, N_{cp}]$;
  while $|part_1| \leq N_{cp}$ **do**
    $r \leftarrow$ Gerar um valor aleatório entre $[0, 1]$;
    if $r \leq 0.5$ **then**
      $l \leftarrow$Inicializar camada convolucional com configurações aleatórias;
    else
      $l \leftarrow$Inicializar camada de *pooling* com configurações aleatórias;
    end if
    $part_1 \leftarrow part_1 \cup l$;
  end while
  while $|part_2| \leq N_f$ **do**
    $l \leftarrow$Inicializar camada completamente conectada com configurações aleatórias;
    $part_2 \leftarrow part_2 \cup l$;
  end while
  $P_0 \leftarrow P_0 \cup (part_1 \cup part_2)$;
end while
return P_0;

3.1.4 Cálculo de aptidão

O processo de cálculo de aptidão ocorre em duas etapas, na primeira etapa o cromossomo representando uma CNN será treinado utilizando o conjunto de treinamento, D_{train}, e posteriormente sua média de classificação (acurácia) e desvio-padrão no conjunto de validação, D_{valid}, são utilizados como a aptidão do indivíduo. Além disso, o número de parâmetros (pesos) da CNN é capturado como atributo do indivíduo. Entende-se que uma CNN com menos parâmetros seja mais desejável que uma CNN com um número superior de parâmetros desde que essas possuam desempenho semelhante, visto que o custo computacional para treinamento e utilização é inferior. Devido a natureza do AG, o processo de treinamento ocorre com um número reduzido de épocas, pois seria com-

putacionalmente custoso o treinamento completo, usualmente maior que 100 épocas, de todas as redes avaliadas durante a execução do AG. Essa estratégia funciona, pois de acordo com o autor, no AG é importante capturar a tendência de desempenho, que pode ser descrita através da média de classificação e desvio-padrão.

3.1.5 Torneio Binário Relaxado

No EvoCNN, uma adaptação do torneio binário foi criada. O Algoritmo 3 ilustra o processo de comparação de indíviduos. Dois parâmetros, a diferença entre médias α e a diferença entre o número de parâmetros, β, devem ser definidos como parâmetros do AG. Esses parâmetros atuam como a diferença mínima necessária para considerar um indivíduo superior a outro em relação à aptidão e número de parâmetros.

Durante a comparação dos indíviduos, aquele que possuir média de classificação superior e acima do limite α vencerá o torneio e será selecionado. Caso contrário, o número de parâmetros é utilizado, o indivíduo com um número de parâmetros inferior será selecionado. Persistindo-se o empate, utiliza-se o de menor desvio-padrão e por fim uma seleção aleatória em caso de empate em todas as métricas.

Algoritmo 3 Torneio Binário Relaxado

Entrada: dois indivíduos, o limite de média α, e o limite do número de parâmetros β.
Saída: o indivíduo selecionado.
$s_1 \leftarrow$ indivíduo com a maior média;
$s_2 \leftarrow$ o outro indivíduo;
$\mu_1, \mu_2 \leftarrow$ valores de média de s_1, s_2, respectivamente;
$std_1, std_2 \leftarrow$ desvios-padrão de s_1, s_2, respectivamente;
$c_1, c_2 \leftarrow$ número de parâmetros de s_1, s_2, respectivamente;
if $\mu_1 - \mu_2 > \alpha$ **then**
  return s_1;
else
  if $c_1 - c_2 > \beta$ **then**
    return s_2;
  else
    if $std_1 < std_2$ **then**
      return s_1;
    else
      if $std_1 > std_2$ **then**
        return s_2;
      else
        return escolha aleatória entre s_1, s_2
      end if
    end if
  end if
end if

3.1.6 Cruzamento

No EvoCNN, um operador de cruzamento é proposto a partir de um método denominado Alinhamento de Unidade (UA) para recombinar dois cromossomos de tamanhos distintos. Durante o processo de cruzamento, para cada cromossomo, cada tipo de camada será organizado numa lista, denominada Coleção de Unidades (UC). Uma vez alinhadas, etapa denominada pelo autor como UA, há a troca de material genético das mesmas posições nas UC através do crossover SBX (DEB; AGRAWAL et al., 1995) e a combinação ocorre na fase denominada Restauração de Unidade (UR). No caso em que as UC tenham tamanhos diferentes, as unidades restantes não serão combinadas, mantendo a mesma posição no cromossomo original. A mutação pode ocorrer após o cruzamento, através da mutação polinomial (DEB; DEB, 2014), visto que as configurações de camadas são codificadas a partir de números reais. O processo de cruzamento é ilustrado na Figura 20.

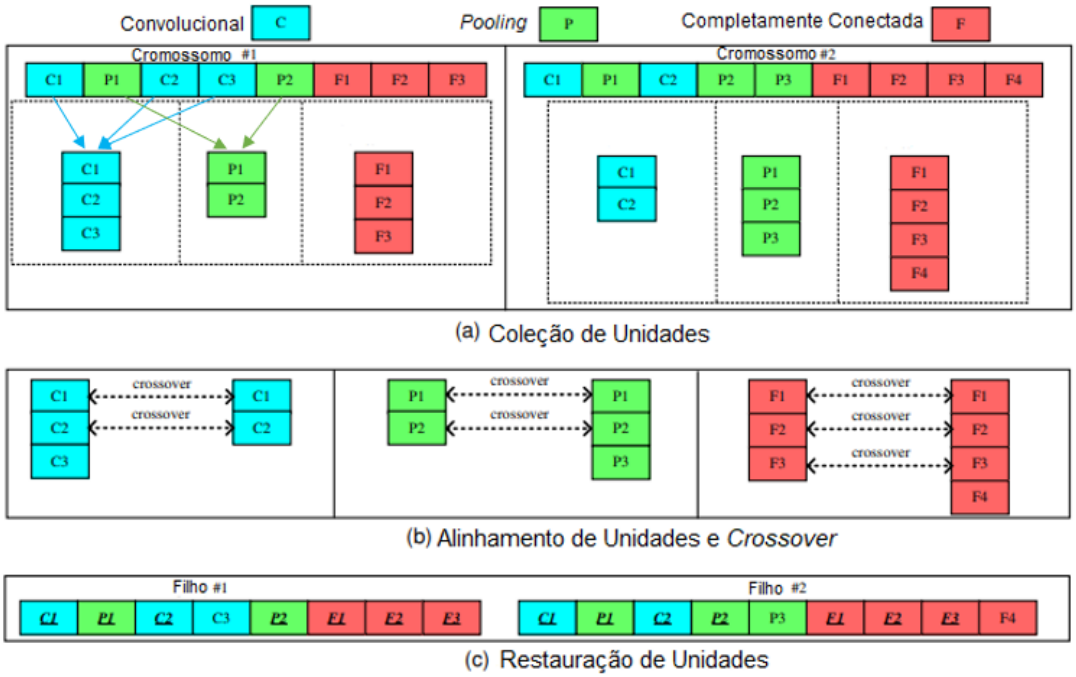

Figura 20 – Ilustração do processo de cruzamento do EvoCNN.

Fonte: Adaptado de Sun et al. (2020).

3.1.7 Reinserção

O processo de reinserção ocorre em duas etapas, balanceando elitismo e diversidade. A princípio, os indivíduos com as melhores médias serão selecionados para a próxima população. Os indivíduos restantes são selecionados a partir do Torneio detalhado na

Subseção 3.1.5, possibilitando uma maior diversidade. É importante ressaltar que os indivíduos selecionados na primeira etapa não participarão do torneio.

3.1.8 Escolha do melhor indivíduo

No fim do AG, a decisão de melhor indivíduo pode depender da aplicação. Pode-se optar pelo indíviduo de melhores médias ou o indivíduo que possa balancear a quantidade de parâmetros com o desempenho. Uma vez escolhido o melhor indivíduo, este será decodificado como uma CNN e passará pelo treinamento com um número maior de épocas.

3.2 AG para seleção de camadas

Neste método, um AG é proposto para a seleção de camadas em AT (NAGAE; KAWAI; NOBUHARA, 2020). Este processo de seleção de camadas é normalmente realizado manualmente. Porém, com o desenvolvimento de novas CNN o número de camadas tem aumentado, tornando o processo cada vez mais difícil. Desta forma, o desenvolvimento de um AG para a otimização dessa seleção torna-se necessário.

3.2.1 Introdução

No AG de seleção de camadas, o cromossomo é representado por um vetor binário, onde o tamanho do vetor é definido pelo número de camadas e cada valor irá indicar quais camadas serão treinadas (atualização de pesos) e quais camadas terão os pesos fixos, ou seja, não serão atualizadas pelo Gradiente Descendente, como ilustrado na Figura 21.

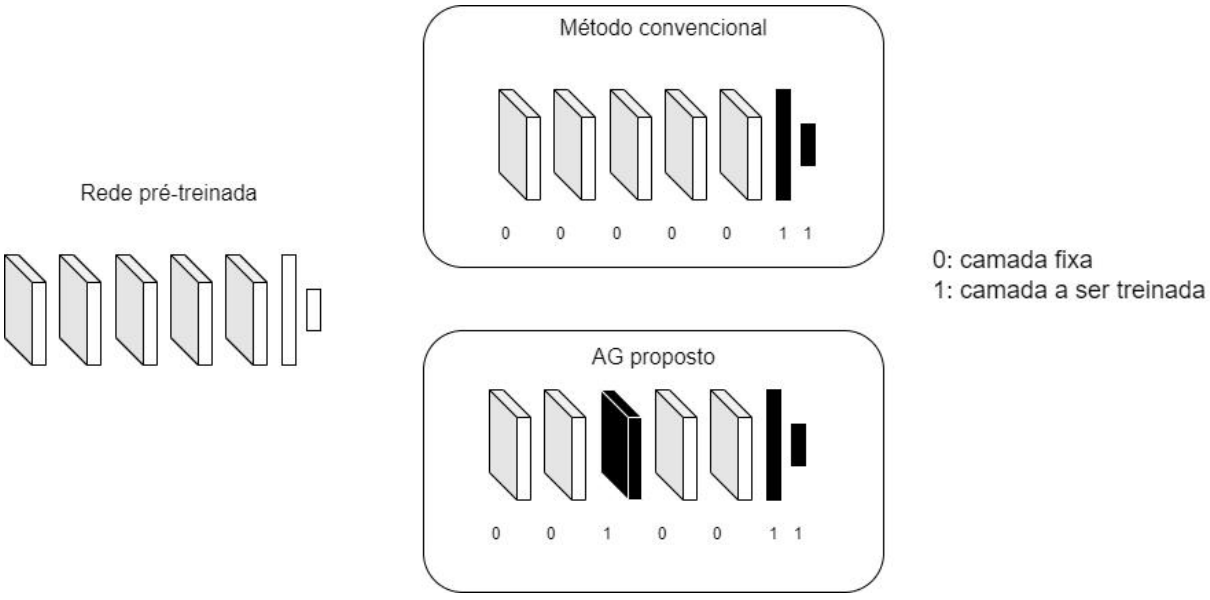

Figura 21 – Visão geral do AG para seleção de camadas.

3.2.2 Inicialização

Os cromossomos são representados como vetores binários, onde 1 indica uma camada treinável e 0 indica uma camada de pesos fixos. Por exemplo, um cromossomo estruturado como [1,0,1,0,0] representa uma CNN de 5 camadas em que apenas as camadas 1 e 3 seriam treináveis. É importante ressaltar que a CNN pode ter um número superior de camadas, porém apenas as camadas treináveis são consideradas na codificação, isto é, as camadas de *pooling* não são representadas.

Além disso, considerando que a última camada é responsável pela classificação final, esta sempre é considerada treinável e não será representada no AG. Neste método, o cromossomo é inicializado com apenas 1 camada treinável, sendo esta uma restrição imposta por Nagae, Kawai e Nobuhara (2020) de modo a obter soluções con um número reduzido de camadas treinadas. Uma visão geral do método pode ser observada na Figura 22.

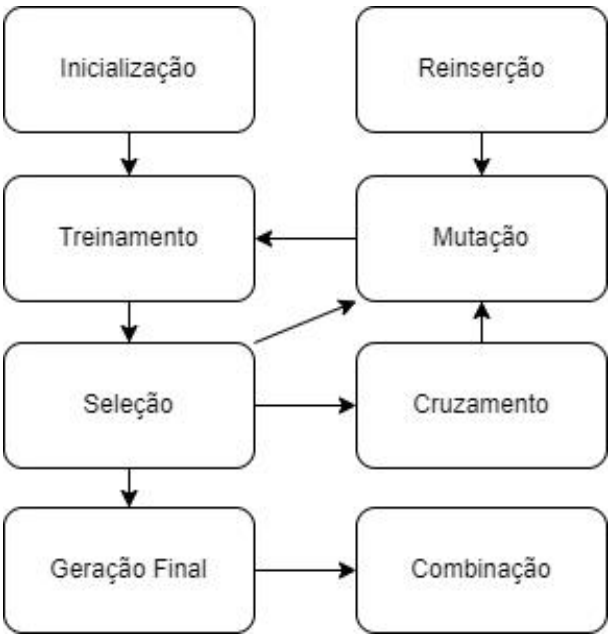

Figura 22 – Rotina principal do AG de seleção de camadas.

3.2.3 Treinamento e cálculo de aptidão

Diferentemente do método descrito na Seção 3.1, neste caso as CNN são treinadas com um número elevado de épocas (maior que 50 épocas) com o gradiente descendente. Isto é possível pois o custo de treinamento da rede com um número elevado de camadas fixas é reduzido em relação a treinar uma CNN do zero como no método anterior.

O cálculo de aptidão é feito a partir da acurácia de classificação, esta métrica é comumente utilizada desde que as classes estejam balanceadas.

3.2.4 Seleção

No processo de seleção, a metade dos cromossomos com melhor aptidão são selecionados como parentais para cruzamento. Neste método o autor explora quatro métodos para seleção: elitismo, roleta, torneio e torneio + elitismo. Estes métodos serão explicados detalhadamente a seguir.

3.2.4.1 Elitismo

No elitismo, um número pré-determinado de indivíduos seguirá para a nova população, os indivíduos são ordenados de acordo com as suas aptidões e os melhores serão selecionados.

3.2.4.2 Roleta

Na seleção por roleta, a probabilidade de um indivíduo ser escolhido é proporcional a sua aptidão. Este processo ainda é elitista, porém ainda permite uma maior diversidade quando comparado ao método elitismo.

Considerando N o número total de indivíduos na população, $f(i)$ a aptidão para o i-ésimo cromossomo, a probabilidade de um cromossomo ser selecionado pode ser expressa pela equação 4.

$$p_i = \frac{f_i}{\sum_{n=1}^{N} f(n)}, onde \tag{4}$$

- p - Probabilidade de seleção;
- f - Aptidão;
- N - Número total de cromossomos.

3.2.4.3 Torneio

Na seleção por torneio do AG para seleção de camadas é utilizado o tamanho de torneio de 2 indivíduos.

3.2.4.4 Elitismo + Roleta

Na seleção com combinação de elitismo e roleta, o elitismo é utilizado para a fase de cruzamento e a roleta é utilizada na etapa de reinserção.

3.2.5 Cruzamento

No cruzamento, os indivíduos filhos são gerados a partir da combinação de dois pais utilizando uma operação lógica OU. Os pais são organizados de forma sequencial numa

lista e o cruzamento ocorre com a combinação do pai na i-ésima posição com o pai na posição $i + 1$. A Figura 23 ilustra o processo de cruzamento.

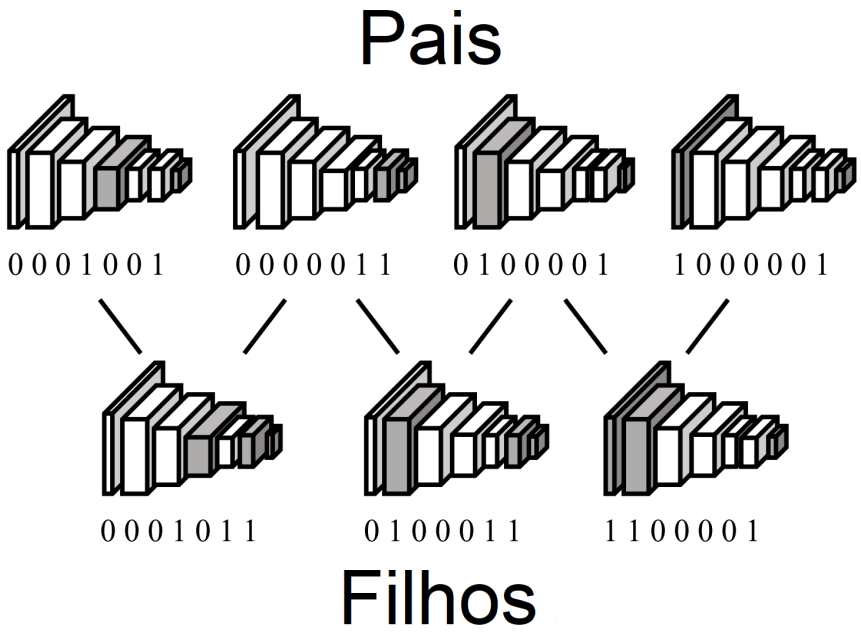

Figura 23 – Processo de cruzamento no AG para seleção de camadas.

Fonte: adaptado de Nagae, Kawai e Nobuhara (2020).

3.2.6 Reinserção

No processo de reinserção, considerando um tamanho de população T_{pop}, a nova população será composta pelos $T_{pop}/2$ indivíduos mais aptos da geração atual, os $T_{pop}/4$ filhos gerados a partir do cruzamento e $T_{pop}/4$ indivíduos inicializados. Esta última composição permite uma maior diversidade, porém pode impactar na convergência do AG.

3.2.7 Mutação

A mutação utilizada é a inversão dos elementos, ou seja, o elemento final será o novo elemento inicial e vice-versa.

3.2.8 Combinação de indivíduos

No fim do processo de evolução, os melhores m indivíduos são utilizados para formar um modelo *ensemble* (VALENTINI; MASULLI, 2002), em que as soluções geradas irão participar de um processo de votação majoritária, e este conjunto é utilizado para fazer a

classificação no conjunto de testes. A técnica *ensemble* é interessante no contexto da evolução de diversos indivíduos no AG, entretanto, deve-se considerar o custo computacional de uma classificação, que pode ser elevado e de pouco valor prático.

Capítulo 4

Proposta

Considerando o impacto que as diferentes configurações de CNN podem ter no desempenho final de um modelo, métodos sistemáticos de otimização apresentam-se como uma alternativa promissora no desenvolvimento de modelos de AM utilizando redes neurais.

Métodos de otimização bio-inspirados tem demonstrado resultados promissores nesse estudo, porém entende-se que seja necessário o aprimoramento constante dessas técnicas.

O algoritmo proposto por Sun et al. (2020) possui uma representação cromossômica robusta, e produz bons resultados na classificação de diversos *datasets*. Entretanto, entende-se que outros aspectos da CNN podem ser explorados no processo de otimização, como por exemplo, o uso de funções de ativação heterogêneas (WANG et al., 2020) e outros aspectos evolutivos como novos operadores de mutação.

O AG proposto por Nagae, Kawai e Nobuhara (2020) apresenta-se como uma técnica inovadora na utilização de processos evolutivos em outros aspectos da criação de modelos de AM, especialmente considerando a AT. Além disso, este AG possui um custo computacional consideravelmente mais reduzido, o que o torna bastante promissor para aplicações do mundo real. Portanto, entende-se que esta técnica pode ser explorada e aprimorada especialmente em relação à dois aspectos do AG: extensão do número de camadas treináveis e o comportamento do AG em *datasets* mais próximos de aplicações reais, visto que os experimentos realizados restringem-se a *datasets benchmarks*, como o MNIST de dígitos escritos à mão (DENG, 2012).

4.1 gaCNN

4.1.1 Introdução

O algoritmo proposto, chamado de gaCNN, é uma extensão da técnica EvoCNN (SUN et al., 2020). As principais contribuições desta proposta são a adição de funções de ativação heterogêneas, Unidade Linear Retificadora (ReLU), sigmoide e Tangente Hiperbólica (Tanh), novos operadores de mutação adicionar bloco (*add*), remover bloco (*remove*),

alteração de bloco (*change*), e a adição de parâmetros *Dropout* (BALDI; SADOWSKI, 2013) e normalização de lotes como parte do processo evolutivo através da adição destes parâmetros nos atributos das camadas. Ademais, uma nova representação de indivíduo é proposta, para representar as funções de ativação.

O gaCNN utiliza a estratégia de codificação de indivíduos descrita na subseção . A população é inicializada aleatoriamente e a aptidão de cada indivíduo é calculada através do treinamento das CNN em uma época devido ao custo elevado para o treinamento completo, que varia normalmente entre 50 e 250 épocas, esta limitação permite o treinamento mais rápido, porém pode não necessariamente ser um indicador final do desempenho preditivo do indivíduo. Esta abordagem é utilizada em trabalhos similares (SUN et al., 2020), (KENNEDY; EBERHART, 1995). O algoritmo apresenta a rotina principal do AG proposto. O método de seleção utilizado é o torneio e o processo de reinserção elitista, compondo a nova população dos melhores pais, filhos gerados e as mutações. Neste caso, as mutações podem ocorrer em todos os indivíduos, não somente nos filhos, adicionando diversidade ao AG.

Após a execução do AG, o melhor indivíduo será treinado completamente com um número maior de épocas e esta será a saída principal.

Algoritmo 4 gaCNN: rotina principal

1: **Entrada:** Tamanho da população T_{pop}, número de gerações $gens$, número de épocas ne, tamanho de lote bs, taxa de aprendizado lr;
2: **Saída:** melhor arquitetura treinada;
3: $g \leftarrow 0$;
4: $pop \leftarrow inicializarPop(T_{pop})$;
5: **while** $g \leq gens$ **do**
6: $\quad calcularAptidao(pop)$;
7: $\quad melhoresPais \leftarrow SelecionaKMelhores(pop, k = 40\%)$;
8: $\quad filhos \leftarrow TourCrossover(bestParents, k = 40\%)$;
9: $\quad mutacoes \leftarrow Mutacao(melhoresPais + filhos, taxaMutacao = 20\%)$;
10: $\quad pop \leftarrow melhoresPais + filhos + mutacoes$;
11: $\quad g \leftarrow g + 1$;
12: **end while**
13: $melhor \leftarrow pop.melhor$;
14: $modelo \leftarrow treinoCompleto(melhor, ne, bs, lr)$;
15: **return** $modelo$;

4.1.2 Estratégia de codificação dos indivíduos

A representação do cromossomo consiste de dois vetores, um vetor representando as camadas e outro vetor representando as funções de ativação. Um dos vetores armazena a sequência de camadas da CNN. Cada camada pode ser de três tipos: Convolucional, *Pooling* ou Completamente Conectada. Outro vetor armazena a sequência de funções de

ativação para todas as camadas, as ativações podem ser ReLU, sigmoide ou Tanh, além de um booleano indicando se a ativação será aplicada naquela camada. Esta representação permite uma flexibilidade para adição de novas camadas, novas funções de ativação e tamanho variável.

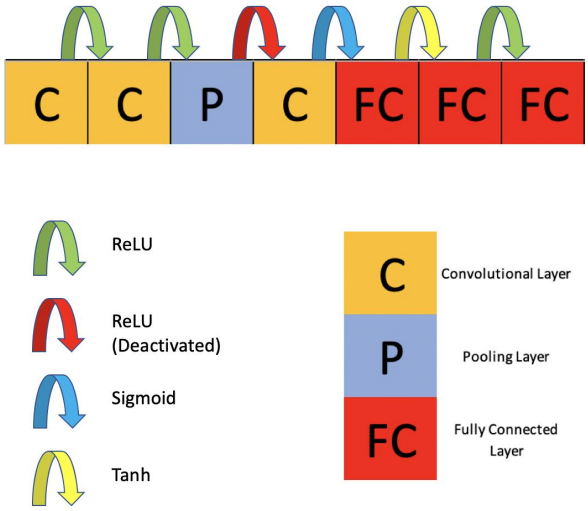

Figura 24 – Codificação de indivíduo proposta em gaCNN.

4.1.2.1 Representação de Camadas

As camadas da CNN além de diferentes tipos, possuem atributos diferentes e podem ser alterados no processo evolutivo do AG. As camadas em gaCNN são representadas da seguinte maneira:

- ❏ Camada Convolucional
  - Formato do passo (*stride*) (altura, largura): (1,1);
  - Quantidade de Mapas na Saída: 1 a 256;
  - Formato do Filtro: (1,1), (3,3) or (5,5);
  - Normalização de Lotes: Verdadeiro ou Falso;
  - *Dropout*: 0,0; 0,1; 0,2; 0,3; 0,4 ou 0,5.
- ❏ Camada de *Pooling*
  - Tipo de *Pooling*: Média ou Máximo;
  - Formato do passo (*stride*): (1,1), (2,2) or (3,3);
  - Formato do Filtro: (2,2), (3,3) or (5,5);
  - Normalização de Lotes: Verdadeiro ou Falso;

- *Dropout*: 0,0; 0,1; 0,2; 0,3; 0,4 ou 0,5.

❏ Camada Completamente Conectada

- Número de neurônios: 128, 256 ou 512;
- Normalização de Lotes: Verdadeiro ou Falso;
- *Dropout*: 0,0; 0,1; 0,2; 0,3; 0,4 ou 0,5.

4.1.2.2 Representação de Funções de Ativação

As funções de ativação em gaCNN são representadadas a seguir:

❏ Ativação

- Tipo: ReLU, Sigmoide ou Tanh;
- Ativa: Verdadeiro ou Falso.

4.1.3 Inicialização

A inicialização da população consiste na geração de arquiteturas aleatórias válidas, através do processo descrito no Algoritmo 5. Neste processo, por convenção, a primeira camada sempre é do tipo convolucional com parâmetros aleatórios de acordo com as configurações apresentadas em 4.1.2.1. Este tipo de camada pode ser seguida de qualquer outro tipo de camada. Entretanto, uma vez que uma camada completamente conectada é selecionada, esta só poderá ser seguida deste mesmo tipo. Por razões computacionais, o número de camadas completamente conectadas é limitado ao máximo de 3.

4.1.4 Seleção e Cruzamento

A operação de seleção utiliza o torneio com tamanho 2, os indivíduos que participam do torneio são os 40% mais aptos. O processo de cruzamento é similar ao apresentado no Capítulo 3, proposto para o algoritmo evoCNN (SUN et al., 2020), com a adição das funções de ativação que são mantidas de acordo com os cromossomos parentais.

4.1.5 Mutação

Neste trabalho é proposto um processo de mutação com até 4 modificadores de cromossomos. Os modificadores podem alterar a estrutura do cromossomo através de adição, remoção e edição de camadas ou a adição de blocos. Estes modificadores são: *Add*, *Remove*, *Change*, e *Add Block*. O modificador *Add* adiciona uma camada aleatória numa posição aleatória do cromossomo. De maneira análoga, o modificador *Remove* irá remover uma camada do cromossomo e o modificador *Change* irá modificar as configurações

Algoritmo 5 gaCNN: inicialização de indivíduo

Entrada: tamanho mínimo de camadas L_{min}, tamanho máximo de camadas L_{max}, número máxido de camadas completamente conectadas FC_{max}
Saída: Indivíduo inicializado;
$numCamadas \leftarrow rand(L_{min}, L_{max})$;
$i \leftarrow 0$;
$camadaAnterior \leftarrow inicial$;
while $i \leq numCamadas$ **do**
  if $camadaAnterior == inicial$ **then**
    addCamadaConvolucional();
    addActivacaoAleatoria();
    $camadaAnterior \leftarrow Convolucional$
  else
    if $camadaAnterior \neq CompletamenteConectada$ **then**
      addCamadaAleatoria(Convolucional, Pooling, completamenteConectada)
      $camadaAnterior \leftarrow ultimaCamadaAdicionada$;
      if $camadaAnterior == CompletamenteConectada$ **then**
        $camadasCC = camadasCC + 1$;
      end if
    else
      if $camadaAnterior == CompletamenteConectada$ **then**
        addCamadaCompletamenteConectada()
        $camadaAnterior \leftarrow CompletamenteConectada$;
        $camadasCC \leftarrow camadasCC + 1$;
        if $camadasCC \leq FC_{max}$ **then**
          addActivacaoAleatoria();
        else
          $numCamadas \leftarrow i$;
          break;
        end if
      end if
    end if
  end if
  $i \leftarrow i + 1$;
end while

de uma camada no cromossomo. O modificador *Add Block* adiciona um bloco com as seguintes camadas: Convolucional e *Pooling* seguido de uma ativação ReLU. A mutação pode ocorrer após a geração dos filhos e também nos indivíduos pais.

4.1.6 Cálculo de aptidão

Assim como os trabalhos apresentados no Capítulo 3, a aptidão é calculada a partir da acurácia de classificação no conjunto de validação. Além disso, as CNN são treinadas por um número reduzido de épocas de modo a avaliar o potencial de cada arquitetura. Por fim, o melhor indivíduo é treinado por um número de épocas entre 50 e 250 para geração de um modelo de aprendizagem de máquina otimizado a depender do algoritmo de otimização e ao tamanho do conjunto de dados de interesse.

4.2 MLTLGA

O algoritmo proposto, MLTLGA, é baseado na proposta de Nagae, Kawai e Nobuhara (2020) com alterações no processo evolutivo e na inicialização das camadas.

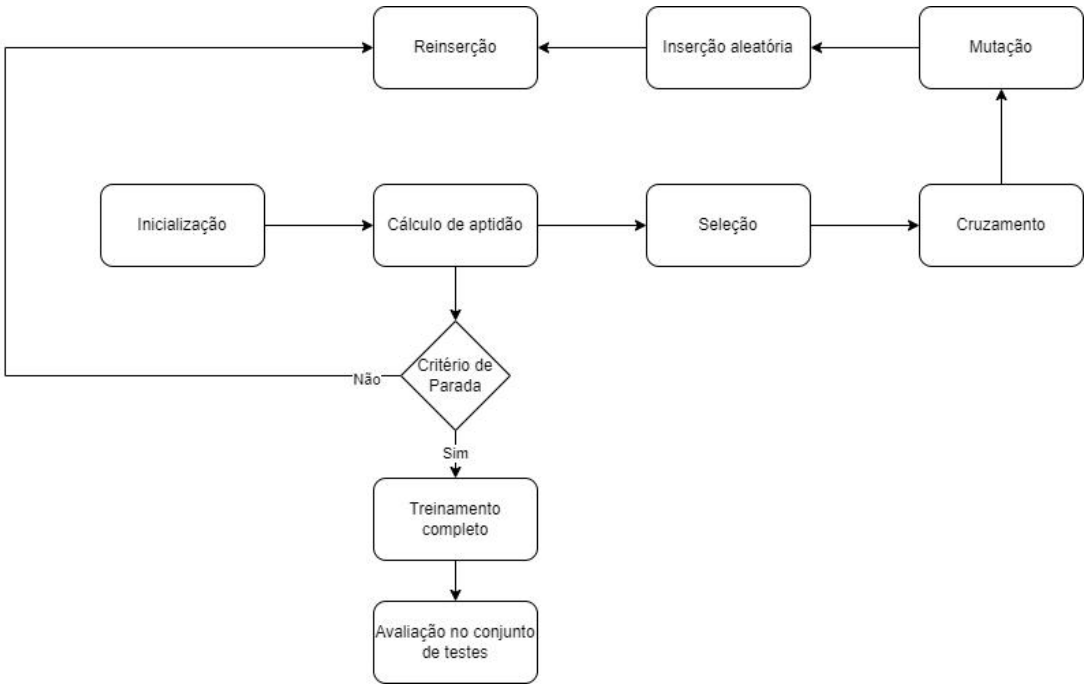

Figura 25 – Fluxo principal do MLTLGA.

4.2.1 Inicialização

Os cromossomos são representados por um vetor binário, onde 1 indica uma camada treinável e 0 indica uma camada de pesos fixos. As camadas completamente conectadas não são representadas no vetor. A principal diferença dessa abordagem de inicialização é

a remoção da limitação do número de camadas treináveis. No algoritmo apresentado no capítulo anterior, na Seção 3.2, este número é limitado a uma camada treinável.

4.2.2 Cálculo de aptidão

A acurácia no conjunto de validação é utilizada como aptidão no AG. Todos os parâmetros do modelo são herdados do modelo pré-treinado, exceto a última camada completamente conectada, que precisa necessariamente ser alterada para a classificação no novo conjunto de dados. Durante o processo de treinamento dos modelos gerados a partir dos cromossomos, somente os pesos das camadas indicadas pelo vetor como 1 serão atualizados. Cada CNN é treinada por 5 épocas, número obtido empiricamente, com um bom balanço entre custo computacional e indicação de potencial de desempenho final da CNN.

4.2.3 Seleção

No processo de seleção dois métodos foram avaliados: torneio e roleta.

4.2.4 Cruzamento

Devido a simplicidade de codificação de indivíduo, foi utilizado o método de cruzamento de dois pontos.

4.2.5 Mutação

A mutação ocorre nos indivíduos filhos com um probabilidade P_{mut}. A mutação utiliza o processo de inversão.

4.2.6 Reinserção aleatória

Com o objetivo de aumentar a diversidade, em cada geração, indíviduos que não fazem parte do processo evolucionário são inicializados e adicionados à população.

4.2.7 Reinserção

O método elitista foi utilizado, com a adição da reinserção aleatória, a diversidade é adicionada a cada geração. O processo de reinserção ordena os indivíduos dos seguintes grupos: população anterior, filhos e indivíduos aleatórios.

Capítulo 5

Desenho Experimental

Neste capítulo serão apresentados os experimentos realizados nesta pesquisa. Os experimentos são divididos em duas grandes partes, uma para cada algoritmo proposto no Capítulo 4. Em ambos os casos, a mesma métrica de avaliação foi utilizada.

5.1 Métrica de Avaliação

A métrica utilizada para avaliação dos dois métodos é a acurácia de classificação. Esta métrica é comumente utilizada em problemas de classificação binários ou com classes balanceadas. O cálculo é descrito através da equação 5.

$$acurácia = \frac{tp + tn}{tp + tn + fn + fp} \tag{5}$$

Onde os resultados da classificação são representados a partir de: tp (*true positive*), tn (*true negative*), fp (*false positive*) e fn (*false negative*).

5.2 gaCNN

Nesta Seção, o desenho experimental para o método gaCNN é apresentado.

5.2.1 Conjunto de Dados

No desenho experimental do gaCNN, dois conjuntos de dados considerados de referência foram utilizados: MNIST (DENG, 2012) e *Fashion* MNIST (XIAO; RASUL; VOLLGRAF, 2017a). Estes conjuntos de dados são interessantes pois são amplamente utilizados na literatura e podem ser utilizados para comparar o método proposto neste trabalho com resultados divulgados.

5.2.1.1 MNIST

Este conjunto de dados consiste de 60 mil imagens de treinamento e 10 mil imagens para teste. As imagens são compostas por digitos escritos à mão entre 0 e 9 e as classes são balanceadas. Este conjunto é amplamente utilizado pela comunidade de AM e é considerado relativamente fácil para CNN modernas, atingindo acima de 99% de acurácia (STOJNIC, 2020). As imagens são em escala de cinza com dimensões 28x28 e pode ser utilizado gratuitamente via *download* [1].

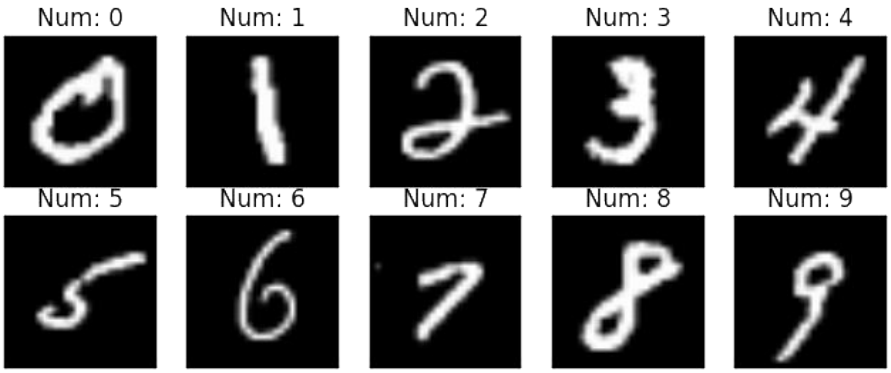

Figura 26 – Imagens extraídas do conjunto MNIST.

5.2.1.2 *Fashion* MNIST

Este conjunto de dados consiste de 60 mil imagens de treinamento e 10 mil imagens para teste. As imagens são compostas por 10 classes de vestuário (*T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, Ankle boot*). As imagens possuem dimensão 28x28 em escala de cinza. É uma alternativa mais desafiadora em relação ao MNIST convencional e pode ser utilizado gratuitamente via *download* [2].

[1] http://yann.lecun.com/exdb/mnist/
[2] https://github.com/zalandoresearch/fashion-mnist

Figura 27 – Imagens extraídas do conjunto *Fashion* MNIST.

5.2.2 Configuração de Experimentos

A configuração dos experimentos para o gaCNN é descrita na Tabela 2. Devido a limitações computacionais e de custo, o AG foi executado 30 vezes para avaliação do conjunto de dados *Fashion* MNIST, que é mais desafiador para CNN que o conjunto MNIST de dígitos escritos à mão. Após a execução dos AG, as melhores arquiteturas são treinadas completamente e os resultados discutidos posteriormente.

As métricas para os modelos são a acurácia de classificação e a acurácia em cada classe, permitindo a avaliação indireta da precisão das CNN otimizadas. Em relação à comparação com métodos bio-inspirados e não bio-inspirados, foram extraídos os resultados publicados pelos autores, considerando que a reimplementação dos outros métodos tornaria a pesquisa impraticável devido aos elevados custos computacionais.

O método foi implementado usando Python v3.6, *framework* PyTorch v1.4 (PASZKE et al., 2019) e CUDA 10 aceleradas via GPU NVIDIA Tesla K80 em arquitetura de nuvem.

Tabela 2 – Configuração dos experimentos para gaCNN.

Parâmetros	**Fashion MNIST (AG)**	**Fashion MNIST (Treinamento)**	**MNIST (AG)**	**MNIST (Treinamento)**
Execuções	30	-	1 -	-
Gerações	100	-	40	-
Tamanho de população	100	-	50	-
Taxa de mutação	20%	-	20%	-
Taxa de cruzamento	40%	-	40%	-
Seleção (Torneio)	40%	-	40%	-
Número de épocas	1	250	1	50
Tamanho de lote	24	16	24	16
Taxa de aprendizado	0,001	0,001	0,001	0,001

5.3 MLTLGA

Nesta seção, o desenho experimental para o método MLTLGA é apresentado.

5.3.1 Conjunto de Dados

O conjunto de dados utilizado para os experimentos com o método MLTLGA foi o de imagens torácicas de raio-x com anotações de duas classes: *Pneumonia* e *Normal* (KERMANY et al., 2018). Originalmente, este conjunto possui 3.875 e 1.341 imagens, para as classes *Pneumonia* e *Normal*, respectivamente.

O conjunto foi reduzido a 467 imagens, sendo 239 da classe *Pneumonia* e 228 da classe *Normal*. Ao reduzir o número de imagens no conjunto de dados, é possível reduzir o tempo de computação dos AG além de aproximar-se de uma situação de AT.

O conjunto de validação possui 149 imagens, sendo 75 da classe *Pneumonia* e 74 da classe *Normal*. De modo a se manter uma base de comparação justa com outros métodos, o conjunto de teste se manteve como o original, com 624 imagens, sendo 390 imagens da classe *Pneumonia* e 234 da classe *Normal*.

Uma restrição em aprendizagem de transferência é o redimensionamento da entrada, de modo que as dimensões da entrada têm que ser as mesmas do conjunto de dados utilizado para treinar a CNN inicial. A CNN pré-treinada InceptionV3 (SZEGEDY et al., 2016) espera dimensões de 224 x 224 x 3 e os valores normalizados, dessa forma foi necessário aplicar o redimensionamento no conjunto de dados de raio-x.

5.3.2 Configuração de Experimentos

Os parâmetros dos experimentos com MLTLGA podem ser observados na Tabela 3. Como referência, o método proposto por Nagae, Kawai e Nobuhara (2020) foi reimplementado com todas as variações propostas. Os mesmos parâmetros foram utilizados em ambos os AG, exceto o tamanho do torneio.

Além disso, comparou-se o método proposto com as técnicas convencionais de AT, como o treinamento apenas da última camada (*Last Layer*) e com métodos de treinamento a partir de pesos inicializados utilizando uma distribuição normal (*from Scratch*), ou seja, sem AT.

Para os AG (MLTLGA e NAGAE), 4 execuções foram feitas para cada uma das possíveis variações. Os melhores indivíduos encontrados foram treinados completamente por 50 épocas e o seu desempenho e capturado a partir do conjunto de testes. Note que o conjunto de testes não faz parte do processo evolucionário.

As variações possuem nomenclatura na língua inglesa e são, para o MLTLGA: *Tournament*, *Roulette*, para torneio e roleta, respectivamente. Para o método proposto por Nagae, Kawai e Nobuhara (2020): *elite, roulette, tournament* e *elite + roulette*, para as variações elitismo, roleta, torneio e elitismo com roleta, respectivamente.

As CNN finais são treinadas com uma taxa de aprendizagem de 0,001 com taxa de queda de 0,1 a cada 7 épocas, o otimizador é o gradiente descendente estocástico com momento de 0,9 e tamanho de lote de 16.

Tabela 3 – Configuração dos experimentos para MLTLGA.

Parâmetro	**MLTLGA**	**Nagae et. al**
Tamanho de População	20	20
Gerações	5	5
Taxa de cruzamento	50%	50%
Taxa de mutação	1%	1%
Tamanho do torneio	3	2

5.4 Avaliação dos Resultados

Nesta seção, os resultados são discutidos. Esta seção está dividida em duas partes, uma para cada algoritmo proposto.

5.4.1 gaCNN

Conforme apresentado na Seção 5.2, os resultados serão apresentados para cada conjunto de dados, inicialmente para o *Fashion* MNIST e posteriormente para o MNIST.

5.4.1.1 Resultados para *Fashion* MNIST

Uma investigação inicial do gaCNN sobre a sua capacidade de evolução ao longo das gerações pode ser observada na Figura 28. Este gráfico indica um comportamento de acordo com o esperado em AG e indica que os operadores propostos estão de fato contribuindo para um bom desempenho evolucionário do AG. É possível observar que as 10 gerações iniciais apresentam o maior crescimento em acurácia, demonstrando que

o AG é capaz de descartar arquiteturas de desempenho baixo, porém é possível observar que os ganhos a partir da gerações seguintes são pequenos, isto pode ser explicado pela dificuldade de se encontrar arquiteturas muito superiores com as limitações impostas, ou pela falta de diversidade.

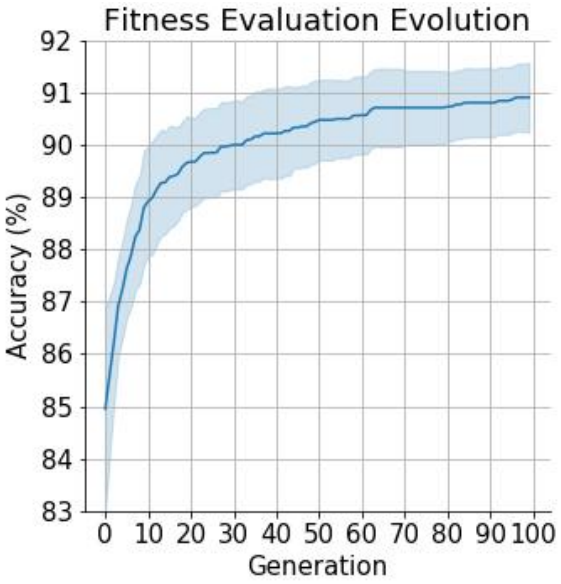

Figura 28 – Média de acurácia de classificação, trecho sombreado indica o desvio-padrão, ao longo das gerações para todas as execuções.

As 30 melhores arquiteturas encontradas para cada execução foram completamente treinadas e os resultados são apresentados na Figura 29. A melhor CNN obtida apresentou desempenho de 93,75% de acurárica e a pior CNN apresentou 92,27% de acurácia. Não foram observados *outliers* durante a execução do experimento, mostrando que o método gaCNN é estável e possui média de acurácia de 93,13% e limites de desempenho entre 92,97% e 93,30% com intervalo de confiança de 95%.

Os resultados das melhores arquiteturas encontrados em cada classe podem ser observados na Tabela 4. As CNN encontradas apresentam melhor desempenho nas classes *Trouser*, *Sandal*, *Sneaker*, *Bag*, e *Ankle Boot*. Este resultado é esperado, visto que estas classes apresentam características que as diferenciam mais facilmente que as classes *Pullover*, *T-Shirt/top*, *Coat* e *Shirt*. Estas apresentam características semelhantes entre si, o que torna esse conjunto mais desafiador para a CNN. Este resultado é promissor, visto que as arquiteturas encontradas não favorecem ou penalizam classes específicas, demonstrando um bom equilíbrio, ou precisão de classificação neste conjunto de dados.

Os resultados do método gaCNN comparado a métodos bio-inspirados e não bio-inspirados são apresentados na Tabela 5. Nesta Tabela, o (+) indica que o método gaCNN obteve melhor desempenho, o (-) indica resultados superiores ao método gaCNN. É possível observar que gaCNN possui melhor desempenho que 9 dos 13 métodos. O método evoCNN (SUN et al., 2020) utilizou um número mais elevado de camadas, que

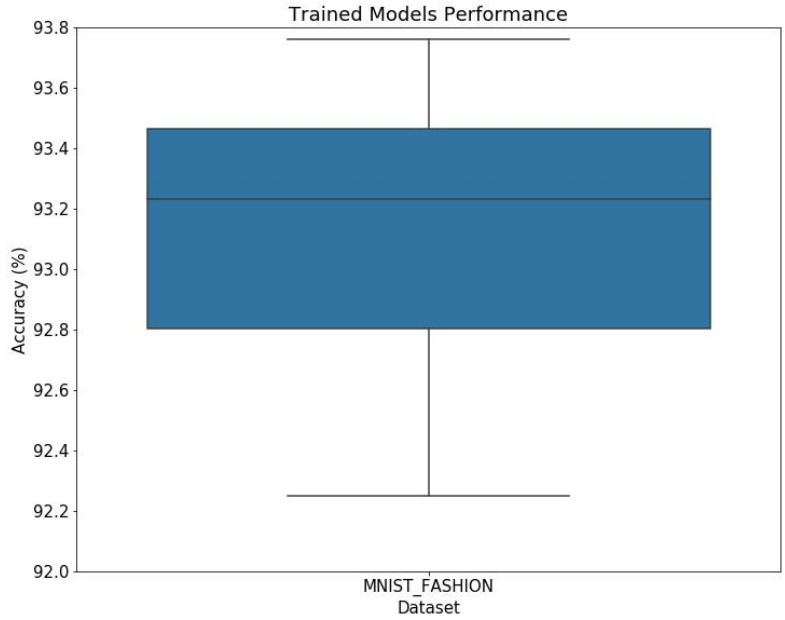

Figura 29 – Box plot dos resultados das CNN treinadas completamente.

Tabela 4 – Acurácia do gaCNN por classe no conjunto de dados Fashion MNIST, a classe *Shirt* apresentou a menor acurácia de classificação.

Class	Acurácia
T-shirt/top	86%
Trouser	98%
Pullover	89%
Dress	93%
Coat	91%
Sandal	97%
Shirt	79%
Sneaker	98%
Bag	98%
Ankle Boot	96%

pode ter impactado no desempenho superior no melhor caso. Entretanto, os resultados das 30 execuções demonstram uma boa estabilidade do gaCNN, o que pode indicar uma maior chance de se obter uma boa arquitetura numa única execução. Além disso, o custo de uma execução no gaCNN foi de 18 horas de GPU, o que é um resultado promissor visto que os outros métodos podem levar dias e até anos para executar (WISTUBA; RAWAT; PEDAPATI, 2019).

Tabela 5 – Método gaCNN versus métodos bio-inspirados e não bio-inspirados no conjunto de testes.

Modelo	**Acurácia**
Desempenho humano (XIAO; RASUL; VOLLGRAF, 2017b)	83,5% (+)
MLP 256-128-100 (XIAO; RASUL; VOLLGRAF, 2017b)	88,33% (+)
GRU + SVM (XIAO; RASUL; VOLLGRAF, 2017b)	88,8% (+)
HOG + SVM (XIAO; RASUL; VOLLGRAF, 2017b)	92,6% (+)
AlexNet (XIAO; RASUL; VOLLGRAF, 2017b)	89,9% (+)
3CONV + 3FC (XIAO; RASUL; VOLLGRAF, 2017b)	93,4% (+)
VGG16 (XIAO; RASUL; VOLLGRAF, 2017b)	93,5% (+)
GoogLeNet (XIAO; RASUL; VOLLGRAF, 2017b)	93,7% (+)
evoCNN (melhor) (SUN et al., 2020)	94,53% (-)
evoCNN (média) (SUN et al., 2020)	92,72% (+)
MobileNet (XIAO; RASUL; VOLLGRAF, 2017b)	95% (-)
psoCNN (melhor) (JUNIOR; YEN, 2019)	94,47% (-)
psoCNN (média) (JUNIOR; YEN, 2019)	94,1% (-)
gaCNN (melhor)	**93,76**%
gaCNN (média)	**93,13**%

5.4.1.2 MNIST

De acordo com os resultados obtidos na Seção 5.4.1.1, é possível demonstrar que o método gaCNN possui resultados estáveis sem *outliers*. Devido a restrições computacionais, e o fato de que este conjunto de dados é relativamente fácil, este experimento irá demonstrar o desempenho do método gaCNN em apenas uma execução, por questões de custo computacional e do experimento.

O gráfico de evolução ao longo de gerações no conjunto de dados MNIST é mostrado na Figura 30. Como mencionado anteriormente, o conjunto MNIST é considerado fácil para CNN modernas, atingindo desempenho acima de 99% até para arquiteturas mais simples, porém, devido a esta característica, os resultados são mais suscetíveis a diferenças de inicialização de parâmetros do que a arquitetura em si. O desempenho do método gaCNN após o treinamento completo do melhor indivíduo pode ser observado na Tabela 6. Os resultados comparativos com métodos competidores são apresentados na Tabela 7. O método gaCNN é melhor que 12 dos 16 métodos neste conjunto de dados, embora mais execuções sejam necessárias para apresentar um intervalo de confiança do método proposto, entretanto, pode-se observar que o método gaCNN apresentou melhores resultados que o método evoCNN (SUN et al., 2020).

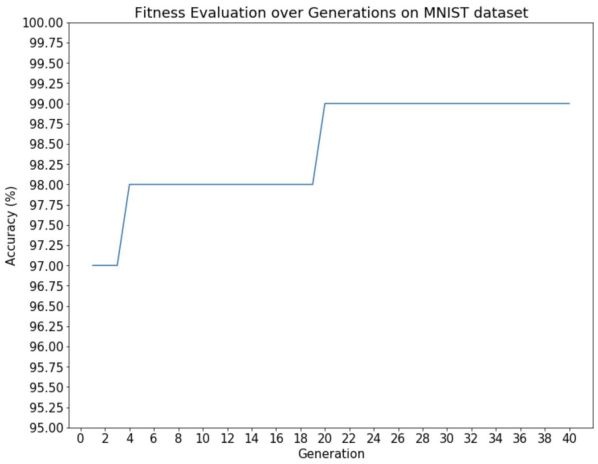

Figura 30 – Aptidão ao longo das gerações para o método gaCNN no conjunto de dados MNIST.

Tabela 6 – Acurácia por classe para o conjunto de dados MNIST, as classes com pior desempenho são 3 e 5.

Classe	**Acurácia**
0	99,39%
1	99,82%
2	99,05%
3	98,76%
4	99,19%
5	98,55%
6	99,39%
7	99,38%
8	99,19%
9	99,61%
Geral	99,14%

5.4.2 MLTLGA

Conforme apresentado na Seção 5.3, os resultados serão apresentados para o conjunto de dados de imagens de raio-x. Inicialmente, são comparados os dois AG, seguidos da comparação com os métodos convencionais de AT e seguido pela comparação com o método de treinamento sem AT. Além disso, um breve estudo das camadas selecionadas é apresentado.

5.4.2.1 Comparação entre AG

Os resultados obtidos durante os experimentos podem ser observados nas Tabela 8. O MLTLGA apresentou acurária de validação próxima a 97%. Este número indica que durante o treinamento, os métodos de seleção por torneio e *role* não apresentaram resultados diferentes na execução do AG. Porém, quando comparado com o método NAGAE e todas as suas variações, os números apresentados pelo MLTLGA são pelo menos 2%

Tabela 7 – Método gaCNN versus métodos bio-inspirados e não bio-inspirados no conjunto de testes MNIST.

Modelo	**Acurácia**
CAE-1 (RIFAI et al., 2011)	97,17% (+)
CAE-2 (RIFAI et al., 2011)	97,52% (+)
PCANet-2 (CHAN et al., 2015)	98,94% (+)
RandNet-2 (CHAN et al., 2015)	98,63% (+)
LDANet-2 (CHAN et al., 2015)	98,60% (+)
IPPSO(melhor) (WANG et al., 2018)	98,87% (+)
IPPSO(média) (WANG et al., 2018)	98,79% (+)
LeNet-1 (LECUN et al., 1998)	98,3% (+)
LeNet-4 (LECUN et al., 1998)	98,9% (+)
LeNet-5 (LECUN et al., 1998)	99,05% (+)
RCNN (LIANG; HU, 2015)	99,69% (-)
DropConnect (LEE et al., 2015)	99,79% (-)
evoCNN(melhor)(SUN et al., 2020)	98,82% (+)
evoCNN(média)(SUN et al., 2020)	98,82% (+)
psoCNN(melhor)(JUNIOR; YEN, 2019)	99,68% (-)
psoCNN(média)(JUNIOR; YEN, 2019)	99,56% (-)
gaCNN	99.14%

superiores. Um *box plot* é apresentado na Figura 31 para o conjunto de validação. Ainda que o conjunto de validação não seja a métrica final, esta permite indicar como o processo evolucionário responde às diferentes configurações propostas em ambos os métodos. Porém, a hipótese para o desempenho superior do método MLTLGA é de que a inicialização de múltiplas camadas permite uma vantagem nas soluções iniciais em relação ao método NAGAE.

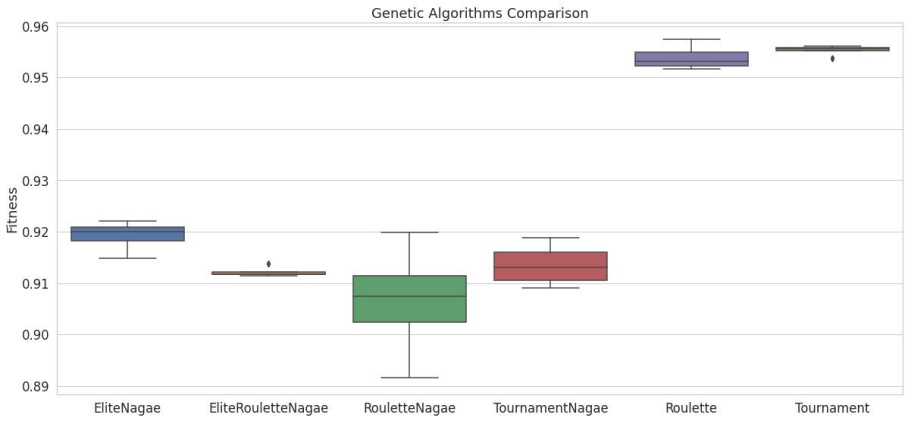

Figura 31 – MLTLGA vs NAGAE no conjunto de validação.

Para a acurácia no conjunto de testes, comparado à validação, o desempenho foi reduzido em pouco mais de 10% em ambos os AG. Entretanto, a acurácia média obtida pelo MLTLGA, utilizando Roleta, foi ao menor 2% superior ao método NAGAE e suas variações. Esta queda de desempenho em relação ao conjunto de testes pode ser expli-

cada pelo tamanho do conjunto de testes ser muito superior em relação ao conjunto de validação.

Os números apresentados na Tabela 8, são as médias de desempenho dos modelos treinados a partir do melhor indivíduo em cada execução por 50 épocas. Considerando a natureza não determinística do gradiente descendente estocástico, cada indivíduo foi treinado 5 vezes. Na Figura 32, os *box plots* apresentam a distribuição para cada variação experimentada neste trabalho.

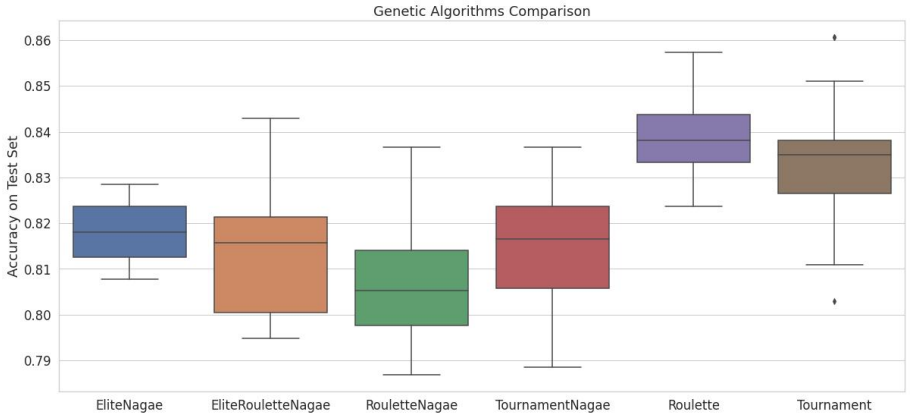

Figura 32 – MLTLGA vs NAGAE no conjunto de testes.

Entretanto, em relação à exploração do espaço de buscas, os resultados na Figura 33 mostram que o método NAGAE explora mais soluções únicas ao longo da execução do AG. Considerando que cada cromossomo é inicializado com apenas uma camada treinável, o número de possibilidade é maior. Além disso, o método de seleção utilizado em NAGAE para reinserção permite uma maior diversidade. No método MLTLGA, a reinserção ordenada ou elitista foi utilizada.

Na Figura 34 é possível observar que a estratégia de inicialização proposta em MLTLGA de fato corrobora com a hipótese de que a inicialização do MLTLGA é mais assertiva do que o método de apenas uma camada proposto por NAGAE.

Tabela 8 – Resultados detalhados das médias de classificação no conjunto de testes após o treinamento completo dos melhores indivíduos.

	NAGAE				MLTLGA	
Execução	Roulette	EliteRoulette	Tournament	Elite	Roulette	Tournament
1	0,81	0,82	0,82	0,81	0,84	0,83
2	0,79	0,82	0,82	0,82	0,84	0,84
3	0,83	0,80	0,80	0,82	0,84	0,84
4	0,80	0,83	0,82	0,82	0,84	0,82
Média	0,81	0,81	0,81	0,82	0,84	0,83
Desvio-padrão	0,01	0,01	0,01	0,00	0,00	0,01

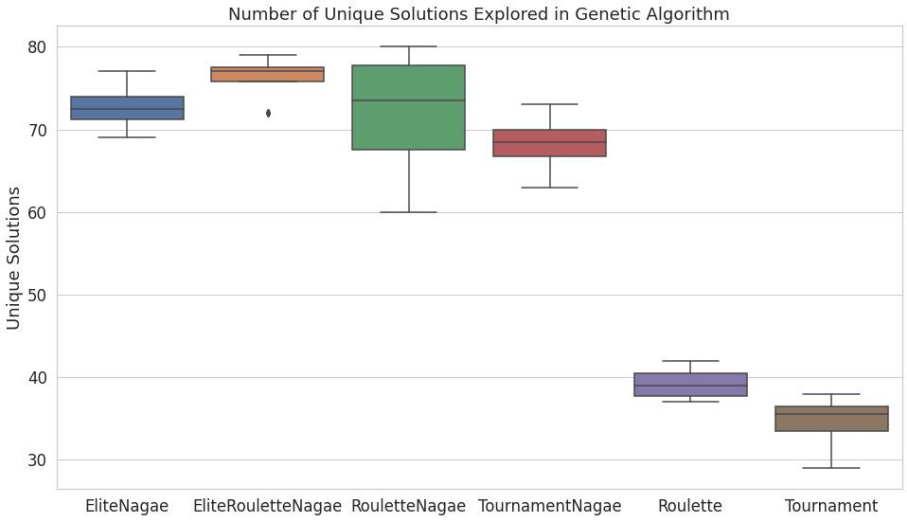

Figura 33 – Número de soluções únicas exploradas.

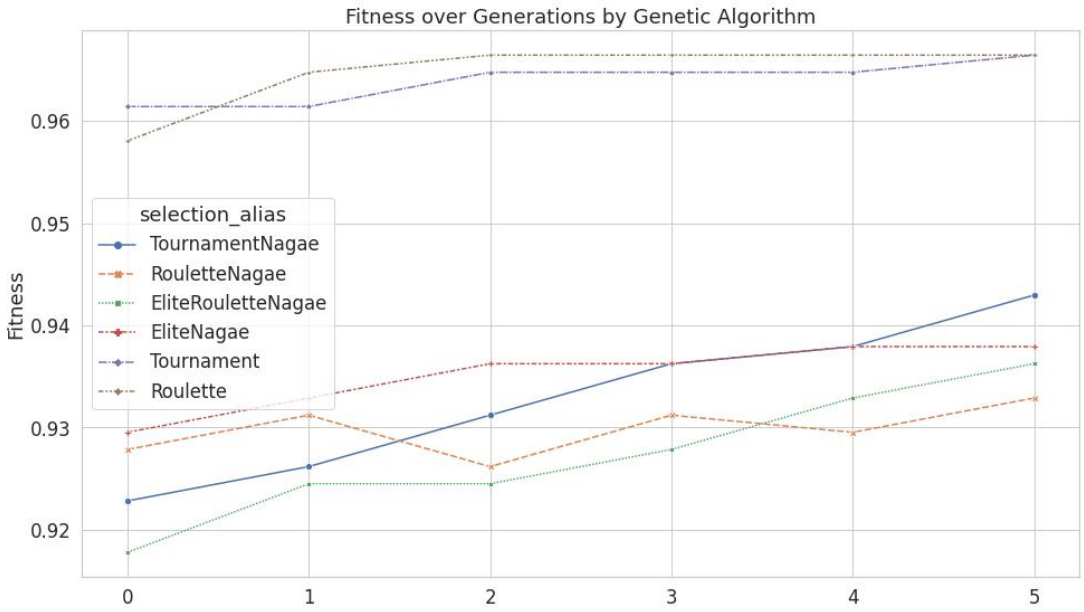

Figura 34 – Média de aptidão (conjunto de validação) ao longo das gerações.

5.4.2.2 Comparação entre MLTLGA e métodos convencionais

O método MLTLGA foi capaz de obter um desemepenho superior aos métodos convencionais de AT, como apresentado na Figura 35. O MLTLGA é, em média, cerca de 2% superior que treinar a CNN a partir do zero e cerca de 4% superior ao método convencional de treinamento apenas da última camada. Ambos os métodos de seleção (Torneio e Roleta) apresentaram melhores resultados que os métodos convencionais. O resultado é esperado, visto que ao adicionar a evolução espera-se que mais camadas ajudem no melhor desempenho do modelo. Outro resultado importante é a demonstração de que para um conjunto de dados de treinamento pequeno, o modelo gerado a partir do treinamento do zero, com todas as camadas, não foi superior ao método evolutivo. Entende-se que caso

o conjunto de dados fosse maior, este método seria o de melhor desempenho, porém a redução do tamanho do conjunto de treinamento permitiu um processo de treinamento mais rápido.

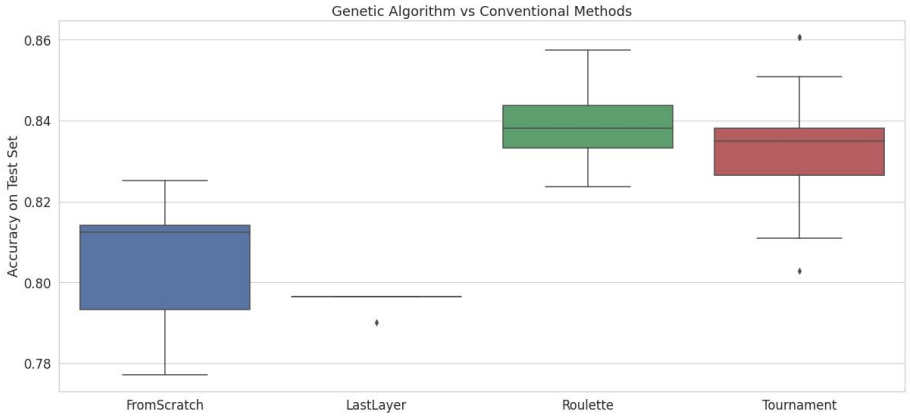

Figura 35 – MLTLGA comparado aos métodos convencionais de treinamento do zero e AT.

5.4.2.3 Seleção de Camadas

As camadas selecionadas durante a execução do método MLTLGA para ambos os métodos de seleção podem ser observados nas Figuras 36 e 37. Em geral, apenas uma camada não foi selecionada pelo menos uma vez (camada 81). A seleção nas camadas iniciais foi bastante comum, o que é contra-intuitivo considerando as características discutidas anteriormente das CNN. Esta observação também ocorreu para o método NAGAE. Considerando, como as camadas treináveis estão espalhadas, seria interessante comparar o AG proposto com um método de treinamento fino (do inglês, *fine tuning*) onde todas as camadas são retreinadas, mas os pesos são herdados do modelo inicial.

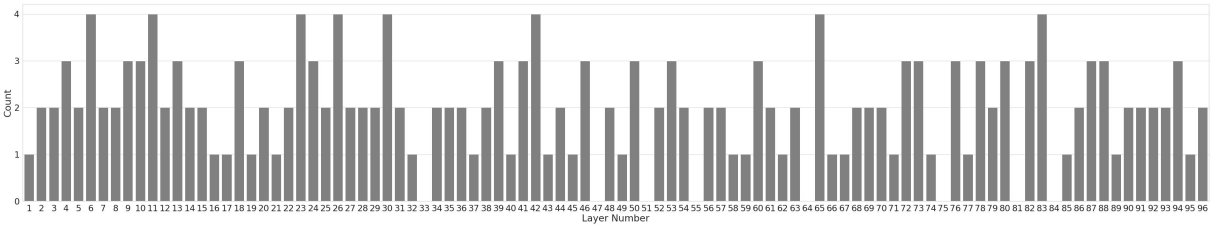

Figura 36 – Histograma das camadas selecionadas nas 4 execuções usando método de seleção por Roleta (Roulette).

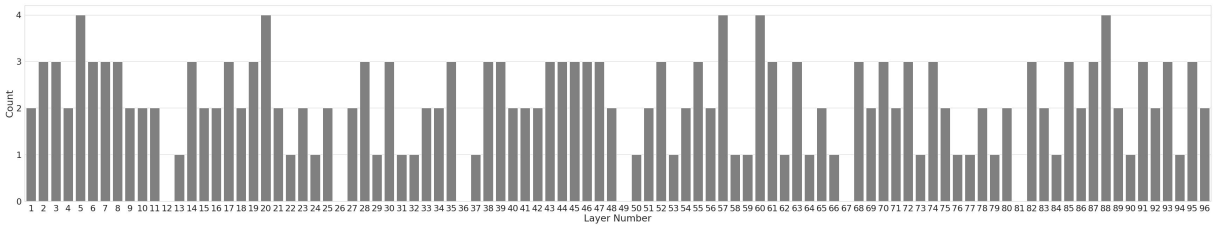

Figura 37 – Histograma das camadas selecionadas nas 4 execuções usando método de seleção por Torneio (Tournament).

Capítulo 6

Conclusão

Este trabalho teve como objetivo principal explorar duas abordagens do uso de AGs no processo evolucionário de CNN para classificação de imagens. Os dois métodos propostos expandem os métodos propostos na literatura com a adição de novos parâmetros evolutivos e conjuntos de dados estudados no caso do MLTLGA.

Inicialmente, o método proposto gaCNN foi capaz de evoluir arquiteturas CNN de maneira eficiente e com resultados promissores comparado a outros métodos disponíveis na literatura. As alterações propostas no cromossomo como a adição das funções de ativação e operadores de mutação ajudaram o AG a ter um desempenho superior mesmo quando o treinamento das CNN foi limitado a 1 época.

Uma outra abordagem de evolução utilizando AT foi proposta. Este método apresentou resultados animadores em relação a métodos bio-inspirados e não bio-inspirados, o que demonstra o poder e flexibilidade dos AG. As mudanças propostas na alteração dos parâmetros de inicialização e os métodos de seleção permitiram que este método superasse o AG proposto por Nagae, Kawai e Nobuhara (2020).

Portanto, através das análises conduzidas no Capítulo anterior, as hipóteses levantadas inicialmente, na Seção 1.2, podem ser avaliadas:

1. Desenvolver um AG para otimização do processo de AT; **O AG MLTLGA foi desenvolvido e apresentou resultados promissores para o processo de AT.**

2. Estender a técnica evoCNN com a introdução de otimização de funções de ativação e novos operadores genéticos; **Mudanças na representação cromossômica foram realizadas para adicionar as funções de otimização como aspecto evolutivo, e os operadores de mutação foram criados.**

3. Propor melhorias na representação cromossômica e dos operadores do evoCNN (SUN et al., 2020); **Adição do segundo vetor para representação cromossômica e operadores de mutação.**

4. Comparar o método desenvolvido com métodos bio-inspirados semelhantes;**O método gaCNN foi comparado com os métodos evoCNN e psoCNN e o método MLTLGA com o método NAGAE.**

6.1 Principais Contribuições

Devido à natureza das CNN, organizações estruturais, hiperparâmetros e mudanças na AT permitem que o estado da arte seja constantemente melhorado na construção de modelos de classificação mais poderosos. Neste trabalho, dois métodos que abordam aspectos distintos do universo de neuroevolução foram propostos. Ambos os métodos expandiram o estudo dessa área com resultados promissores através da adição de novos aspectos evolucionários e de inicialização das técnicas base.

Dentre as principais contribuições no gaCNN pode-se destacar: adição de funções de ativação no processo evolucionário e adição de operadores genéticos de mutação. No MLTLGA: a principal contribuição está na alteração do processo de inicialização e a exploração da solução proposta por Nagae, Kawai e Nobuhara (2020) a conjuntos de dados ainda não explorados e de caráter mais próximo de aplicações reais.

6.2 Trabalhos Futuros

Diversos aspectos da evolução de redes neurais através de AG ainda precisam ser estudados e aprimorados. O custo computacional ainda é um fator limitante neste estudo, e o desenvolvimento de métodos mais eficientes para avaliação de indivíduos se faz necessário. Além disso, a ampliação do estudo do método gaCNN para arquiteturas mais complexas, como a ResNet (HE et al., 2016b), devem ser exploradas e a avaliação dos métodos propostos em conjuntos de dados como o CIFAR-100 (KRIZHEVSKY; HINTON, 2020).

Outros aspectos, como a quantidade de parâmetros de uma CNN ou até a otimização de CNN para dispositivos de internet das coisas ou dispositivos móveis pode ser uma área promissora do uso de AG.

6.3 Contribuições em Produção Bibliográfica

Dois artigos científicos foram publicados durante o desenvolvimento desta pesquisa. Os artigos científicos foram publicados no *Congress on Evolutionary Computation (CEC) 2021*, evento QUALIS A1, segundo o Documento de Área da CAPES. São eles: *"Many Layer Transfer Learning Genetic Algorithm (MLTLGA): a New Evolutionary Transfer Learning Approach Applied To Pneumonia Classification*"e "*gaCNN: Composing CNNs and GAs to Build an Optimized Hybrid Classification Architecture*".

Referências

AMARAL, L. R. do; OLIVEIRA, G. M. B. de. Mineração de regras para classificação de oncogenes medidos por microarray utilizando algoritmos genéticos. **Revista Brasileira de Cancerologia**, v. 56, n. 3, p. 391–391, 2010. doi:10.32635/2176-9745.RBC.2010v56n3.1491.

BALDI, P.; SADOWSKI, P. J. Understanding dropout. **Advances in neural information processing systems**, v. 26, p. 2814–2822, 2013. doi:10.5555/2999792.2999926.

BOUREAU, Y.-L.; PONCE, J.; LECUN, Y. A theoretical analysis of feature pooling in visual recognition. In: **Proceedings of the 27th international conference on machine learning (ICML-10)**. [S.l.: s.n.], 2010. p. 111–118. doi:10.5555/3104322.3104338.

CARNEIRO, M. G. et al. Abordagens baseadas em autômatos celulares síncronos para o escalonamento estático de tarefas em multiprocessadores. Universidade Federal de Uberlândia, 2012.

CHAN, T.-H. et al. Pcanet: A simple deep learning baseline for image classification? **IEEE transactions on image processing**, IEEE, v. 24, n. 12, p. 5017–5032, 2015. doi:10.1109/TIP.2015.2475625.

CHEN, Y. et al. Automated design of neural network architectures with reinforcement learning for detection of global manipulations. **IEEE Journal of Selected Topics in Signal Processing**, IEEE, v. 14, n. 5, p. 997–1011, 2020. doi:10.1109/JSTSP.2020.2998401.

CUI, P.; SHABASH, B.; WIESE, K. C. Evodnn-an evolutionary deep neural network with heterogeneous activation functions. In: IEEE. **2019 IEEE Congress on Evolutionary Computation (CEC)**. [S.l.], 2019. p. 2362–2369. doi:10.1109/CEC.2019.8789964.

DEB, K.; AGRAWAL, R. B. et al. Simulated binary crossover for continuous search space. **Complex systems**, Citeseer, v. 9, n. 2, p. 115–148, 1995.

DEB, K.; DEB, D. Analysing mutation schemes for real-parameter genetic algorithms. **International Journal of Artificial Intelligence and Soft Computing**, Inderscience Publishers Ltd, v. 4, n. 1, p. 1–28, 2014. doi:10.1504/IJAISC.2014.059280.

DENG, L. The mnist database of handwritten digit images for machine learning research. **IEEE Signal Processing Magazine**, IEEE, v. 29, n. 6, p. 141–142, 2012. doi:10.1109/MSP.2012.2211477.

EGMONT-PETERSEN, M.; RIDDER, D. de; HANDELS, H. Image processing with neural networks—a review. **Pattern recognition**, Elsevier, v. 35, n. 10, p. 2279–2301, 2002. doi:10.1109/TIP.2015.2475625.

FERNANDO, C. et al. Pathnet: Evolution channels gradient descent in super neural networks. **arXiv preprint arXiv:1701.08734**, 2017.

GOLDBERG, D. E. **Genetic Algorithms in Search, Optimization and Machine Learning**. USA: Addison-Wesley Longman Publishing Co., Inc., 1989. doi:10.5555/534133.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. [S.l.]: MIT Press, 2016. doi:10.5555/3086952.

HE, K. et al. Deep residual learning for image recognition. In: **2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2016. p. 770–778. doi:10.1109/CVPR.2016.90.

_____. Identity mappings in deep residual networks. In: SPRINGER. **European conference on computer vision**. [S.l.], 2016. p. 630–645. doi:10.1007/978-3-319-46493-$0_3$8.

HINTON, G. E.; SEJNOWSKI, T. J. et al. Learning and relearning in boltzmann machines. **Parallel distributed processing: Explorations in the microstructure of cognition**, v. 1, n. 282-317, p. 2, 1986.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. **Neural computation**, MIT Press, v. 9, n. 8, p. 1735–1780, 1997. doi:10.1162/neco.1997.9.8.1735.

HOLLAND, J. H. **Hierarchical Descriptions, Universal Spaces and Adaptive Systems.** [S.l.], 1968. doi:10.1038.

_____. Genetic algorithms. **Scientific American**, JSTOR, v. 267, n. 1, p. 66–73, 1992. doi:10.1038/scientificamerican0792-66.

IMAI, S.; KAWAI, S.; NOBUHARA, H. Stepwise pathnet: a layer-by-layer knowledge-selection-based transfer learning algorithm. **Scientific Reports**, Nature Publishing Group, v. 10, n. 1, p. 1–14, 2020. doi:10.1038/s41598-020-64165-3.

JUNIOR, F. E. F.; YEN, G. G. Particle swarm optimization of deep neural networks architectures for image classification. **Swarm and Evolutionary Computation**, Elsevier, v. 49, p. 62–74, 2019. doi:10.1016/j.swevo.2019.05.010.

KENNEDY, J.; EBERHART, R. Particle swarm optimization. In: **Proceedings of ICNN'95 - International Conference on Neural Networks**. [S.l.: s.n.], 1995. v. 4, p. 1942–1948 vol.4. doi:10.1109/ICNN.1995.488968.

KERMANY, D. S. et al. Identifying medical diagnoses and treatable diseases by image-based deep learning. **Cell**, Elsevier, v. 172, n. 5, p. 1122–1131, 2018. doi:10.1016/j.cell.2018.02.010.

KRAMER, M. A. Nonlinear principal component analysis using autoassociative neural networks. **AIChE journal**, Wiley Online Library, v. 37, n. 2, p. 233–243, 1991. doi:10.1002/aic.690370209.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: PEREIRA, F. et al. (Ed.). **Advances in Neural Information Processing Systems**. Curran Associates, Inc., 2012. v. 25. Disponível em: <https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf>.

KRIZHEVSKY, V. N. A.; HINTON, G. **CIFAR-100 (Canadian Institute for Advanced Research)**. 2020. Acesso em: 2020-01-23. Disponível em: <http://www.cs.toronto.edu/~kriz/cifar.html>.

KUMAR, N. **Understanding Convolution Neural Networks: CNN the ELI5 way**. 2019. Acesso em: 2020-01-23. Disponível em: <https://towardsdatascience.com/understanding-convolution-neural-networks-the-eli5-way-785330cd1fb7>.

LECUN, Y.; BENGIO, Y. et al. Convolutional networks for images, speech, and time series. **The handbook of brain theory and neural networks**, n. 10, 1995. doi:10.5555/303568.303704.

LECUN, Y. et al. Gradient-based learning applied to document recognition. **Proceedings of the IEEE**, Ieee, v. 86, n. 11, p. 2278–2324, 1998. doi:10.1109/5.726791.

LEE, C.-Y. et al. Deeply-supervised nets. In: **Artificial intelligence and statistics**. [S.l.: s.n.], 2015. p. 562–570.

LIANG, M.; HU, X. Recurrent convolutional neural network for object recognition. In: **2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2015. p. 3367–3375. doi:10.1109/CVPR.2015.7298958.

LIU, C. et al. Progressive neural architecture search. In: **Proceedings of the European Conference on Computer Vision (ECCV)**. [S.l.: s.n.], 2018. p. 19–34. doi:10.1007/978-3-030-01246-5_2.

LIU, M. et al. Towards better analysis of deep convolutional neural networks. **IEEE transactions on visualization and computer graphics**, IEEE, v. 23, n. 1, p. 91–100, 2016. doi:10.1109/TVCG.2016.2598831.

MAHDIANPARI, M. et al. Very deep convolutional neural networks for complex land cover mapping using multispectral remote sensing imagery. **Remote Sensing**, Multidisciplinary Digital Publishing Institute, v. 10, n. 7, p. 1119, 2018. doi:10.3390/rs10071119.

MARTINEZ, A. D. et al. Lights and shadows in evolutionary deep learning: Taxonomy, critical methodological analysis, cases of study, learned lessons, recommendations and challenges. **Information Fusion**, Elsevier, v. 67, p. 161–194, 2021. doi:10.1016/j.inffus.2020.10.014.

MIIKKULAINEN, R. et al. Evolving deep neural networks. In: **Artificial Intelligence in the Age of Neural Networks and Brain Computing**. [S.l.]: Elsevier, 2019. p. 293–312. doi:10.1016/B978-0-12-815480-9.

NAGAE, S.; KAWAI, S.; NOBUHARA, H. Transfer learning layer selection using genetic algorithm. In: IEEE. **2020 IEEE Congress on Evolutionary Computation (CEC)**. [S.l.], 2020. p. 1–6. doi:10.1109/CEC48606.2020.9185501.

PASZKE, A. et al. Pytorch: An imperative style, high-performance deep learning library. In: WALLACH, H. et al. (Ed.). **Advances in Neural Information Processing Systems 32**. Curran Associates, Inc., 2019. p. 8024–8035. Disponível em: <http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf>.

REAL, E. et al. Large-scale evolution of image classifiers. In: PRECUP, D.; TEH, Y. W. (Ed.). **Proceedings of the 34th International Conference on Machine Learning**. [S.l.]: PMLR, 2017. (Proceedings of Machine Learning Research, v. 70), p. 2902–2911.

RECENT advances in convolutional neural networks. **Pattern Recognition**, v. 77, p. 354–377, 2018. ISSN 0031-3203. doi:10.1016/j.patcog.2017.10.013.

REDMON, J. et al. You only look once: Unified, real-time object detection. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2016. p. 779–788. doi:10.1109/CVPR.2016.91.

RIFAI, S. et al. Contractive auto-encoders: Explicit invariance during feature extraction. In: **ICML'11: Proceedings of the 28th International Conference on International Conference on Machine Learning**. [S.l.]: Omnipress, 2011. doi:10.5555/3104482.3104587.

ROJAS, R. The backpropagation algorithm. In: **Neural networks**. [S.l.]: Springer, 1996. p. 149–182. doi:10.1007/978-3-642-61068-4_7.

ROSENBLATT, F. The perceptron: a probabilistic model for information storage and organization in the brain. **Psychological review**, American Psychological Association, v. 65, n. 6, p. 386, 1958. doi:10.1037/h0042519.

RUSSAKOVSKY, O. et al. Imagenet large scale visual recognition challenge. **International journal of computer vision**, Springer, v. 115, n. 3, p. 211–252, 2015. doi:10.1007/s11263-015-0816-y.

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. In: BENGIO, Y.; LECUN, Y. (Ed.). **3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings**. [s.n.], 2015. Disponível em: <http://arxiv.org/abs/1409.1556>.

SINHA, T.; HAIDAR, A.; VERMA, B. Particle swarm optimization based approach for finding optimal values of convolutional neural network parameters. In: IEEE. **2018 IEEE Congress on Evolutionary Computation (CEC)**. [S.l.], 2018. p. 1–6. doi:10.1109/CEC.2018.8477728.

STANLEY, K. O.; MIIKKULAINEN, R. Evolving neural networks through augmenting topologies. **Evolutionary computation**, MIT Press, v. 10, n. 2, p. 99–127, 2002.

STOJNIC, R. **Image Classification on MNIST**. 2020. Acesso em: 2020-01-23. Disponível em: <https://paperswithcode.com/sota/image-classification-on-mnist>.

SUN, Y. et al. Evolving deep convolutional neural networks for image classification. **IEEE Transactions on Evolutionary Computation**, v. 24, n. 2, p. 394–407, 2020. doi:10.1109/TEVC.2019.2916183.

SZEGEDY, C. et al. Rethinking the inception architecture for computer vision. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2016. p. 2818–2826. doi:10.1109/CVPR.2016.308.

TORREY, L.; SHAVLIK, J. Transfer learning. In: **Handbook of research on machine learning applications and trends: algorithms, methods, and techniques**. [S.l.]: IGI global, 2010. p. 242–264. doi:10.4018/978-1-60566-766-9.ch011.

VALENTINI, G.; MASULLI, F. Ensembles of learning machines. In: SPRINGER. **Italian workshop on neural nets**. [S.l.], 2002. p. 3–20. doi:10.1007/3-540-45808-5_1.

WANG, B. et al. Evolving deep convolutional neural networks by variable-length particle swarm optimization for image classification. In: IEEE. **2018 IEEE Congress on Evolutionary Computation (CEC)**. [S.l.], 2018. p. 1–8. doi:10.1109/CEC.2018.8477735.

WANG, T. et al. End-to-end text recognition with convolutional neural networks. In: IEEE. **Proceedings of the 21st international conference on pattern recognition (ICPR2012)**. [S.l.], 2012. p. 3304–3308.

WANG, Y. et al. The influence of the activation function in a convolution neural network model of facial expression recognition. **Applied Sciences**, Multidisciplinary Digital Publishing Institute, v. 10, n. 5, p. 1897, 2020. doi:10.3390/app10051897.

WHITLEY, D.; STARKWEATHER, T.; BOGART, C. Genetic algorithms and neural networks: Optimizing connections and connectivity. **Parallel computing**, North-Holland, v. 14, n. 3, p. 347–361, 1990. doi:10.1016/0167-8191(90)90086-O.

WIJNHOVEN, R. G.; WITH, P. de. Fast training of object detection using stochastic gradient descent. In: IEEE. **2010 20th International Conference on Pattern Recognition**. [S.l.], 2010. p. 424–427. doi:10.1109/ICPR.2010.112.

WILLIAMS, D. P. Transfer learning with sas-image convolutional neural networks for improved underwater target classification. In: IEEE. **IGARSS 2019-2019 IEEE International Geoscience and Remote Sensing Symposium**. [S.l.], 2019. p. 78–81. doi:10.1109/IGARSS.2019.8898611.

WISTUBA, M. Deep learning architecture search by neuro-cell-based evolution with function-preserving mutations. In: SPRINGER. **Joint European Conference on Machine Learning and Knowledge Discovery in Databases**. [S.l.], 2018. p. 243–258. doi:10.1007/978-3-030-10928-$8_1$5.

WISTUBA, M.; RAWAT, A.; PEDAPATI, T. A survey on neural architecture search. **arXiv preprint arXiv:1905.01392**, 2019.

WU, T. et al. A multi-objective particle swarm optimization for neural networks pruning. In: IEEE. **2019 IEEE Congress on Evolutionary Computation (CEC)**. [S.l.], 2019. p. 570–577. doi:10.1109/CEC.2019.8790145.

XIAO, H.; RASUL, K.; VOLLGRAF, R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. **arXiv preprint arXiv:1708.07747**, 2017.

_____. **Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms**. 2017.

ZHOU, X.-H. et al. Shallow and deep neural network training by water wave optimization. **Swarm and Evolutionary Computation**, Elsevier, v. 50, p. 100561, 2019. doi:10.1016/j.swevo.2019.100561.

ZOPH, B. et al. Learning transferable architectures for scalable image recognition. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2018. p. 8697–8710. doi:10.1109/CVPR.2018.00907.