UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

CENTRO UNIVERSITÁRIO NORTE DO ESPÍRITO SANTO

PROGRAMA DE PÓS-GRADUAÇÃO EM ENERGIA

VINICIUS WITTIG VIANNA

# REDE NEURAL CONVOLUCIONAL AUTOCONFIGURADA PARA IDENTIFICAÇÃO DE CARGAS ELÉTRICAS SIMILARES EM SMART GRID

SÃO MATEUS

2021

VINICIUS WITTIG VIANNA

# REDE NEURAL CONVOLUCIONAL AUTOCONFIGURADA PARA IDENTIFICAÇÃO DE CARGAS ELÉTRICAS SIMILARES EM SMART GRID

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Energia da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Mestre em Energia.
Orientador: Prof. Dr. Wanderley Cardoso Celeste.
Coorientador: Prof. Dr. Helder Roberto de Oliveira Rocha.

SÃO MATEUS
2021

Ficha catalográfica disponibilizada pelo Sistema Integrado de Bibliotecas - SIBI/UFES e elaborada pelo autor

---

Wittig Vianna, Vinicius, 1991-
W832r Rede neural convolucional autoconfigurada para identificação de cargas elétricas similares em smart grid / Vinicius Wittig Vianna. - 2021.
163 f. : il.

Orientador: Wanderley Cardoso Celeste.
Coorientador: Helder Roberto de Oliveira Rocha.
Dissertação (Mestrado em Energia) - Universidade Federal do Espírito Santo, Centro Universitário Norte do Espírito Santo.

1. NILM. 2. Aprendizado de Máquina. 3. Otimização Heurística. I. Cardoso Celeste, Wanderley. II. Oliveira Rocha, Helder Roberto de. III. Universidade Federal do Espírito Santo. Centro Universitário Norte do Espírito Santo. IV. Título.

CDU: 620.9

---

**Vinícius Wittig Vianna**

# REDE NEURAL CONVOLUCIONAL AUTOCONFIGURADA PARA IDENTIFICAÇÃO DE CARGAS ELÉTRICAS SIMILARES EM SMART GRID

Dissertação apresentada ao Programa de Pós-Graduação em Energia da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do título de Mestre em Energia.

Aprovada em 29 de março de 2021.

**COMISSÃO EXAMINADORA**

**Prof. Dr. Wanderley Cardoso Celeste**
**Universidade Federal do Espírito Santo**
**Orientador**

**Prof. Dr. Daniel José Custódio Coura**
**Universidade Federal do Espírito Santo**

**Prof. Dr. Leonardo José Silvestre**
**Universidade Federal do Espírito Santo**

**Prof. Dr. Hélder Roberto de Oliveira Rocha**
**Universidade Federal do Espírito Santo**

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

**PROTOCOLO DE ASSINATURA**

O documento acima foi assinado digitalmente com senha eletrônica através do Protocolo Web, conforme Portaria UFES nº 1.269 de 30/08/2018, por
WANDERLEY CARDOSO CELESTE - SIAPE 1723581
Departamento de Computação e Eletrônica - DCE/CEUNES
Em 29/03/2021 às 13:06

Para verificar as assinaturas e visualizar o documento original acesse o link:
https://api.lepisma.ufes.br/arquivos-assinados/165698?tipoArquivo=O

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

**PROTOCOLO DE ASSINATURA**

O documento acima foi assinado digitalmente com senha eletrônica através do Protocolo Web, conforme Portaria UFES nº 1.269 de 30/08/2018, por
DANIEL JOSE CUSTODIO COURA - SIAPE 1870073
Departamento de Computação e Eletrônica - DCE/CEUNES
Em 29/03/2021 às 15:06

Para verificar as assinaturas e visualizar o documento original acesse o link:
https://api.lepisma.ufes.br/arquivos-assinados/165796?tipoArquivo=O

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

**PROTOCOLO DE ASSINATURA**

O documento acima foi assinado digitalmente com senha eletrônica através do Protocolo Web, conforme Portaria UFES nº 1.269 de 30/08/2018, por
LEONARDO JOSE SILVESTRE - SIAPE 1504334
Departamento de Computação e Eletrônica - DCE/CEUNES
Em 29/03/2021 às 15:52

Para verificar as assinaturas e visualizar o documento original acesse o link:
https://api.lepisma.ufes.br/arquivos-assinados/165845?tipoArquivo=O

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

**PROTOCOLO DE ASSINATURA**

O documento acima foi assinado digitalmente com senha eletrônica através do Protocolo Web, conforme Portaria UFES nº 1.269 de 30/08/2018, por
HELDER ROBERTO DE OLIVEIRA ROCHA - SIAPE 1860639
Departamento de Engenharia Elétrica - DEE/CT
Em 29/03/2021 às 16:04

Para verificar as assinaturas e visualizar o documento original acesse o link:
https://api.lepisma.ufes.br/arquivos-assinados/165852?tipoArquivo=O

# Agradecimentos

Agradeço à Deus pela oportunidade da vida e por me permitir questionar e refletir sobre as coisas do universo.

Aos meus familiares, por proporcionar todo o suporte necessário, me permitindo manter o foco exclusivo ao trabalho e à pesquisa.

Ao meu orientador, professor Dr. Wanderley Cardoso Celeste, por todo o conhecimento compartilhado, pelo suporte às incontáveis dúvidas e, mais do que isso, por sua humildade no trato com seus orientados.

Ao meu coorientador, professor Dr. Helder Roberto de Oliveira Rocha, pela presteza e suporte nos momentos difíceis, em que as (meta)heurísticas se mostravam complexas, ao mesmo tempo que revelavam toda sua beleza.

Ao professor, Dr. Leonardo José Silvestre, pela atitude solícita em prestar suporte, mesmo sem nenhum tipo de compromisso, nas dúvidas e modelagens "pythônicas".

Aos demais professores do Programa, em especial, à Dra. Gisele de Lorena Diniz Chaves, Dr. Daniel da Cunha Ribeiro, Dr. Rodrigo Randow de Freitas e Dr. Paulo Sérgio da Silva Porto, por agregarem incontáveis conhecimentos e por transformarem minha maneira de pensar sobre Energia.

Aos profissionais terceirizados da manutenção e limpeza, por garantirem um ambiente adequado para o estudo e pesquisa.

À Universidade Federal do Espírito Santo, mais especificamente, ao Centro Universitário Norte do Espírito Santo (CEUNES), à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) e à Fundação de Amparo à Pesquisa e Inovação do Espírito Santo (FAPES), por proporcionarem e contribuírem nesta qualificação gratuita de alto nível, através do Programa de Pós-graduação em Energia (PPGEN).

Finalmente, a todos que, direta ou indiretamente, contribuíram com palavras de conforto ou com diferentes pontos de vista, o que me fez enxergar este desafio sob novas perspectivas.

"*Meu cérebro é apenas um receptor, no universo existe um núcleo a partir do qual obtemos conhecimento, força e inspiração. Eu não penetrei nos segredos deste núcleo, mas eu sei que ele existe*."

Nikola Tesla

# Lista de figuras

# Lista de tabelas

# Sumário

# Resumo

Redes Neurais Convolucionais (CNN) têm se mostrado ferramentas muito eficientes na tarefa de identificação de cargas elétricas similares em *smart grid*. Contudo, tais redes atualmente dependem de mão-de-obra altamente especializada para serem adequadamente arquitetadas, tendo em vista a grande quantidade de hiperparâmetros e variedade de ajustes, o que torna este empreendimento altamente trabalhoso e custoso. Além disso, os ajustes tradicionalmente manuais e completamente empíricos não garantem a obtenção de uma arquitetura ótima, devido à impossibilidade, em geral, de se testar todas as combinações de ajuste para um conjunto de hiperparâmetros, dentro de um espaço de valores previamente definido. Logo, este trabalho tem como objetivo principal agregar um mecanismo de ajuste automatizado dos hiperparâmetros de uma CNN dedicada à identificação autônoma de cargas elétricas altamente similares em *smart grid*. Para isso, é inicialmente utilizado um sistema de classificação baseado em uma arquitetura CNN manualmente obtida em trabalho prévio, a fim de se encontrar a quantidade mínima de casos necessários e suficientes para permitir, estrategicamente, uma acurácia de classificação de pelo menos 95%. Em seguida, a quantidade ótima de casos é usada para otimizar o número de camadas convolucionais e densas da CNN, além do número de neurônios em tais camadas, sem comprometer o desempenho da arquitetura de referência (a manualmente ajustada). O sistema foi testado usando dois conjuntos de dados, um baseado em arranjos de até quatro lâmpadas fluorescentes tecnicamente idênticas e outro baseado em arranjos de até quatro microcomputadores também tecnicamente idênticos. Com o primeiro conjunto, a redução do número de casos necessários para o treinamento da CNN de referência foi de 90%, enquanto que no segundo caso, foi de 33,4%. Em seguida, as respectivas quantidades mínimas de casos foram usadas para ajuste de hiperparâmetros a partir da CNN de referência, resultando em uma redução de 56,02% e 90,41% sobre o número de parâmetros treináveis de tais redes, a partir dos respectivos conjuntos de dados.

**Palavras-chave:** NILM. Aprendizado de Máquina Automatizado. Otimização Heurística. Custo Computacional.

# Abstract

Convolutional Neural Networks (CNN) have been shown to be very efficient tools in the task of identifying similar electrical charges in a smart grid. However, these networks currently depend on highly skilled labor to be properly designed, in view of the large number of hyperparameters and variety of adjustments, which makes this undertaking highly laborious and costly. In addition, traditionally manual and completely empirical adjustments do not guarantee the achievement of an optimal architecture, due to the impossibility, in general, of testing all adjustment combinations for a set of hyperparameters, within a previously defined value space. Therefore, this work has as main objective to add an automated adjustment mechanism of the hyperparameters of a CNN dedicated to the autonomous identification of highly similar electrical charges in a smart grid. For this purpose, a classification system based on a CNN architecture manually obtained from previous work is initially used, in order to find the minimum number of necessary and sufficient cases to strategically allow a classification accuracy of at least 95%. Then, the optimum number of cases is used to optimize the number of CNN convolutional and dense layers, in addition to the number of neurons in such layers, without compromising the performance of the reference architecture (the manually adjusted one). The system was tested using two sets of data, one based on arrays of up to four technically identical fluorescent lamps and the other based on arrays of up to four microcomputers also technically identical. With the first set, the reduction in the number of cases required for training the reference CNN was 90%, while in the second case, it was 33.4%. Then, the respective minimum number of cases were used to adjust hyperparameters from the reference CNN, resulting in a reduction of 56.02% and 90.41% over the number of trainable parameters of such networks, from the respective databases.



**Keywords:** NILM. Automated Machine Learning. Heuristic Optimization. Computational Cost.

# 1 Introdução

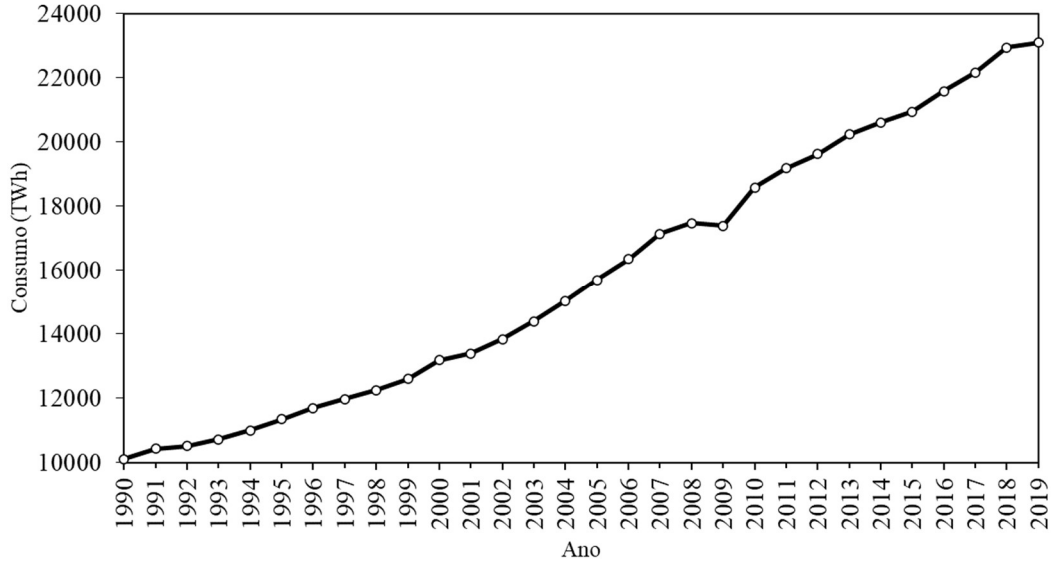
## 1.1 Motivação

Concomitantemente às questões climáticas globais, o fator sustentabilidade tem se tornado aspecto preponderante no tocante, tanto à cultura organizacional e ao planejamento estratégico dos grandes *players* mundiais do setor energético, quanto ao uso racional de energia pelo consumidor final, de modo que, perseguir altos índices de competitividade financeira ou, simplesmente, fazer uso deste tipo de recurso de maneira arbitrária, não mais representa uma condição satisfatória de sustentabilidade, considerando os prismas ambiental e econômico.

Em (KOBER et al., 2020) é dito que a demanda por energia elétrica tende a ser duplicada até o ano de 2060, devido ao desenvolvimento da classe média, ao aumento médio da renda da população e à inclusão de dispositivos e máquinas elétricas. Tais efeitos tendem a demandar mais investimentos em infraestrutura e em sistemas de monitoramento para que seja possível garantir a sustentabilidade energética.

Em paralelo, a partir do último banco de dados aberto, disponibilizado gratuitamente em (ENERDATA, 2021), é possível observar um aumento acentuado no consumo mundial de eletricidade (Figura 1), revelando a importância da discussão e do desenvolvimento de tecnologias que tornem seu gerenciamento mais eficiente.

Figura 1 – Consumo de eletricidade por ano (mundial).

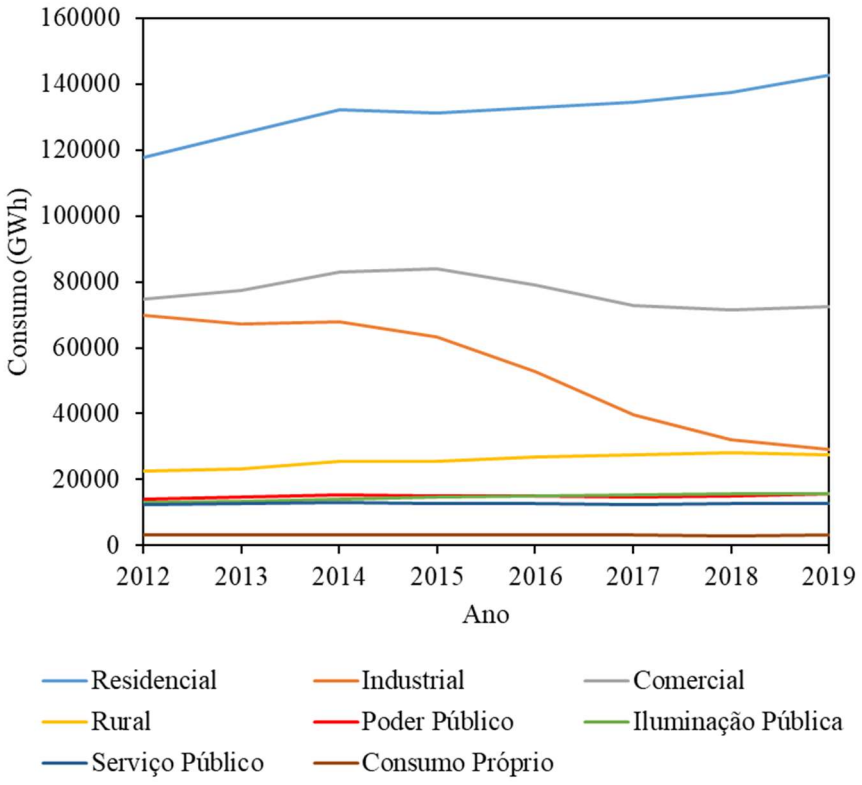



Fonte: Enerdata. *Electricity domestic consumption* (2020)

De maneira análoga aos dados apresentados, revela-se importante especificar os principais agentes contribuintes para esta tendência. A eletricidade e o aumento populacional representam fatores preponderantes no desenvolvimento de uma economia, os quais exercem influências relacionadas à urbanização, industrialização, infraestrutura energética e acesso à novas tecnologias. Paralelamente, percebe-se que nações com maior investimento em planejamento energético possuem maior robustez econômica, sugerindo que o consumo de eletricidade representa importante elemento de estratégia governamental e industrial (APERGIS; PAYNE, 2011).

Partindo para uma análise mais específica, como no caso do Brasil, percebe-se um comportamento singular no consumo de eletricidade (Figura 2). Segundo o banco de dados mais recente, disponibilizado pela Empresa de Pesquisa Energética (EPE), identifica-se um crescimento médio no consumo de eletricidade em 2019 de cerca de 1,6% maior em relação à 2018. Contudo, observa-se uma queda na classe industrial, puxada, sobretudo, pelo decréscimo na demanda do setor de extração de minerais metálicos que, segundo o documento, apresenta queda acentuada de 10,3% só em 2019 (EPE, 2021).

Figura 2 – Consumo médio de eletricidade, por classes (Brasil).




Fonte: Empresa de Pesquisa Energética (EPE). Consumo Anual de Energia Elétrica por classe

Este perfil corrobora com o comportamento de consumo globalizado evidenciado em (APERGIS; PAYNE, 2011), de modo que o consumo de energia elétrica tende a crescer à medida que as nações buscam, dentro de suas possibilidades, viabilizar maior industrialização, urbanização, acesso à novas tecnologias e aumento da qualidade de vida. No entanto, esta condição de aumento constante no consumo de recursos energéticos de eletricidade, tanto a nível pontual como no caso do Brasil, quanto a nível global, tem acarretado em consequências ambientais adversas.

Estudos mais recentes, como em (JACK; STEPHENSON; KHAN, 2018), indicam que há uma tendência de que a alta demanda por energia elétrica continue sendo um efeito do elevado crescimento econômico, populacional e industrial, potencializando, assim, a emissão de gases de efeito estufa. De acordo com os autores, a redução da emissão global desses gases pode ser conquistada através de duas principais frentes de trabalho: viabilizar investimentos de maneira que a geração de energias renováveis assuma um percentual majoritário na matriz energética global; e aperfeiçoar o uso da capacidade de energia renovável e não renovável.

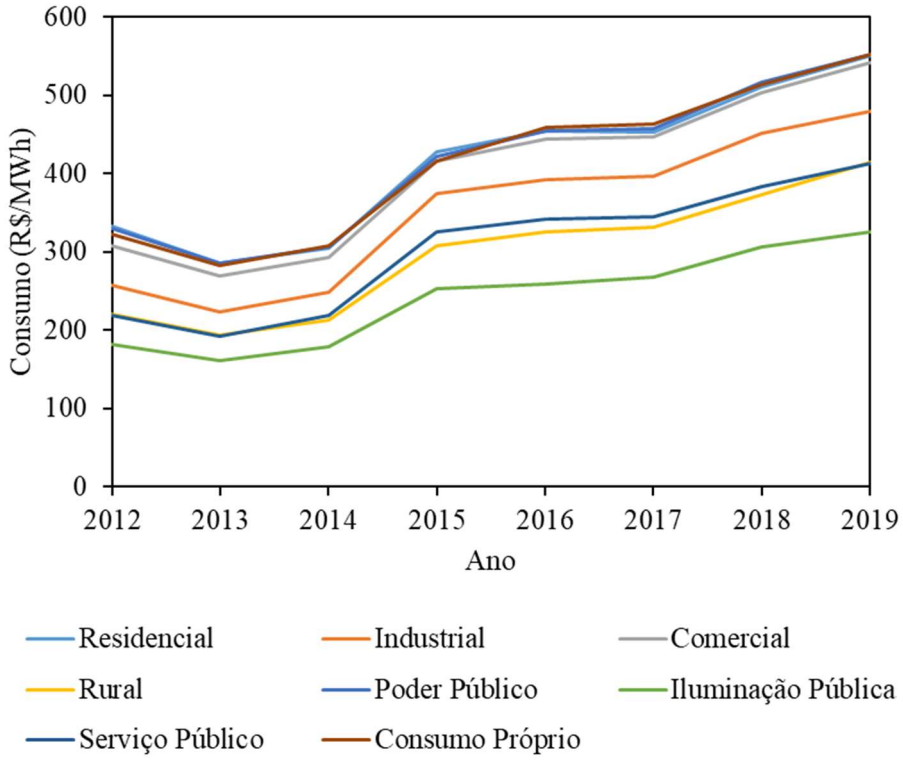
Paralelamente à análise de consumo de eletricidade global, torna-se também relevante trazer à luz aspectos econômicos envolvidos neste contexto, especificamente os custos repassados ao consumidor final. Verifica-se na série histórica compreendida entre 2012 e 2019 (Figura 3), um perfil de aumento acentuado na tarifa média anual referente ao consumo de eletricidade. Esse comportamento também contribui para a relevância em discutir metodologias mais eficientes no que diz respeito ao gerenciamento de energia elétrica (EPE, 2021). É neste contexto que novas tecnologias podem ser utilizadas, promovendo o uso e o gerenciamento de eletricidade de forma mais eficiente, no sentido de reduzir os custos percebidos, assim como a emissão de gases de efeito estufa, em detrimento de sua produção.

Seja em ambientes industriais, comerciais ou residenciais, uma gama de tecnologias digitais tem sido utilizada, sobretudo, impulsionadas pelo contexto energético da Indústria 4.0 (VIANNA; CELESTE; FREITAS, 2019), com o objetivo de lidar, estrategicamente, desde o controle de processos até a aquisição de informações implícitas (oriundas do processamento de grandes quantidades de dados), promovendo, portanto, a otimização no consumo de recursos, o que foi abordado em (WANG, Y. et al., 2019).

Tecnologias como redes inteligentes (WANG, Y. et al., 2019), internet das coisas (SISINNI et al., 2018), redes de sensores sem fio (LIN et al., 2016), gerenciamento de recursos em nuvem (CHENG; LIU, 2018), aplicações generalizadas de inteligência artificial (BUKATA

et al., 2017) e processamento de grandes massas de dados (ALAHAKOON; YU, 2016), têm causado modificações profundas no modo de gerenciar operações e recursos energéticos de forma mais assertiva e econômica. Nessa linha de raciocínio, pautada pelo aumento na demanda de eletricidade e seus respectivos impactos ambientais, emerge como tópico de relevância o monitoramento inteligente das redes elétricas.

Figura 3 – Tarifas médias anuais por classes de consumo (Brasil).




Fonte: Empresa de Pesquisa Energética (EPE). Anuário Estatístico de Energia Elétrica

No trabalho investigativo de (GANGALE; MENGOLINI; ONYEJI, 2013), é evidenciado que o futuro das redes elétricas através das redes inteligentes, ou *Smart Grids*, permitirá o fluxo bidirecional de comunicação e energia elétrica entre fornecedores e consumidores, devido à implementação massiva de tecnologias de informação e comunicação, de modo que os usuários finais, tradicionalmente passivos (com pouco ou nenhum poder de decisão), transformem-se em usuários ativos (com alto poder de decisão).

As *Smart Grids* representam uma gama relevante de oportunidades no caminho de promover eficiência, confiabilidade e disponibilidade, afetando aspectos econômicos e ambientais através da transmissão de eletricidade mais eficiente, restauração da rede elétrica com maior rapidez e redução de custos operacionais e de gerenciamento, acarretando em custos de eletricidade mais baixos para os consumidores, menor pico de demanda, integração de

sistemas de energia em pequena e larga escala, assim como segurança aprimorada (U.S. DEPARTMENT OF ENERGY, 2021).

Em (ALAM; REAZ; ALI, 2012) complementou-se ainda que o fornecedor pode controlar, de maneira implícita, os equipamentos conectados à rede, permitindo a manutenção da qualidade durante o fornecimento de energia elétrica através de medidores inteligentes, que representam parte fundamental em uma rede, permitindo o controle estratégico de energia. Ademais, a sinergia de residências, redes e medidores inteligentes, representa um aspecto indispensável aos fornecedores e consumidores de eletricidade.

Permeando o conceito de *Smart Grid* e as possibilidades que a aplicação de tal conceito podem proporcionar, é significativo salientar que as informações de consumo de eletricidade são geralmente fornecidas ao consumidor através de taxas, as quais não fornecem o detalhamento do consumo elétrico. Por conseguinte, é fundamental a realização do monitoramento instantâneo de cargas, o que permitirá o detalhamento energético atribuído à composição do consumo. Esse nível de informação fornecerá ao usuário o poder de decisão ativa baseada, por exemplo, em critérios econômicos (WU et al., 2017).

Portanto, apresentados os dados que evidenciam o perfil de intensificação do consumo de energia elétrica e possíveis implicações adversas de ordem ambiental, revela-se importante investigar meios de consumi-la de forma mais coerente. O monitoramento de cargas conectadas à rede elétrica, através do emprego de sistemas baseados em aprendizagem profunda representa um passo importante por viabilizar o controle e a supervisão de variáveis inerentes ao processo de maneira mais rápida e menos custosa, resultando em eficiência energética e maior sustentabilidade.

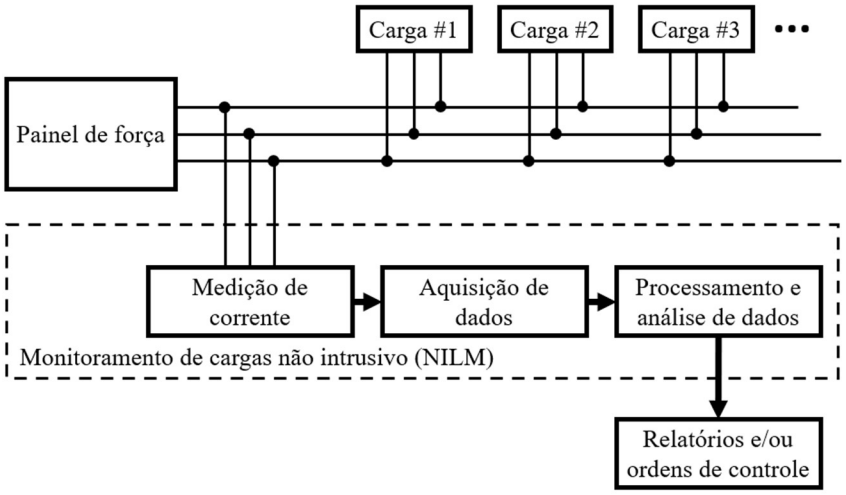
## 1.2 Justificativa

Declarados os aspectos de motivação deste trabalho, justifica-se a averiguação e o desenvolvimento de ferramentas que possibilitam gerenciar recursos de energia elétrica com maior estratégia e segurança, em menor tempo possível, utilizando, para isso, o emprego de tecnologias inteligentes que possibilitem fornecer aos interessados um grau de informação mais acurado, como o detalhamento sobre o consumo elétrico instantâneo ou perturbações no funcionamento de um equipamento conectado à rede, viabilizando planejamento e tomadas de

decisão mais assertivas, oportunizando segurança operacional, economia de recursos e preservação do meio ambiente.

Com relação à concepção de gerenciamento de energia elétrica baseada, em grande parte, na utilização de medidores inteligentes, cabe ressaltar o monitoramento instantâneo de cargas. Tal procedimento pode ser realizado de forma intrusiva ou de forma não intrusiva, sendo esta última apontada na literatura como a mais viável na grande maioria das vezes devido a uma série de fatores como, por exemplo, não depender de uma intervenção na infraestrutura das instalações elétricas (LIANG, J. et al., 2010). Portanto, o monitoramento não intrusivo de cargas elétricas (NILM), apresentado na Figura 4, tem proporcionado benefícios técnicos e financeiros, culminando na intensificação do uso de *smart grids* em escala global (KONG et al., 2020).

Figura 4 – Esquema de sistema baseado em monitoramento não intrusivo (NILM).




Fonte: Adaptado de (HE et al., 2016)

O NILM tem alavancado a utilização de medidores inteligentes nas mais diversas aplicações (HE et al., 2016), no entanto, tais medidores carecem de maior desenvolvimento. É nesta circunstância que tecnologias baseadas em conceitos da computação científica e inteligência artificial, mais especificamente, o aprendizado de máquina (*machine learning*) e a aprendizagem profunda (*deep learning*), ganham relevância no cenário de investigação acadêmica especializada em gerenciamento de recursos elétricos, como se pode observar em (SHIN et al., 2019), (LI, Ding; DICK, 2019), (XIA et al., 2019), (CHEN, K. et al., 2018), (GILLIS; CHUNG; MORSI, 2018), (TABATABAEI; DICK; XU, 2017) e (CHANG et al., 2016).

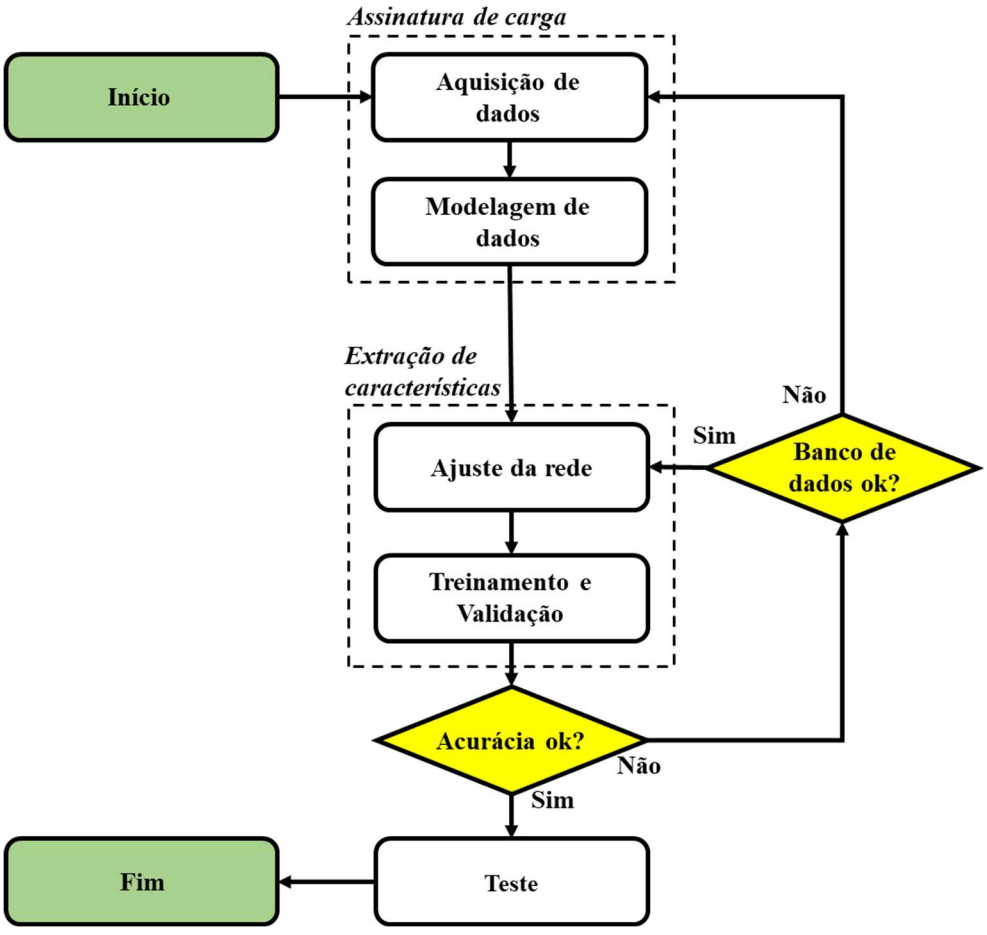
## 1.3 O Problema

Em (FIRMES, 2020), utilizou-se um tipo específico de rede neural artificial (RNA) para abordar o problema de monitoramento não-intrusivo de cargas elétricas altamente similares. O experimento, ilustrado na Figura 5, é constituído de uma metodologia desenvolvida a partir de uma bancada experimental.

A etapa de aquisição de dados (Figura 5) representa a coleta do conjunto de dados através da aferição do comportamento elétrico do sistema. Naquela etapa foi utilizado um equipamento constituído de módulos sensoriais (sensores, amplificadores e filtros), módulo de conversão analógico/digital e armazenamento dos conjuntos de dados adquiridos.

Os dados coletados passam por um processo de extração de características que definem uma assinatura de carga, servindo assim ao propósito de identificação de padrões (LIANG, J. et al., 2010).

Figura 5 – Fluxograma do sistema de monitoramento de cargas.




Fonte: Adaptado de (FIRMES, 2020)

A partir da obtenção de tal conjunto de amostras, as quais contém informações comportamentais das cargas durante o período de monitoramento, é feita a organização e adequação estruturada dos dados, seguido de ajuste dos parâmetros de configuração da rede, para que, enfim, sejam alimentados no sistema de monitoramento inteligente, que é capaz de extrair as informações através de procedimentos de aprendizagem de máquina chamados de treinamento, validação e teste.

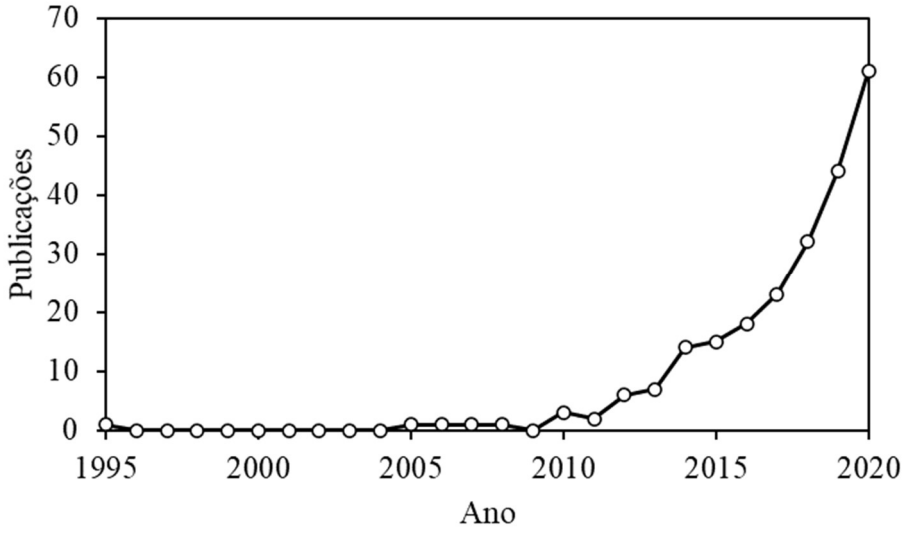
O sistema, uma vez treinado, validado e testado, deve ser capaz de identificar novos exemplos cuja assinatura de carga corresponda a padrões já conhecidos. Enquanto o critério de acurácia não for atingido e, caso o conjunto de dados esteja adequado, repete-se o treinamento, validação e teste com novos ajustes, senão, retoma-se à etapa de aquisição e modelagem dos dados procurando possíveis falhas. O detalhamento concernente a esta sistemática pode ser verificado em (FIRMES, 2020).

Com relação à classificação das cargas elétricas, é relevante reforçar que podem ser constituídas de diferentes graus de similaridade, o que significa que duas cargas podem ter similaridade mínima, ou seja, absolutamente diferentes, ou similaridade máxima, no sentido de serem absolutamente idênticas. Esta similaridade pode ser identificada, por exemplo, a partir da comparação de características extrínsecas, que são as que estão na frequência fundamental de funcionamento ou no nível de corrente contínua, ou com as características intrínsecas, que são as constituídas em componentes harmônicas.

É interessante ressaltar que na literatura ainda não há uma quantidade relevante de trabalhos que dissertem sobre a identificação de cargas com alto grau de similaridade, uma vez que, à medida que esta condição se acentua, as taxas de sucesso tendem a se degradar em sistemas de classificação convencionais, como mostrado em (PAIXÃO, 2016) e (SCHNEIDER, 2018), representando um caminho que demanda pesquisas e o uso de tecnologias que contornem este tipo de obstáculo, como foi feito em (FIRMES, 2020), o qual empregou ferramentas baseadas em *deep learning*.

A Figura 6 apresenta o desenvolvimento de estudos ao longo dos anos sobre o monitoramento de cargas elétricas utilizando ferramentas de aprendizado profundo. A utilização de ferramentas de *machine learning* e *deep learning* vem assumindo um papel de destaque na solução de problemas que exijam o processamento e a classificação de grandes bancos de dados, fornecendo informações estratégicas e controle mais apurado em tempos de resposta cada vez menores.

Figura 6 – Avanço de publicações sobre o monitoramento de cargas utilizando ferramentas de *deep learning*.




Fonte: Web of Science – Clarivate Analytics, 2021

É a partir deste ponto que, de fato, se inicia a problemática deste trabalho, que representa uma continuação da solução proposta em (FIRMES, 2020) para a identificação de cargas elétricas similares utilizando uma ferramenta específica de *deep learning*, as redes neurais convolucionais (CNN).

O uso de redes neurais convolucionais para aplicações no monitoramento NILM possui fundamentação teórica e motivação semelhante às das aplicações de reconhecimento de imagens. Nesse tipo de sistema, a classificação bem sucedida inclui o reconhecimento de padrões sutis no consumo de energia e respectivas flutuações entre seus níveis, que podem ser mascarados em perfis de consumo energético compostos de vários equipamentos eletroeletrônicos, provocando um cenário com alto índice de ruído elétrico/eletromagnético.

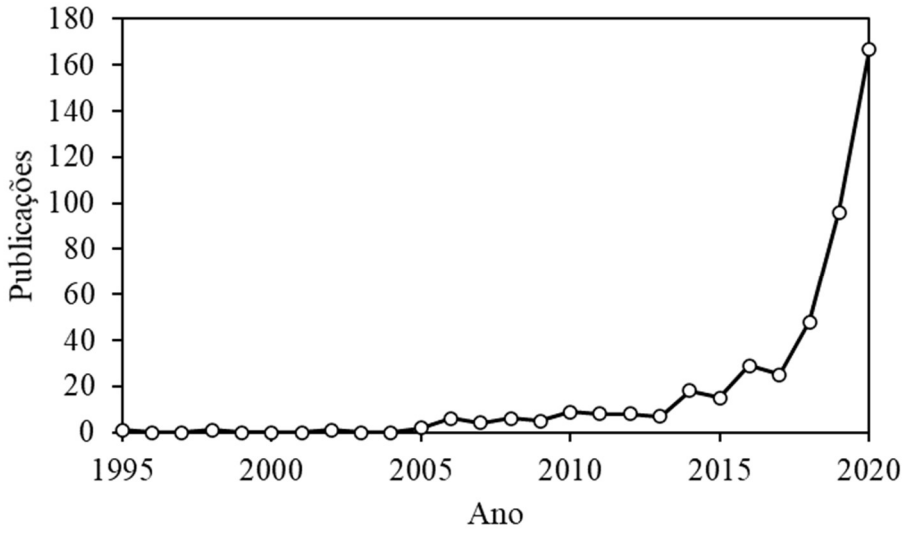
Já foi provado que as CNNs são capazes de identificar imagens específicas, mesmo cerceadas por ruídos diversos, como em (RUSSAKOVSKY et al., 2015), mostrando-se uma ferramenta promissora também na desagregação de cargas elétricas (KONG et al., 2020). Tais modelos são, geralmente, treinados a partir de um método supervisionado e, mais especificamente, de classificação. Além disso, podem utilizar uma quantidade menor de neurônios artificiais se comparadas a redes clássicas como a *Feed foward*. Isso é conveniente para o processamento de dados complexos, entretanto, necessitam de uma quantidade elevada de amostras na fase de treinamento, além de possuir um elevado número de hiperparâmetros que devem ser individualmente ajustados (DARWISH; EZZAT; HASSANIEN, 2020).

O ajuste de hiperparâmetros é fundamental para melhorar o desempenho de quaisquer algoritmos de aprendizado de máquina. Entender como eles funcionam e afetam o desempenho de uma rede neural artificial (RNA) ainda é um problema em aberto, de modo que esforços contínuos são empregados no intento de se encontrar um meio eficiente de ajustá-los, como destacado em (LUJAN-MORENO et al., 2018).

É difícil e custoso treinar muitas arquiteturas diferentes em busca da otimização de um resultado, pois a determinação dos hiperparâmetros a serem ajustados tem sido feita de forma empírica. Outro fator impeditivo consiste na possibilidade de não haver um conjunto de dados suficientemente grande para treinar adequadamente determinada rede, em determinados contextos (SRIVASTAVA et al., 2014).

Logo, a configuração dos hiperparâmetros de RNAs requer o trabalho minucioso de especialistas, enquanto que os não especialistas ficam limitados a utilizar configurações determinadas quase de forma aleatória, a partir de um processo empírico. Esse tipo de metodologia pode consumir tempo considerável durante a fase de construção da arquitetura da rede e, por isso, alguns pesquisadores têm abordado esta temática como um problema de otimização, a qual vem apresentando resultados superiores aos alcançados por especialistas humanos (DARWISH; EZZAT; HASSANIEN, 2020). A Figura 7 ilustra o avanço das pesquisas nesse sentido, que se demonstra como um tema relativamente recente, porém, promissor.

Figura 7 – Avanço de publicações sobre otimização de hiperparâmetros em RNAs.




Fonte: Web of Science – Clarivate Analytics, 2021

Portanto, haja vista a implementação de rede neural convolucional para identificação de cargas elétricas similares anteriormente apresentada, o problema chave deste trabalho se reflete na busca de uma metodologia capaz de identificar as melhores configurações para uma parcela selecionada de hiperparâmetros, de maneira a automatizar este processo, anteriormente realizado de maneira pura e empiricamente manual.

## 1.4 Objetivo

### 1.4.1 Objetivo Geral

Baseado nos fatores motivadores expostos na Seção 1.1, na justificativa apresentada na Seção 1.2, assim como no problema indicado na Seção 1.3, apresenta-se como objetivo geral neste trabalho a automatização do processo de busca de um conjunto ótimo de hiperparâmetros de rede neural convolucional para o problema especificado.

### 1.4.2 Objetivos Específicos

Como objetivos específicos, citam-se os seguintes itens:

- Definir uma arquitetura CNN de referência, baseando-se no trabalho desenvolvido em (FIRMES, 2020);
- Definir uma quantidade ótima de casos para treinar, validar e testar adequadamente a arquitetura CNN de referência;
- Elencar um conjunto de hiperparâmetros normalmente ajustados por especialistas em arquiteturas CNN;
- Aplicar ajuste ótimo dos hiperparâmetros elencados;
- Comparar a arquitetura CNN otimizada com a de referência.

## 1.5 Estruturação

De maneira a organizar logicamente o conteúdo e promover um avanço ordenado até o foco deste trabalho, no Capítulo 2 trata das redes neurais artificiais (RNA), a configuração dos hiperparâmetros de RNA e os detalhes das redes neurais convolucionais. No Capítulo 3,

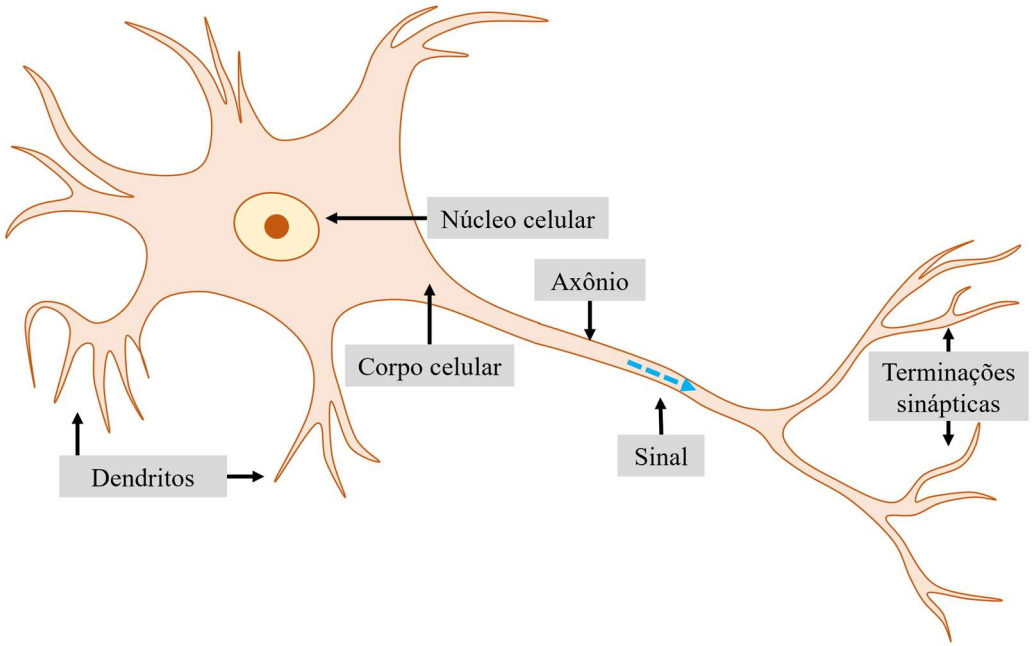
apresenta-se o referencial teórico sobre a aplicação de otimização heurística como ferramenta de ajuste para os hiperparâmetros. O Capítulo 4 descreve o detalhamento da metodologia utilizada, seguido do Capítulo 5, que apresenta os resultados dos experimentos e desenvolve as discussões pertinentes. Finalmente, no Capítulo 6, é desenvolvido o raciocínio de conclusão da investigação que resultou neste trabalho.

# 2 Redes Neurais Artificiais

## 2.1 Introdução

O neurônio é uma célula biológica de uso específico, capaz de processar informações através da condução de estímulos elétricos advindos de reações físico-químicas. É composto, conforme a Figura 8, pelo corpo celular e dois tipos de terminais nervosos (espécie de ramificação), os dendritos (ou soma) e o axônio. O corpo celular possui um núcleo, que carrega suas informações hereditárias, e uma espécie de plasma, necessário para produzir seu material constituinte. Esta célula nervosa recebe os impulsos de outras células por meio de seus dendritos, que funcionam como elementos receptores e transmite os sinais através de sua estrutura, por meio do axônio, que se ramifica em terminações sinápticas. A sinapse representa a estrutura fundamental de comunicação entre dois neurônios, e está sujeita à influência de substâncias denominadas neurotransmissores, que podem estimulá-las ou inibi-las, a depender de ajustes específicos de ordem biológica (JAIN; JIANCHANG MAO; MOHIUDDIN, 1996).

Figura 8 – Esquema de um neurônio biológico.



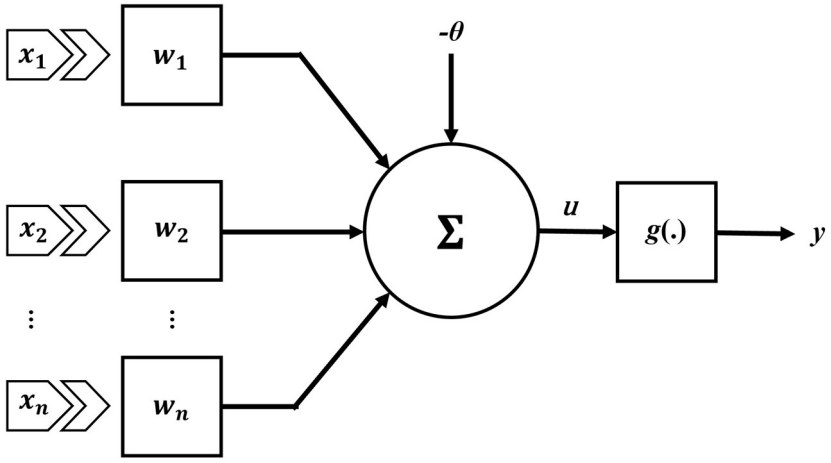

Fonte: Adaptado de (BASHEER; HAJMEER, 2000)

Nesta linha de raciocínio, foi desenvolvida uma estrutura de rede neural artificial baseada nos aspectos biológicos do sistema nervoso e do cérebro humano, de maneira que esses elementos computacionais representam uma réplica simplificada dos neurônios biológicos. O

modelo de neurônio mais simples, inicialmente proposto por (MCCULLOCH; PITTS, 1943), desempenha estas principais funções biológicas, o paralelismo e alta conectividade.

Conforme ilustra a Figura 9, os variados sinais de entrada são denominados pelo conjunto $\{x_1, x_2, x_3, \dots, x_n\}$, semelhantes aos impulsos externos recebidos pelos dendritos. Já as ponderações executadas pelas sinapses são representadas pelo conjunto de pesos sinápticos $\{w_1, w_2, w_3, \dots, w_n\}$. Portanto, a magnitude de cada um dos impulsos é dada através do produto com os pesos sinápticos, o que torna possível constatar que a saída da célula nervosa computacional, denominada $u$, é a soma ponderada de suas entradas.

Figura 9 – Neurônio artificial.




Fonte: Adaptado de (SILVA, I. N. da; SPATTI; FLAUZINO, 2016)

Em (SILVA, I. N. da; SPATTI; FLAUZINO, 2016) é possível averiguar que um neurônio artificial se estrutura, matematicamente, através da Equação 2.1 e Equação 2.2, composta de sete elementos básicos, dispostos a seguir:

$$u = \sum_{i=1}^{n} w_i . x_i - \theta \qquad (2.1)$$

$$y = g(u) \qquad (2.2)$$

a) Sinais de entrada $\{x_1, x_2, x_3, \dots, x_n\}$, anteriormente mencionados. Cabe salientar que são comumente normalizados, com o objetivo de potencializar o processamento computacional nos algoritmos de aprendizado;

b) Pesos sinápticos $\{w_1, w_2, w_3, \dots, w_n\}$, anteriormente mencionados;

c) Combinador linear {Σ}, cuja funcionalidade é agregar os sinais de entrada, previamente ponderados, com o objetivo de resultar um valor potencial de ativação;

d) Limiar de ativação {θ}, que compreende uma variável cujo papel é especificar qual será o nível adequado para que o combinador linear seja capaz de fornecer um valor de desencadeamento em direção à saída do neurônio artificial;

e) Potencial de ativação {$u$}, o qual representa o resultado obtido pela diferença do valor produzido pelo combinador linear e pelo limiar de ativação. Caso u ≥ θ, o neurônio artificial fornece um potencial excitatório; caso contrário, tem-se um potencial inibitório;

f) Função de ativação {$g$}, cujo papel é restringir a saída do neurônio a partir de um intervalo de valores estabelecidos;

g) Sinal de saída {$y$}, que compreende o valor final calculado pelo neurônio em associação a um determinado conjunto de impulsos, sendo, inclusive, possível utilizá-lo como sinal de entrada num processo de encadeamento de neurônios interligados.

As funções de ativação não-lineares inseridas por cada neurônio têm fundamental importância na manutenção de um aprendizado eficaz. Sua omissão, mesmo em redes neurais profundas, resulta em regressões lineares simples e, por esta razão, são essenciais no trabalho de tornar os modelos computacionais mais representativos.

Durante anos, funções como do tipo sigmóide (logística) e tangente hiperbólica (tanh) foram popularmente utilizadas. No entanto, em tempos mais atuais, as funções do tipo *Rectified Linear Units* (ReLUs), assim como, a *leaky* ReLUs, *randomized* ReLUs e *parametric* ReLUs têm sido frequentemente implementadas. É possível, ainda, citar outras funções de ativação amplamente utilizadas, como as do tipo *Exponential Linear Units* (ELUs) e *Scaled* ELUs (SELUs) (GODFREY, 2018).

Apresenta-se na Tabela 1, de forma mais objetiva e comparativa, as principais funções de ativação utilizadas em redes neurais artificiais. As funções sigmóide e linear possuem saída com valor 0 ou 1, onde o fator $\alpha$ permite um comportamento de variadas inclinações na primeira função. Ainda que outras funções também possam ser utilizadas, nos casos onde se deseja uma saída entre -1 e 1, comumente se utiliza a função tangente hiperbólica (tanh). Nas situações onde se emprega redes neurais profundas, recomenda-se as funções do tipo ReLU, PReLU, ELU e *Softplus*. A função de ativação *Softmax* pode ser implementada quando se deseja forçar

a saída da rede à probabilidade de os dados de entrada pertencerem a uma das classes, e pode ser descrita através da Equação 2.3, que ilustra a forma como a função identifica cada classe:

$$P_c = \sigma\big(S_c(x)\big) = \frac{\exp\,(S_c(x))}{\sum_{j=1}^{C} \exp\,(S_j(x))} \qquad (2.3)$$

onde $S_c(x)$ é um vetor com as pontuações de cada classe $c$ a instância $x$; $C$ é a quantidade de classes envolvidas; e $P_c$ é a probabilidade de $x$ pertencer à classe $c$ (GÉRON, 2017).

Tabela 1 – Funções de ativação.

| **Nome** | **Função** |
|---|---|
| Degrau (*Heaviside*) | $f(u) = \begin{cases} 1\ se\ u \geq 0 \\ 0\ se\ u < 0 \end{cases}$ |
| Linear | $f(u) = \begin{cases} 1\ se & u \geq {}^1/_2 \\ u\ se & -{}^1/_2 < u < {}^1/_2 \\ 0\ se & u < {}^1/_2 \end{cases}$ |
| Sigmóide (logística) | $f(u) = \sigma(u) = (1 + \exp\,(-\alpha u))^{-1}$ |
| Tangente hiperbólica | $f(u) = \tanh\,(u)$ |
| Arco tangente | $f(u) = \tan^{-1}\,(u)$ |
| *Rectified Linear Unit* (ReLU) | $f(u) = \begin{cases} u\ se\ u \geq 0 \\ 0\ se\ u < 0 \end{cases}$ |
| *Parametric Rectified Linear Unit* (PReLU) | $f(u) = \begin{cases} u\ se & u \geq 0 \\ \alpha u\ se\ u < 0 \end{cases}$ |
| *Exponential Linear Unit* (ELU) | $f(u) = \begin{cases} u\ se & u \geq 0 \\ \alpha(e^u - 1)\ se\ u < 0 \end{cases}$ |
| *Softplus* | $f(u) = \log\,(1 + \exp(u))$ |

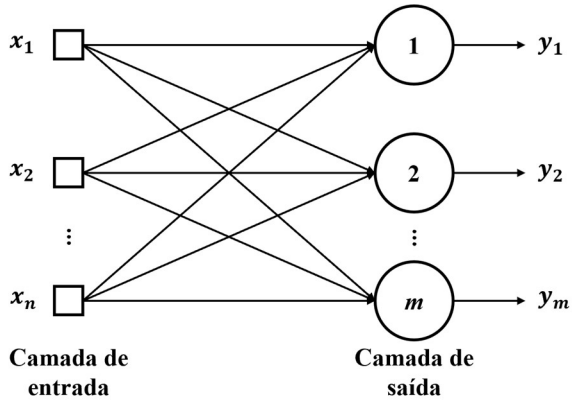
Fonte: (SILVA, R. D., 2018)

Partindo de uma visão simplificada, é possível descrever uma rede neural artificial em três principais elementos: camada de entrada, camadas escondidas (intermediárias, ocultas ou invisíveis) e camada de saída. A camada de entrada é o elemento responsável pelo recebimento dos dados, que representam os sinais ou medições advindas da coleta de amostras e padrões adquiridos do meio externo, as quais são geralmente normalizadas, conforme descrito anteriormente. As camadas ocultas são constituídas de neurônios com a responsabilidade de extrair atributos relativos ao processo implementado. Cabe ressaltar que quase todo o processamento interno da rede é desenvolvido nessas camadas. Já a camada de saída, também

composta por neurônios, é encarregada pelo processamento e apresentação dos resultados finais da rede, como consequência do processamento nas camadas ocultas (SILVA, I. N. da; SPATTI; FLAUZINO, 2016).

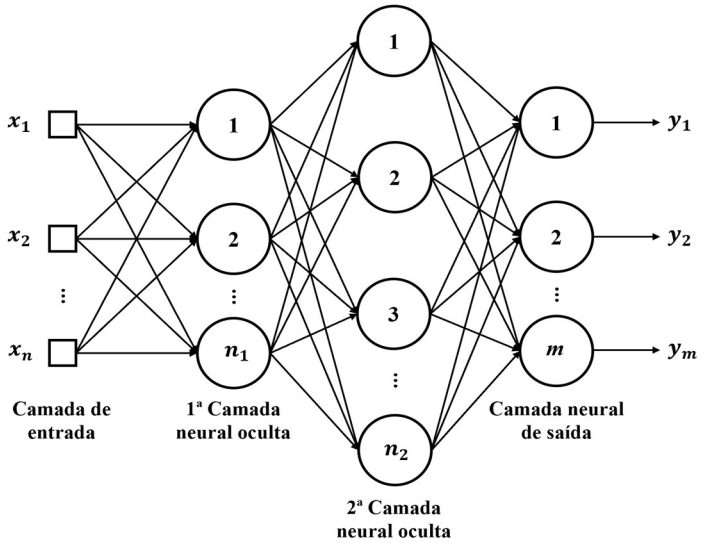
As configurações, ou tipos, de redes neurais são, basicamente, as diferenças na arquitetura estabelecida a partir das várias formas de arranjo e comunicação dos neurônios. A Figura 10 ilustra um modelo de rede neural, denominada *feed foward* (alimentação à frente), com apenas uma camada com *m* neurônios, conectada a *n* entradas. Observe que a própria camada é a saída. A Figura 11 também ilustra uma rede *feed forward*, porém com arquitetura multicamada (duas camadas ocultas com $n_1$ e $n_2$ neurônios, respectivamente).

Figura 10 – Esquema de uma rede tipo *feed foward* com camada única.




Fonte: Adaptado de (SILVA, I. N. da; SPATTI; FLAUZINO, 2016)

Figura 11 – Esquema de uma rede tipo *feed foward* multicamada.




Fonte: Adaptado de (SILVA, I. N. da; SPATTI; FLAUZINO, 2016)

## 2.2 Processo de Aprendizado em Redes Neurais Artificiais

À medida que a complexidade dos dados se acentua, o layout da arquitetura das redes neurais deve se adaptar. Uma característica capaz de lidar satisfatoriamente com o caso de condições variantes, isto é, que exigem o dinamismo do processamento neural, é o procedimento de treinamento, ou aprendizado.

Este recurso consiste na implementação sistematizada de operações estatísticas, capazes de sintonizar os pesos sinápticos e limiares dos neurônios, possibilitando a generalização de soluções. Durante este processo, cada apresentação completa dos dados do conjunto de treinamento é denominada época de treinamento.

Além disto, é possível classificá-lo em, basicamente, quatro tipos de treinamento: o supervisionado, quando o algoritmo de aprendizagem tenta prever uma variável dependente a partir de variáveis independentes; não supervisionado, através da inferência, de forma a identificar possíveis padrões ou características em dados não necessariamente rotulados; lote de padrões (*offline*), onde os ajustes dos pesos e limiares são realizados somente após apresentação de todo o conjunto de treinamento; e padrão-por-padrão (*online*) que, ao contrário do *offline*, implementa os ajustes após a apresentação de cada amostra do conjunto de treinamento (SILVA, I. N. da; SPATTI; FLAUZINO, 2016).

Investigado e proposto em (ROSENBLATT, 1958), a *Perceptron* (Figura 12) é considerada a arquitetura mais simples de uma rede neural artificial, e possui a mesma razão de funcionamento supracitada, isto é, em função de um processo de treinamento. Apesar de sua simplicidade, a *Perceptron* funciona seguindo o mesmo princípio de um neurônio biológico, que fortalece suas conexões de forma proporcional à frequência de desencadeamento com outro neurônio (HEBB, 1949).

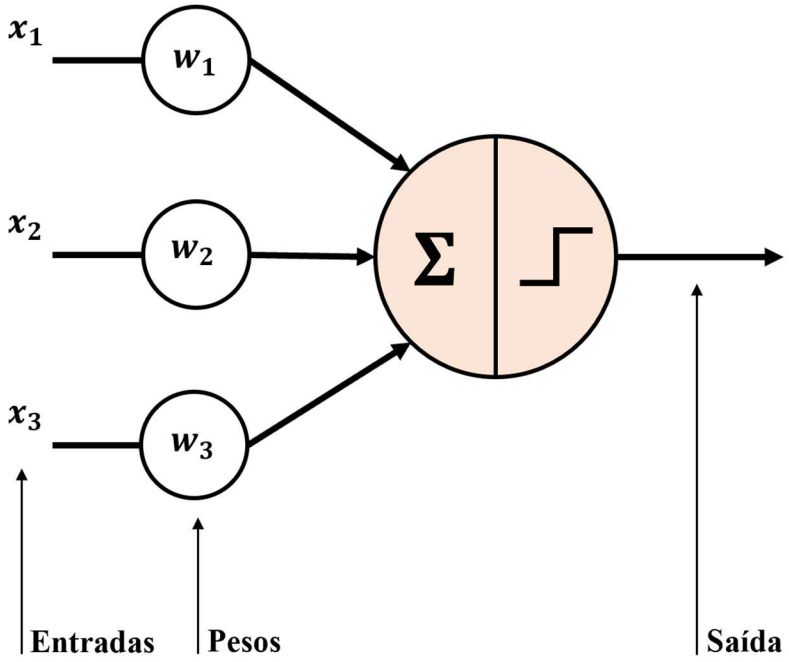
De acordo com (GÉRON, 2017), neste tipo de rede, para cada neurônio de saída que produziu uma previsão incorreta, serão reforçados os pesos de conexão das entradas que participaram para a previsão correta. Isto é matematicamente formulado como:

$$w_{i,j}{}^{(próximo\ degrau)} = w_{i,j} + \eta(y_j - \hat{y}_j)x_i \qquad (2.4)$$

onde $w_{i,j}$ é o peso de conexão entre o i-*ésimo* neurônio de saída; $x_i$ é o valor da i-*ésima* entrada da instância de treinamento atual, $\hat{y}_j$ é a i-*ésima* saída do neurônio de saída para a instância de

treinamento atual, $y_j$ é a i-*ésima* saída alvo do neurônio de saída para a instância de treinamento atual e $\eta$ é a taxa de aprendizado.

Figura 12 – Representação esquemática de um *Perceptron*.



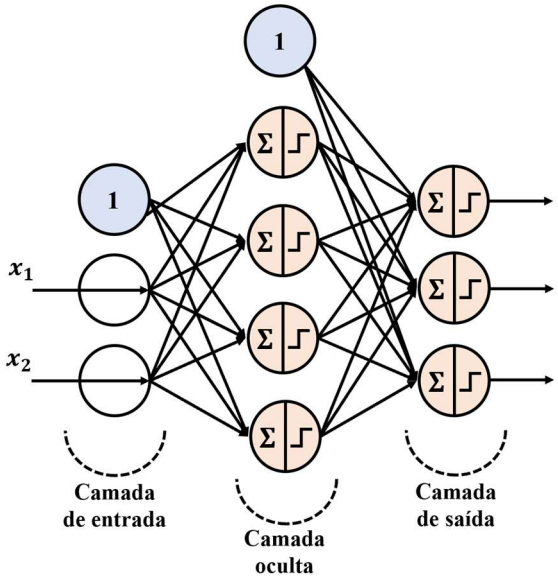

Fonte: Adaptado de (GÉRON, 2017)

Em (SILVA, I. N. da; SPATTI; FLAUZINO, 2016) complementa-se que este tipo de rede se comporta como um classificador de padrões cuja função é unicamente dividir classes que sejam linearmente separáveis.

Outro tipo de rede relevante na literatura é a *Multilayer Perceptron* (MLP) (Figura 13), que é composta por uma camada de entrada, uma ou mais camadas do tipo *Linear Threshold Units* (LTUs) ocultas e uma última camada de saída LTU. Todas as camadas, com exceção da camada de saída, possuem um neurônio de viés (*bias*) e estão plenamente conectadas à camada seguinte. Nas arquiteturas neurais onde há a presença de uma ou mais camadas ocultas, emprega-se a designação de rede neural profunda (DNN).

Referente ao treinamento de uma rede neural tipo *Perceptron* multicamada, durante anos encontrou-se dificuldade em estabelecer alguma técnica com sucesso. Entretanto, em (RUMELHART; HINTON; WILLIAMS, 1986), foi concluído e apresentado um algoritmo de treinamento a partir de uma técnica denominada retro propagação.

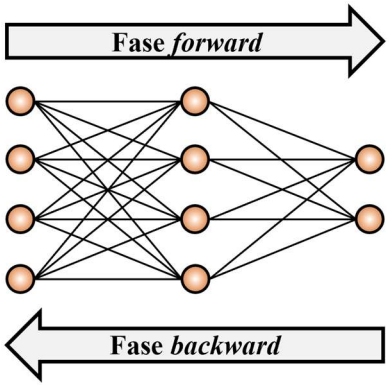
Figura 13 – Representação esquemática de um *Perceptron* Multicamada.




Fonte: Adaptado de (GÉRON, 2017)

Conforme discutido em (FURTADO, 2019), retro propagação (*back propagation*) é um algoritmo de treinamento, do tipo supervisionado, para redes neurais *feedfoward* que utilizem funções de ativação de classe derivável. Esta ferramenta permite alterar, aos poucos, os valores das sinapses, de modo a otimizar a saída da rede. Em (BRAGA; LUDERMIR; CARVALHO, 2000) é esclarecido que este tipo de treinamento ocorre em duas etapas distintas (Figura 14) denominadas *foward* e *backward*, representando os sentidos em que ocorre a propagação. Durante a primeira etapa, utiliza-se um valor de entrada para se definir a saída da rede e, na etapa seguinte, utilizam-se os erros no sentido oposto, de maneira a atualizar os pesos das conexões.

Figura 14 – Conceito de *Backpropagation*.




Fonte: Adaptado de (BRAGA; LUDERMIR; CARVALHO, 2000)

A partir da Equação 2.5, como desenvolvido em (HAYKIN, 2001), pode-se representar o sinal de erro na saída do neurônio *j*, na iteração *n* (o n-*ésimo* exemplo de treinamento apresentado).

$$e_j(n) = d_j(n) - y_j(n), \tag{2.5}$$

onde $d_j(n)$ corresponde ao valor desejado para a saída $y_j(n)$ referente ao neurônio *j* da iteração *n*. Prosseguindo, pode-se definir o valor instantâneo da energia do erro para o neurônio *j* como $\frac{1}{2}{e_j}^2(n)$, vide Equação 2.6. Portanto, obtém-se o valor instantâneo $\xi(n)$ da energia total do erro dada por

$$\xi(n) = \frac{1}{2}\sum_{j\in C} {e_j}^2(n), \tag{2.6}$$

em que *C* representa o conjunto que inclui os neurônios da camada de saída. Portanto, considerando *N* a quantidade total de padrões no conjunto de treinamento, a energia média do erro quadrático pode ser obtida a partir de

$$\xi_{med} = \frac{1}{N}\sum_{n=1}^{N} \xi(n). \tag{2.7}$$

Conforme (HAYKIN, 2001), ajustando os parâmetros (pesos e *bias*) para minimizar $\xi_{med}$, que é comumente denominada função de custo, ou perda. Por conseguinte, a implementação da correção $\Delta w_{ji}(n)$ no peso $w_{ji}(n)$, durante a iteração *n* é expressa como

$$\Delta w_{ji}(n) = \eta \delta_j(n) y_i(n). \tag{2.8}$$

Em seguida, é implementada a retropropagação a partir da derivação de gradientes locais da rede $(\delta s)$, definida como

$$\delta_j{}^{(l)}(n) = \begin{cases} e_j{}^{(l)}(n){\varphi'}_j\left(v_j{}^{(L)}(n)\right) & neurônio\ j\ na\ camada\ de\ saída\ L \\ {\varphi'}_j\left(v_j{}^{(l)}(n)\right)^{\prime \sum_k \delta_k{}^{(l+1)}(n) w_{kj}{}^{(l+1)}(n)} & neurônio\ j\ na\ camada\ oculta\ l \end{cases} \tag{2.9}$$

onde *k* representa os neurônios da camada de saída e ${\varphi'}_j$ representa a derivada em relação ao argumento. Por conseguinte, a alteração dos pesos sinápticos na camada *l* é dada por

$$w_{ji}{}^{(l)}(n+1) = w_{ji}{}^{(l)}(n) + \Delta_{ji}{}^{(l)}(n). \qquad (2.10)$$

Utilizando a correção

$$w_{ji}{}^{(l)}(n) = \eta\delta_j{}^{(l)}(n)y_i{}^{(l-1)} + \alpha\Delta_{ji}{}^{(l)}(n-1), \qquad (2.11)$$

evita-se a convergência da função para um mínimo local, em que $\eta$ representa a taxa de aprendizado do algoritmo e α representa o momento (HAYKIN, 2001).

A taxa de aprendizagem possui papel fundamental nos resultados de uma rede neural artificial. Caso essa taxa seja configurada com um valor muito alto, a função perda irá subir, causando distúrbios na rede. No entanto, se este valor for estabelecido para abaixo do ideal, será possível apresentar demora ou até mesmo não alcançar a convergência para uma solução.

Até o momento, não há um método que generalize, de forma detalhada, a obtenção do valor ótimo para a taxa de aprendizado em todos os casos. Entretanto, várias pesquisas têm abordado o assunto através de métodos heurísticos com resultados promissores (BENGIO, 2012).

## 2.3 Redes Neurais Convolucionais

As redes neurais convolucionais (CNNs) têm sido implementadas em processos de reconhecimento de imagens desde os anos 1980 e, recentemente, devido ao aumento do poder computacional e da grande quantidade de dados disponíveis para treinamento, é possível obter poder de processamento sobre-humano em determinadas tarefas consideradas complexas. Segundo a revisão desenvolvida em (SINGH; MITTAL; BHATIA, 2018), este tipo de tecnologia tem conquistado bons resultados nas mais diversas áreas, como processamento de imagens, visão computacional, processamento de texto, de sinais e de áudio.

O fundamento teórico por trás deste tipo de rede baseia-se na teoria da neurociência visual, publicada em (HUBEL; WIESEL, 1962), a qual afirma que ambos os tipos de células, simples e complexas, foram encontrados no córtex visual de um gato. As células simples disparam em resposta a algumas propriedades de entradas sensoriais, enquanto que as complexas exibem mais invariância espacial.

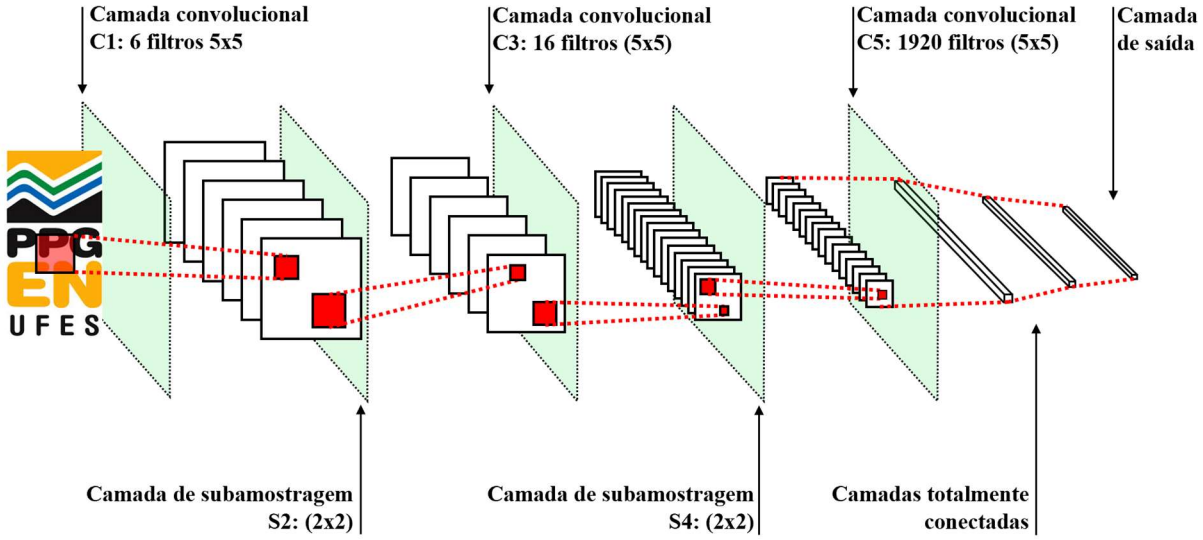
Segundo registros bibliométricos, acredita-se que o *Neocognitron* (FUKUSHIMA, 1980) foi a primeira rede desenvolvida a utilizar uma camada de convolução (SINGH; MITTAL;

BHATIA, 2018). Anos mais tarde, conforme (LECUN et al., 1998), foi proposta uma arquitetura de rede neural convolucional para classificação de caracteres manuscritos e impressos à máquina nos anos 1990, denominada LeNet-5.

A camada convolucional é composta de diversos filtros convolutivos, que são utilizados para diferentes mapas de características (*feature maps*). Cada neurônio de um mapa de característica é conectado a uma região de neurônios vizinhos, enquanto que cada vizinhança está conectada a um campo receptivo de um neurônio da camada anterior. Um *feature map* pode ser obtido através de uma operação de convolução entre a matriz de entrada e um filtro linear (GU, J. et al., 2017).

A Figura 15 ilustra um modelo de arquitetura de rede neural convolucional baseado nos parâmetros desenvolvidos em (LECUN et al., 1998).

Figura 15 – Esquema de rede neural convolucional baseada na LeNet-5 *network*.




Fonte: Adaptado de (LECUN et al., 1998)

## 2.3.1 Camada de Convolução

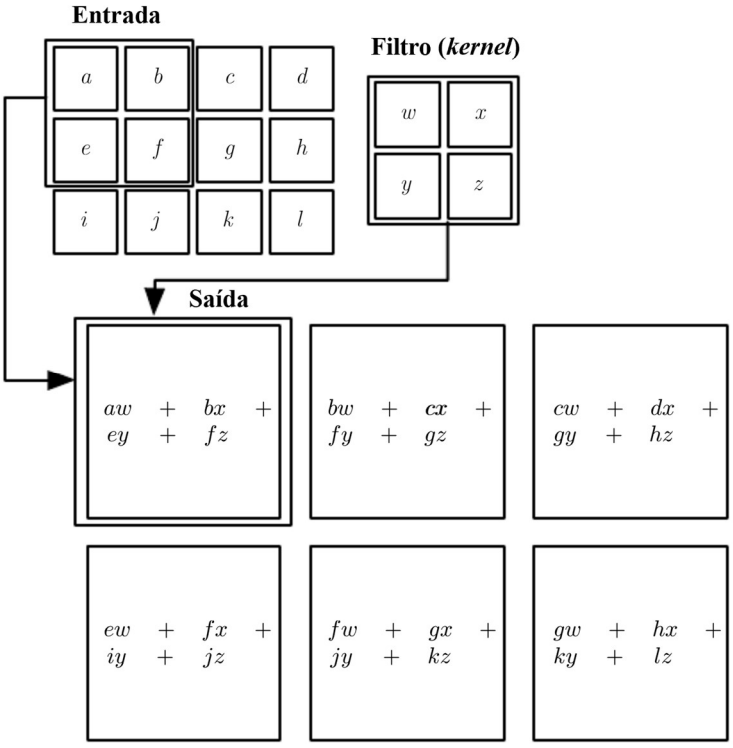
Partindo de uma abordagem mais prática, é possível descrever que a convolução utiliza três aspectos importantes, os quais auxiliam no aprendizado de máquina eficiente: interações esparsas, compartilhamento de parâmetros e representações equivalentes.

O primeiro aspecto é alcançado através da implementação de um filtro (*kernel*) de tamanho menor que a entrada, reduzindo os dados para um conjunto mais significativo (e nem por isso, menos representativo), reduzindo os requerimentos de memória e incrementando a eficiência estatística. O segundo aspecto se refere ao uso do mesmo parâmetro para mais de

uma função no modelo, o que significa dizer que a rede possui pesos conectados, onde cada elemento do filtro é utilizado em todas as posições da entrada, possibilitando o aprendizado em apenas um conjunto. Já o terceiro aspecto possibilita, matematicamente, que a saída represente fielmente as características da entrada, o que a torna eficiente no reconhecimento de imagens (GOODFELLOW; BENGIO; COURVILLE, 2016).

Entre os hiperparâmetros, destacam-se como exemplos o *stride* (passo), que representa o tamanho do deslocamento do filtro a cada passo e o *padding*, que possui a função de preencher as bordas da matriz de entrada com algum elemento (FARIA, 2018). Ambos os hiperparâmetros ajudam a controlar as operações de convolução realizadas durante o treinamento da CNN. A Figura 16 representa o procedimento de convolução (2D) para uma matriz de entrada tamanho 3x4, com aplicação de filtro (*kernel*) tamanho 2x2, passo (*stride*) igual a 1 e sem *padding*.

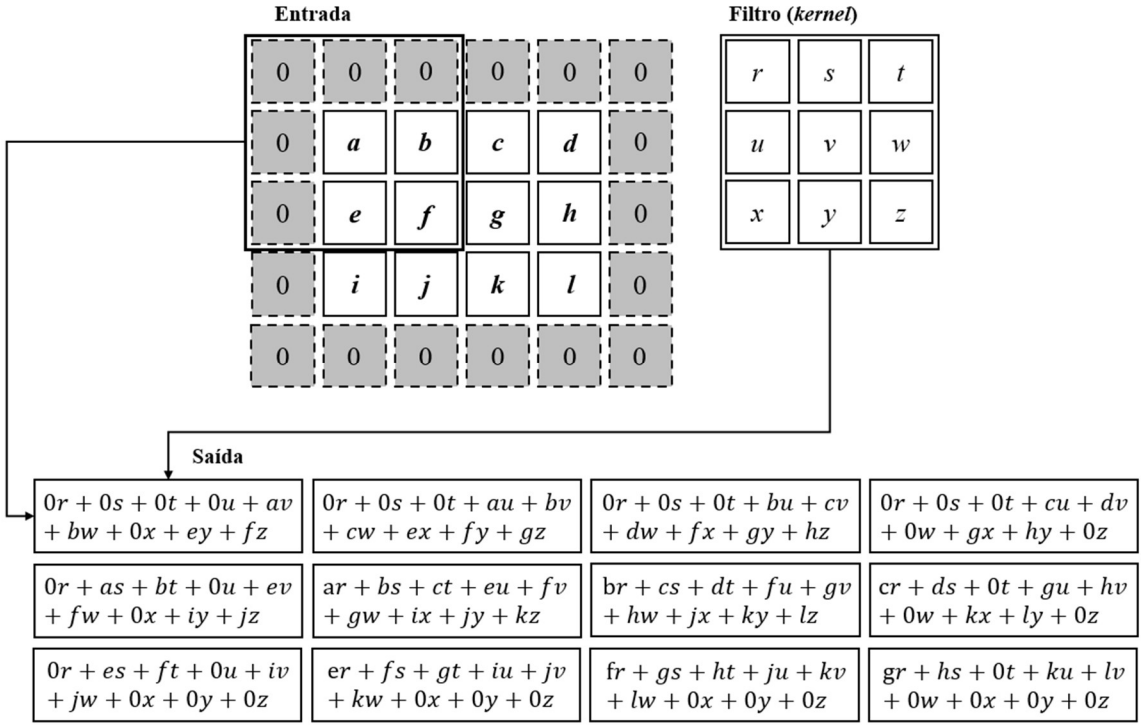
Figura 16 – Exemplo de convolução em duas dimensões.




Fonte: Adaptado de (GOODFELLOW; BENGIO; COURVILLE, 2016)

A Figura 17 representa um procedimento de convolução semelhante, desta vez, com *kernel* 3x3 e aplicação de *zero padding*, isto é, o acréscimo de zeros ao redor da matriz, possibilitando preservar a dimensionalidade da informação.

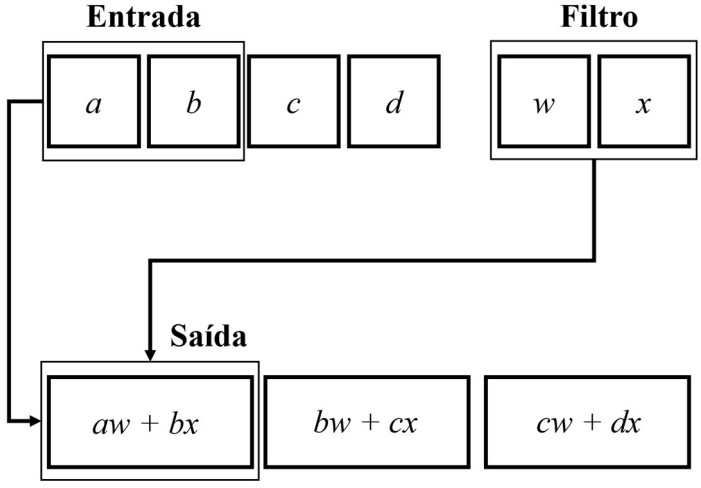
Figura 17 – Exemplo de convolução em duas dimensões (com *padding*).




Fonte: (Autor, 2021)

No caso de convolução em apenas uma dimensão, cabe destacar que o filtro se desloca em apenas uma direção, isto é, da esquerda para a direita, conforme esquematizado na Figura 18.

Figura 18 – Exemplo de convolução em uma dimensão.




Fonte: (Autor, 2021)

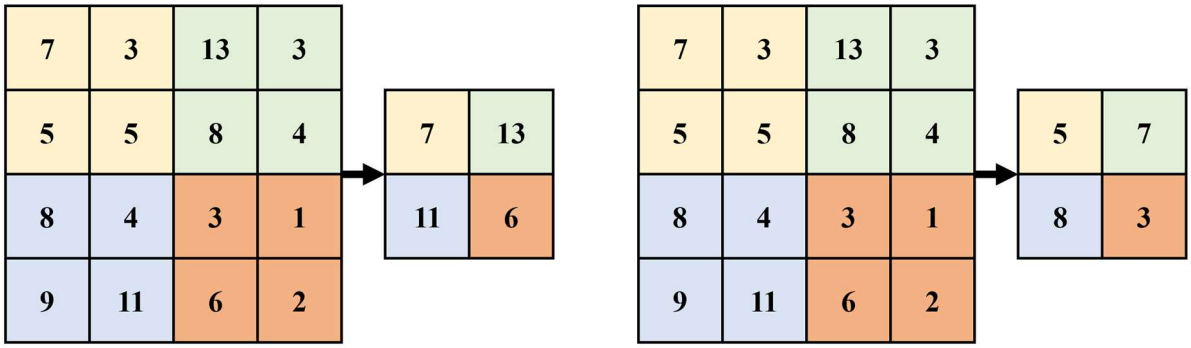
### 2.3.2 Camada de *Pooling*

A camada de *pooling* tem como objetivo reduzir a imagem ou os dados de entrada para reduzir o esforço computacional, o uso de memória e o risco de sobreajuste (*overfitting*). Cada neurônio é, então, conectado às saídas de uma quantidade restrita de neurônios da camada anterior, localizados no respectivo campo receptivo. Nesta etapa, não há a aplicação de pesos, ou seja, os neurônios possuem função exclusivamente agregadora, geralmente selecionando, dentro de cada campo receptivo, os elementos com o valor máximo (*max-pooling*) ou calculando a média destes (*avarage-pooling*). Esta camada de subamostragem funciona de forma independente em todos os canais de entrada, de modo que a profundidade de saída será a mesma da entrada.

É relevante destacar que este tipo de camada representa aspecto destrutivo aos dados, pois há um relativo descarte de informações em detrimento das operações realizadas pelo *kernel*, *stride* e *padding*, respectivamente (GÉRON, 2017). A Figura 19 apresenta um esquema com os dois tipos usuais de *pooling* supracitados, *max-pooling* e *avarage-pooling*, respectivamente.

Cabe ressaltar que as arquiteturas de CNNs geralmente possuem um encadeamento entre camadas convolucionais (geralmente com função de ativação tipo ReLU) e de *pooling*, finalizando com uma rede *feed foward* totalmente conectada e uma camada final para saída dos dados (geralmente com ativação *Softmax*), conforme ilustrado na Figura 20.

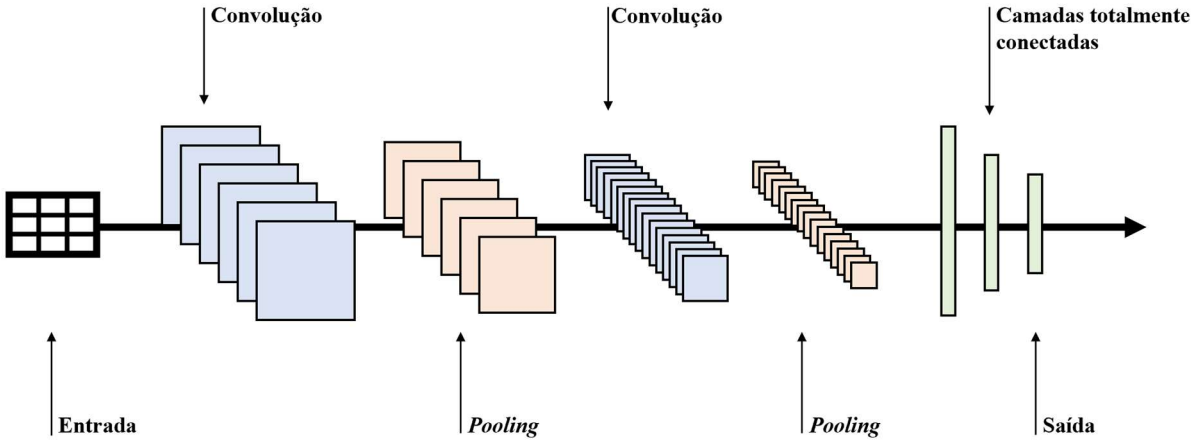
Figura 19 – Tipos usuais de *pooling*.




**(a) *Max-Pooling* (2x2)** **(b) *Avarage-Pooling* (2x2)**

Fonte: (Autor, 2021)

Figura 20 – Arquitetura CNN típica.

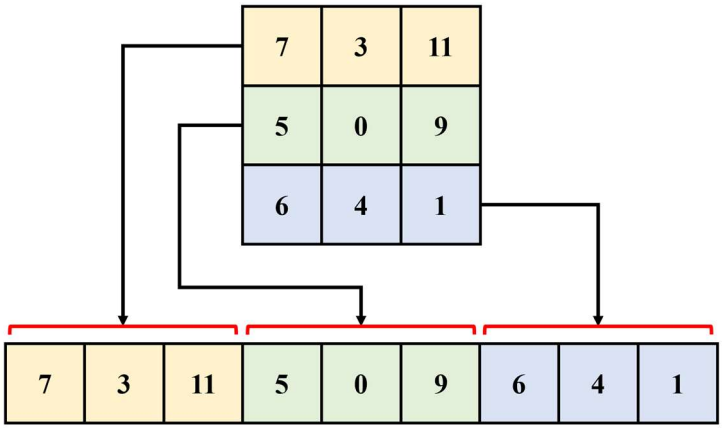



Fonte: Adaptado de (GÉRON, 2017)

### 2.3.3 Camada de Achatamento - *Flatten*

Este tipo de camada (Figura 21) não aprende nenhum parâmetro durante o processo de treinamento e é responsável pelo remodelamento, ou achatamento, dos dados de entrada. Isto acontece a partir do colapso de uma matriz bidimensional de características (que possui mais de uma dimensão espacial) em um vetor que pode ser alimentado em um classificador de rede neural totalmente conectado, isto é, os dados aprendidos nas camadas convolucionais são preparados para serem propagados para as camadas totalmente conectadas, que representam a saída da CNN (MATHWORKS, 2021).

Figura 21 – Esquema de achatamento (*Flatten*).



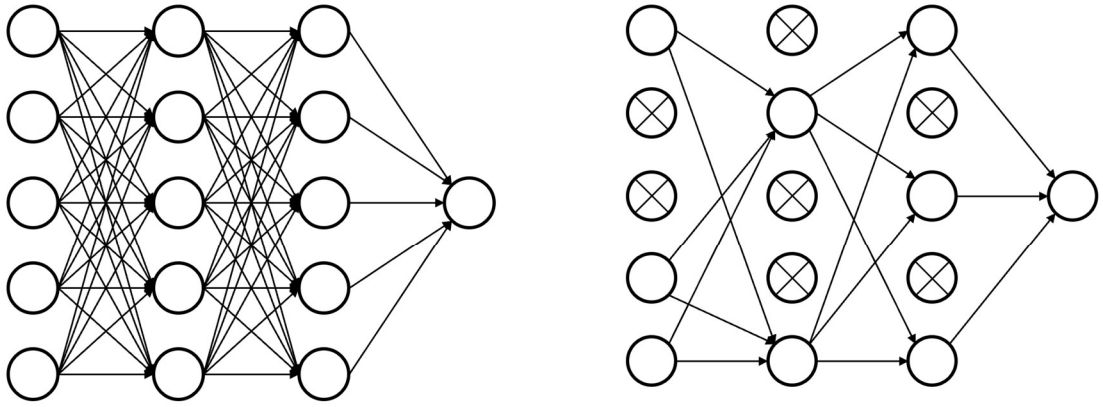

Fonte: (Autor, 2021)

### 2.3.4 Camada Totalmente Conectada

Em uma camada totalmente conectada (*fully connected*) todos os seus neurônios estão completamente conectados com a camada anterior. Este tipo de arranjo é usualmente empregado em arquiteturas de classificação, baseadas na função de ativação *Softmax*, o que reduz a entropia cruzada entre distribuições estimadas (SILVA, R. D., 2018).

Complementando o conceito de camada totalmente conectada, é importante ressaltar uma técnica de regularização comumente utilizada para evitar a condição de *overfitting*, o *dropout* (Figura 22), que é o "congelamento" temporário e aleatório de neurônios, e suas respectivas conexões (Figura 22b), ainda durante a fase de treinamento (GU, J. et al., 2017). Introduzida em (HINTON et al., 2012) e (SRIVASTAVA et al., 2014), foi possível averiguar sua eficiência em impedir que a rede fique muito dependente de quaisquer combinações de neurônios, forçando sua acurácia, mesmo na ausência de determinados conjuntos de informações.

Figura 22 – Arquitetura de rede neural com e sem *dropout*.



**(a) Rede neural padrão** **(b) Após aplicar *dropout***

Fonte: Adaptado de (SRIVASTAVA et al., 2014)

## 2.4 Configuração de Hiperparâmetros

Hiperparâmetro é o termo designado aos parâmetros ajustáveis que controlam o processo de treinamento, os quais são previamente configurados pelo desenvolvedor para que o algoritmo apresente o desempenho almejado. Esse processo de ajuste é normalmente manual e baseado no método de tentativa e erro, contando com a experiência do desenvolvedor para encontrar o ajuste adequado em um espaço de busca geralmente amplo (MICROSOFT, 2021).

Um hiperparâmetro pode estar em diferentes domínios, como dos valores reais (taxa de aprendizado, por exemplo), dos valores inteiros (número de camadas da rede neural, por exemplo), binário (a implementação de parada prematura ou não, por exemplo) ou categórico (escolha de um otimizador, por exemplo). Além disso, sua configuração pode estar sujeita ao estado de condicionalidade, isto é, um hiperparâmetro pode ou não se tornar preponderante apenas se outro hiperparâmetro (ou alguma combinação de hiperparâmetros) assumir um determinado valor, o que torna ainda mais desafiadora a proposta de otimização de uma rede neural artificial (FEURER; HUTTER, 2019).

Qualquer ferramenta de aprendizado de máquina possui hiperparâmetros e, no que diz respeito à automatização do processo de aprendizado de máquina, procura-se estabelecer uma técnica que indique os ajustes mais adequados no sentido de otimizar o desempenho do algoritmo. Um levantamento das principais técnicas de hiperajuste, implementadas a partir de otimização metaheurística, pode ser consultado no Apêndice A.

No caso de redes neurais convolucionais, existem no mínimo quinze hiperparâmetros diferentes para ajuste somente na camada de convolução (KERAS, 2021a), o que leva a deduzir que sua otimização possibilita reduzir o esforço humano durante a modelagem da arquitetura geral, elevar o desempenho do sistema de identificação autônomo e melhorar a reprodutibilidade de estudos científicos através da disponibilidade de configurações padronizadas que permitam a comparação justa de diferentes arquiteturas de redes pré-ajustadas, entre outros aspectos.

Ademais, percebe-se a intensificação de aplicações envolvendo aprendizado de máquina em ambiente corporativo, o que culmina no interesse substancial de técnicas que facilitem a otimização de ajustes em modelos computacionais, seja em ferramentas internas de uma companhia, ou como um serviço ofertado (FEURER; HUTTER, 2019).

# 3 Otimização de Redes Neurais Artificiais

A otimização de algoritmos de aprendizado de máquina e, consequentemente, a preocupação com a configuração de seus parâmetros, inicia-se na década de 90 e pode ser evidenciada em (RIPLEY, 1993), (KING; FENG; SUTHERLAND, 1995), (KOHAVI; JOHN, 1995) e (MICHIE et al., 1995). Ainda no início das pesquisas foi possível constatar que diferentes configurações promoviam diferentes resultados, dependendo do problema analisado e do conjunto de dados utilizado, de acordo com (KOHAVI; JOHN, 1995).

Com a rápida ascensão, diversas metodologias de otimização de hiperparâmetros têm sido aplicadas para aprimorar *pipelines* de uso geral em aplicações mais específicas (ESCALANTE; MONTES; SUCAR, 2009), ou mesmo incrementar a configuração padrão das bibliotecas comuns de aprendizado de máquina, como pesquisado em (THORNTON et al., 2013), (MANTOVANI et al., 2016), (SANDERS; GIRAUD-CARRIER, 2017), (OLSON et al., 2018).

Um dos principais fatores de sucesso na abordagem de otimização de tais algoritmos para problemas reais, consiste na determinação de um modelo com arquitetura que melhor se adapte ao sistema a ser analisado. Neste sentido, destacam-se: a adequação da estrutura do banco de dados disponível, a determinação do método de processamento computacional e a saída que se deseja obter do modelo.

Com o objetivo de promover um desempenho adequado ao processamento do algoritmo, o ajuste de seus hiperparâmetros representa um procedimento de elevada importância, capaz de impactar diretamente na resolução do problema. Nesta empreitada, é possível destacar os métodos heurísticos, os quais são baseados em algoritmos que podem ser utilizados para buscar boas soluções em complexos problemas de otimização (JACOBSON; YÜCESAN, 2004).

Ao contrário dos métodos analíticos empíricos, as heurísticas possibilitam trabalhar com problemas de alta complexidade, fornecendo soluções satisfatórias com poucos recursos computacionais, em um tempo consideravelmente razoável. Este tipo de abordagem tem recebido cada vez mais popularidade nos últimos vinte anos, principalmente em projetos de engenharia em geral, aprendizado de máquina, modelagem e simulação de sistemas, planejamento de logística, meio ambiente e diversos outros (TALBI, 2009).

Desde os primeiros estudos envolvendo inteligência artificial, pesquisadores têm descrito o termo "heurística" de formas distintas. Contudo, em (RUSSELL; NORVIG, 2020) é descrito que em 1957, George Polya publicou uma obra intitulada *How to Solve It* (Como Resolver Isso),

em que o termo "heurística" havia sido utilizado com o objetivo de designar o estudo dos métodos utilizados para investigar e propor técnicas de resolução de problemas sem a necessidade de apresentar provas matemáticas, permitindo abordar assim aqueles problemas não passíveis de explicação.

Em dias atuais, o termo “heurística” tem sido utilizado como um adjetivo, designando técnicas que melhorem o desempenho em uma tarefa de resolução de problemas e, no que tange aos algoritmos de busca, denota-se a uma função que fornece uma estimativa do custo da solução (RUSSELL; NORVIG, 2020).

Portanto, como já abordado na Seção 1.3, o ajuste de redes neurais artificiais a partir da configuração de seus hiperparâmetros é uma tarefa complexa, desafiadora e custosa, mas fundamental para melhorar o desempenho de tais sistemas. Neste sentido, a implementação de um método heurístico pode servir de apoio no intento de identificar um conjunto de boas soluções, dentro de um espaço de busca previamente delimitado e um intervalo de tempo pré-definido.

## 3.1 Otimização de hiperparâmetros de CNNs

A quantidade de investigação em torno dos hiperparâmetros de redes neurais convolucionais (CNNs) tem se intensificado nos últimos anos (FEURER; HUTTER, 2019). Tais hiperparâmetros, por exemplo, podem se referir à quantidade de camadas convolucionais, seus respectivos neurônios, tamanho e passo dos filtros, além de tipos de *Pooling*, quantidade de camadas densas e de neurônios em tais camadas, entre outros (ALBELWI; MAHMOOD, 2016).

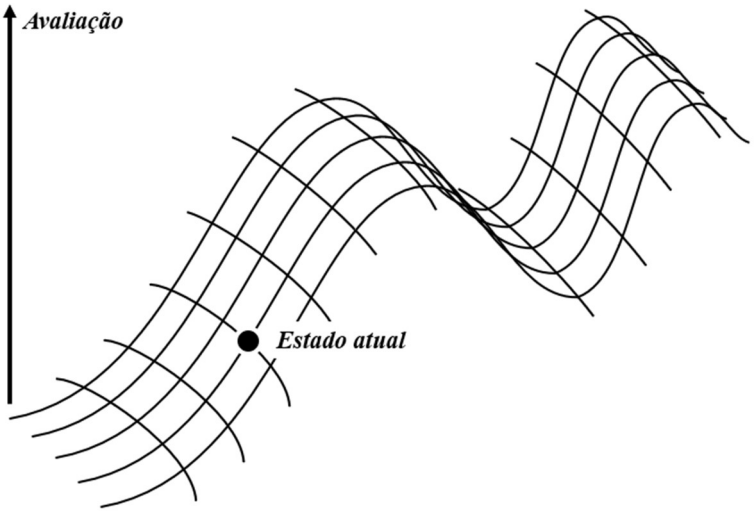
Na literatura ainda não há trabalho que investigue a otimização de todos os hiperparâmetros, e como eles se relacionam, no projeto de uma arquitetura de CNN. Utilizar qualquer metodologia a partir de uma abordagem muito generalista deve ser visto com cuidado, pois a determinação de uma arquitetura ótima será sempre em função da especificidade dos dados utilizados para treinamento (ANDONIE; FLOREA, 2020).

## 3.2 Algoritmo de Busca Local: *Hill Climbing*

O algoritmo intitulado *Hill Climbing* (subida de encosta), comumente designado como algoritmo de busca local simples, é considerado um dos métodos heurísticos de implementação mais simples. Podendo, ou não, ser inicializado com uma solução aleatória, tal método desenvolve um processamento iterativo, partindo de uma solução atual para uma solução vizinha considerada melhor, até que alcance uma solução ótima local (AL-BETAR, 2017). O objetivo pode ser maximizar a função de custo, quando, geralmente, a curva de otimização assume um aspecto ascendente, ou minimizar a função de custo, quando a mesma curva assume um aspecto descendente.

Segundo (RUSSELL; NORVIG, 2020), uma interessante forma de ilustrar os algoritmos de processamento iterativo é representar todas as possíveis soluções em uma superfície denominada espaço de busca, de maneira que a altura de qualquer um dos pontos da superfície equivale à função de custo naquela solução, conforme indicado na Figura 23. O objetivo é que seja promovida a exploração do espaço de busca, visando encontrar os pontos mais altos ou mais baixos, dependendo do objetivo, os quais corresponderão às soluções ótimas. Tal empreitada se desenvolve de maneira que apenas o estado atual seja computado, ao invés de considerar as informações de soluções vizinhas passadas, o que reduz consideravelmente os recursos computacionais necessários para o processo de otimização e, em contrapartida, tornando-se mais sujeito a ficar "preso" em uma solução ótima local.

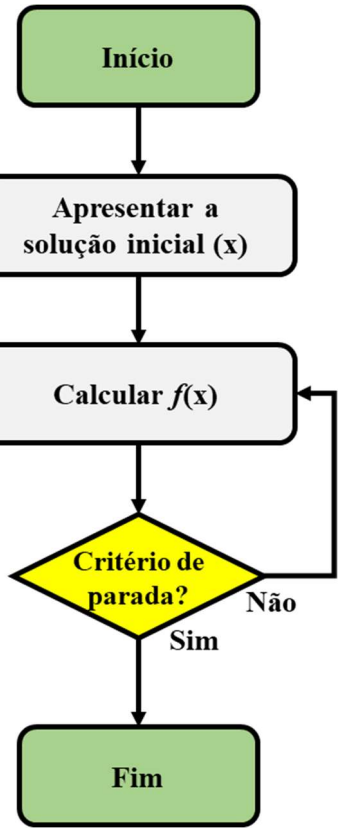
Figura 23 – Estratégia de algoritmo de melhoria iterativa.




Fonte: Adaptado de (RUSSELL; NORVIG, 2020)

Em termos de implementação computacional, tal heurística pode ser representada através do fluxograma da Figura 24. Após a definição da solução inicial, o algoritmo calculará a função de custo até que um critério de parada seja atingido. Caso positivo, o algoritmo retorna à solução, caso contrário, continua o processo de melhoria iterativa até que o objetivo seja alcançado.

Figura 24 – Fluxograma do algoritmo denominado *Hill Climbing*.




Fonte: Adaptado de (AL-BETAR, 2017)

## 3.3 Ajuste de Redes Neurais Artificiais com *Hill Climbing*

Apesar de sua simplicidade e possibilidade de estagnar em regiões ótimas locais, uma quantidade relevante de trabalhos envolvendo otimização de redes neurais artificiais tem sido realizada com o apoio de heurística do tipo *Hill Climbing*. Alguns fatores que justificam sua utilização são:

a) A facilidade de implementação, que não exige o desenvolvimento de complexas estruturas de códigos ou a utilização de bibliotecas de terceiros;

b) Menor custo computacional, uma vez que o algoritmo não armazena iterações passadas para utilizá-las na tomada de decisão atual no espaço de busca;

c) Possibilidade de incorporar outros mecanismos estocásticos que possibilitem a exploração de diferentes regiões do espaço de busca, evitando a estagnação em ótimos locais, como, por exemplo, desenvolvido em (AL-BETAR, 2017).

Com relação ao que se tem na literatura, em (DASGUPTA; SIKDER; MANDAL, 2017), foi utilizada uma rede neural do tipo *Feedfoward* em conjunto com um Algoritmo Genético (GA) (HOLLAND, John Henry, 1992) para otimizar a retenção de corante reativo de soluções aquosas por meio de ultrafiltração aprimorada do polímero polietilenoimina (PEI). A exploração do espaço de busca local, por meio do GA, foi aprimorada através da implementação do método *Hill Climbing*, possibilitando evidenciar soluções melhores do que as encontradas com a utilização do GA apenas.

No trabalho de (ALWESHAH et al., 2020), foi utilizada uma rede neural do tipo probabilística (PNN) (SPECHT, 1990), implementada para problemas de classificação utilizando onze bancos de dados de referência. Uma variante do algoritmo de *Hill Climbing*, denominado *β-Hill Climbing*, foi desenvolvido com o objetivo de encontrar o melhor conjunto de pesos para a rede. Seu diferencial em relação à heurística original está no acréscimo de uma abordagem estocástica, que evita que o algoritmo permaneça em ótimos locais. Foi constatado que a precisão de classificação com este método se demonstrou superior ao método com apoio do *Hill Climbing* simples e outras seis abordagens bem estabelecidas, todas utilizando o mesmo *benchmark*.

Em (KWASIGROCH; GROCHOWSKI; MIKOŁAJCZYK, 2020), utilizou-se uma rede neural convolucional (CNN) previamente treinada com aplicação da heurística *Hill Climbing* para o problema de classificação de imagens de melanomas malignos. Foi concluído que o método de otimização proporcionou a busca eficaz de uma arquitetura que proporcionasse resultados semelhantes às configurações exclusivamente manuais referenciadas na literatura, porém, com uma quantidade de parâmetros aproximadamente 20 vezes menor.

Já em (ZHU et al., 2020), foi utilizada uma CNN para detectar a trepidação durante o fresamento de peças com paredes finas, fazendo a classificação das imagens de superfícies já usinadas. O algoritmo de *Hill Climbing* foi utilizado para aprimorar a eficiência de um Algoritmo Genético (GA) modificado durante a busca pelos melhores ajustes dos hiperparâmetros determinados no espaço de busca.

Finalmente, semelhante a este trabalho, (CHEN, H.; WANG; FAN, 2020) implementaram uma CNN para lidar com o monitoramento não intrusivo de cargas elétricas (NILM) em uma aplicação residencial. Os autores implementaram o algoritmo *Hill Climbing* para a configuração automática do hiperparâmetro denominado taxa de aprendizado, que permitiu resultados entre 36,4% e 83,34% superiores se comparados à mesma arquitetura sem o otimizador.

## 3.4 Algoritmo de Otimização Bayesiana

A Otimização Bayesiana surgiu no final da década de 60, assumindo um papel de destaque no trabalho de (JONES, D. R.; SCHONLAU; WELCH, 1998) em que foi investigado um método denominado Otimização Global Eficiente em Funções "Caixa-Preta" com Alto Custo.

Nos últimos anos, este tipo de otimização tem provado ser uma eficiente estratégia no apoio à resolução de problemas de projeto e, tanto na academia quanto no ambiente corporativo, tem prestado apoio à tomada de decisões envolvendo robótica, aprendizado de máquina automatizado, redes de sensores, entre outros (SHAHRIARI, B. et al., 2016).

Em (DOWNEY, 2013) é apresentado que, fundamentalmente, tal otimizador está baseado no Teorema de Bayes, que especifica a probabilidade condicional de eventos, isto é, um evento ser verdadeiro dada a existência de outro evento, conforme a Equação 3.1

$$P(A|B) = \frac{P(A)\,P(B|A)}{P(B)}, \qquad (3.1)$$

onde, em termos computacionais, $A$ é o parâmetro (ou vetor de parâmetros) estimado; $B$ representa os dados observados; P($A$) é a probabilidade de um parâmetro $A$; P($B|A$) é a probabilidade de um dado $B$, para um determinado parâmetro $A$; e P($B$) é a probabilidade de um dado $B$.

Sob um ponto de vista aplicado, tal algoritmo cria um modelo que tenta predizer uma função de custo real e, à medida que novos dados são observados, este será recursivamente atualizado, reduzindo a incerteza para as regiões não observadas da função (SHAHRIARI, B. et al., 2016).

Segundo (FRAZIER, 2018), o otimizador bayesiano compreende dois componentes principais, sendo o modelo estatístico bayesiano, que criará uma função substituta (*surrogate function*) à função de custo real, e uma função de aquisição (*acquisition function*). Ainda, de acordo com o autor, o modelo estatístico geralmente segue um processo gaussiano (GP) que, em essência, significa que, assumindo que um conjunto finito de valores observados em uma função *f(x)* seguem uma distribuição normal, o modelo distribuirá as incertezas entre os pontos não observados, seguindo, também, uma distribuição normal.

A função de aquisição, por outro lado, representa uma valor matemático que conduzirá o otimizador em direção às regiões contendo potenciais soluções no interior do espaço de busca (BERGSTRA et al., 2011) e, à respeito das funções mais utilizadas, destaca-se a do tipo Melhoria Esperada (*Expected Improvement*) (JONES, D. R., 2001), designada através da Equação 3.2

$$EI_{y^*}(x) \coloneqq \int_{-\infty}^{\infty} \max(y^* - y, 0) P_M(y|x)\, dy, \qquad (3.2)$$

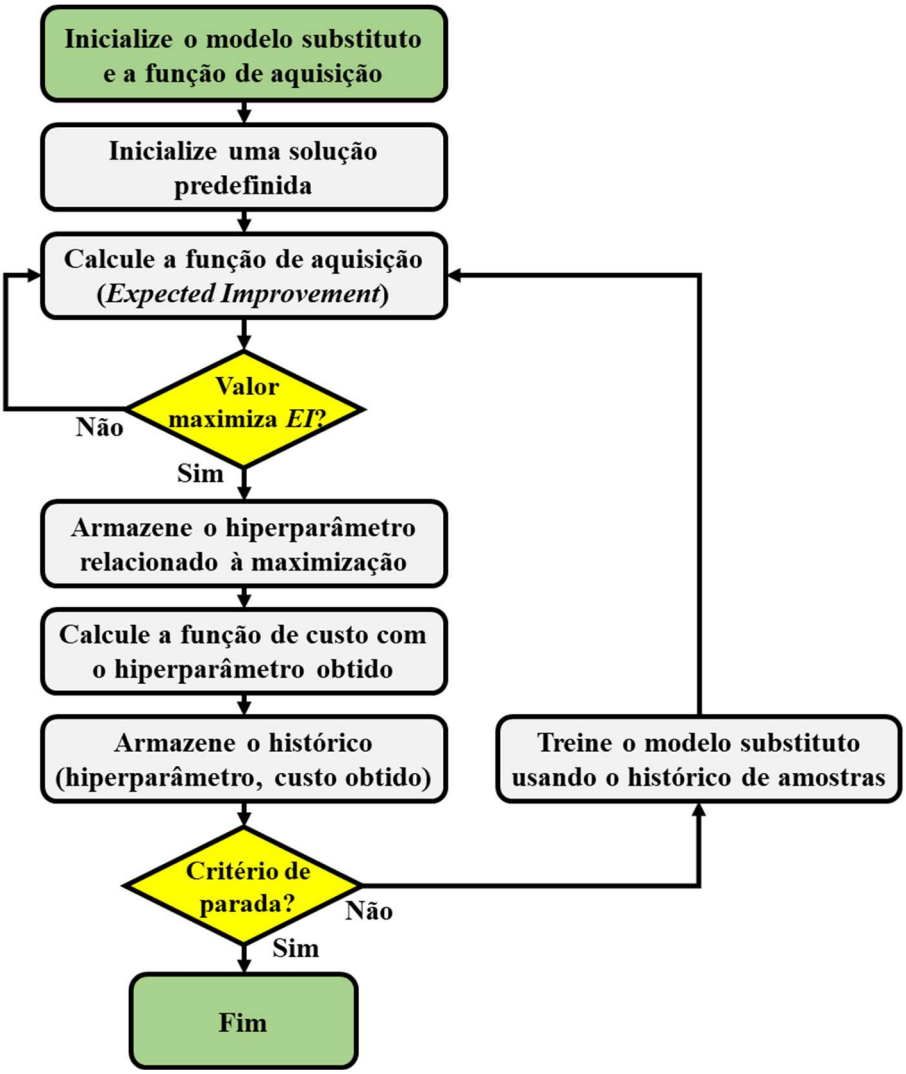
onde $x$ é um dado hiperparâmetro; $y^*$ é a pontuação mínima da função de custo verdadeira observada até o momento; $y$ é a nova pontuação obtida; e $P_M(y|x)$ é a probabilidade, em um modelo $M$, de atingir uma pontuação $y$, dado um hiperparâmetro $x$.

Portanto, o otimizador fará a busca nas regiões em que tais parâmetros maximizarem a saída da função de aquisição (do tipo EI), isto é, as soluções candidatas que compõem uma maior incerteza na predição do modelo substituto (SHAHRIARI, B. et al., 2016).

O fluxograma da Figura 25 detalha as etapas desenvolvidas pelo algoritmo de otimização bayesiana. No momento da inicialização do programa, são criadas a função substituta e a função de aquisição. Haja vista que uma solução candidata tenha sido predefinida pelo desenvolvedor, o algoritmo buscará qual é o hiperparâmetro capaz de provocar a maximização da função de aquisição para, enfim, ser armazenado e testado na verdadeira função de custo.

Logo em seguida, o par de informações (hiperparâmetro e pontuação da função de custo) é armazenado. Enquanto não for atingido o critério de parada, que pode ser o tempo de execução ou o número de iterações, o algoritmo atualiza os dados no modelo substituto e realiza uma nova verificação na função de aquisição, que o direcionará para testes em regiões cada vez mais promissoras no espaço de busca.

Figura 25 – Fluxograma de implementação do otimizador bayesiano com função de ativação do tipo *Expected Improvement* (EI).




Fonte: (Autor, 2021)

## 3.5 Ajuste de Redes Neurais Artificiais com Otimização Bayesiana

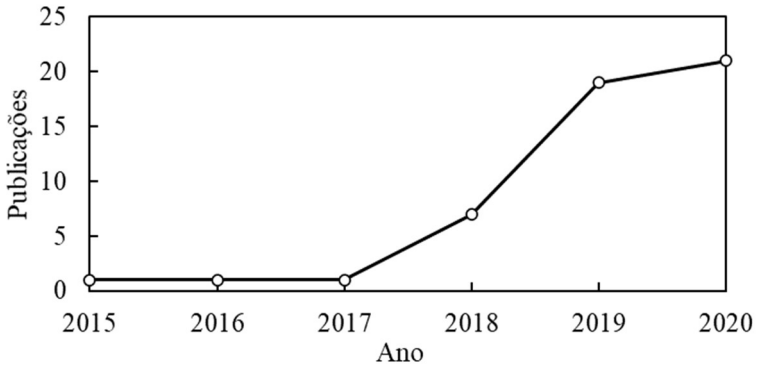
Segundo (ANDONIE; FLOREA, 2020), os métodos mais utilizados para otimização de hiperparâmetros em RNAs são: Busca em Grade, que consiste basicamente no teste com todas as combinações possíveis; Busca Aleatória (BERGSTRA; BENGIO, 2012); Otimização Bayesiana (HUTTER; HOOS; LEYTON-BROWN, 2011); *Nelder-Mead* (NELDER; MEAD, 1965); Recozimento Simulado (KIRKPATRICK; GELATT; VECCHI, 1983); Otimização por Enxame de Partículas (KENNEDY; EBERHART, 1995); e Algoritmos Evolucionários (RECHENBERG, 1975). Num comparativo entre tais técnicas, os autores afirmam ainda que a

Otimização Bayesiana (*Bayesian Optimization*) se mostra uma ferramenta promissora em problemas de baixa e média dimensionalidade.

A otimização bayesiana tornou-se uma das mais populares, principalmente, pelo fato de desempenhar o papel de ajuste automatizado de hiperparâmetros com desempenho tão bom quanto, ou superior, às demais heurísticas disponíveis (WEI et al., 2019). Visando obter um panorama mais detalhado de aplicação da otimização bayesiana para o ajuste de hiperparâmetros em RNAs, especificamente nas CNNs, buscou-se desenvolver uma pesquisa bibliométrica que fornecesse tais dados.

A base de dados utilizada para a pesquisa foi a *Web of Science*. Para delimitar a busca, foi utilizada sequência de palavras-chave: "(*bayesian optimization* AND (*convolutional neural network** OR *CNN*))", e delimitado o filtro para "somente artigos", através da aba de pesquisa avançada. Foi possível constatar um total de 66 artigos, dos quais, após análise e triagem dos títulos e respectivos resumos, restaram 53 itens com alta similaridade ao tema deste trabalho. Fazendo uma análise de tais números, conforme ilustrado na Figura 26, é possível verificar um aumento contínuo na implementação deste tipo de otimizador junto à CNNs, evidenciando consistência na implementação de tal ferramenta e prestação de suporte a trabalhos considerados estado-da-arte em suas respectivas áreas.

Figura 26 – Evolução de pesquisas envolvendo otimização bayesiana e CNNs.




Fonte: Web of Science – Clarivate Analytics, 2021

É possível constatar uma variedade de especialidades envolvendo o tema, tais como elétrica, materiais, química, transportes, operações e medicina. Não é o objetivo deste trabalho discutir todas as aplicações aqui levantadas. Contudo, são destacados a seguir alguns artigos de relevância para o estudo. A listagem de todos os artigos encontrados pode ser conferida no Apêndice B.

Em (BASHA; VINAKOTA; PULABAIGARI; et al., 2021), desenvolveu-se um *benchmark* referente ao desempenho de CNNs previamente treinadas para classificação de imagens em três bases de dados diferentes. Foram implementados os otimizadores de busca aleatória e bayesiano para o ajuste automatizado de hiperparâmetros (número de camadas densas; tipo de função de ativação; número de neurônios e intensidade de *dropout* em tais camadas). Evidenciou-se que os modelos ajustados com o otimizador bayesiano apresentaram as maiores melhorias na acurácia de classificação, levando as implementações ao estado-da-arte em sua respectiva área.

No trabalho de (BASHA; VINAKOTA; DUBEY; et al., 2021), de forma similar, desenvolveu-se um *benchmark* referente ao desempenho de CNNs previamente treinadas para classificação de imagens em outras três bases de dados diferentes. A partir da implementação de diferentes otimizadores para o ajuste automatizado de hiperparâmetros (número de filtros na camada de convolução; número e tamanho de filtros na camada de *pooling*; número de camadas densas; e intensidade do *dropout*), foi evidenciado que os modelos ajustados com o otimizador bayesiano apresentaram as maiores melhorias na acurácia de classificação.

Em (SHI et al., 2020), foi proposta uma CNN visando um baixo esforço computacional para implementação em um sensor de baixo custo acoplado em vagões de carga. O objetivo de detectar a condição de "roda plana" a partir da análise de aceleração em tais vagões, foi concluído com sucesso através do ajuste automatizado dos hiperparâmetros (taxa de aprendizado; tamanho do *batch*; número de épocas de treinamento; número de neurônios / tamanho do filtro em cada uma das 5 camadas convolucionais), que reduziram substancialmente o número de parâmetros da CNN proposta, superando as redes até então consideradas estado-da-arte.

Já em (EKICI et al., 2020), aplicou-se a técnica de Transformada *Wavelet* Contínua (CWT) em sinais de tensão para a análise de distúrbios de qualidade de energia (PQDs). Foram obtidas imagens com diferentes fatores de escala, posteriormente treinadas com CNNs. O otimizador bayesiano foi empregado para ajustar os hiperparâmetros da rede (taxa de aprendizagem; *momentum* e regularização L2). O método proposto alcançou um desempenho superior, se comparado aos outros algoritmos de aprendizado de máquina existentes para este fim.

Finalmente, em (WEI et al., 2019), comparou-se diferentes modelos de *deep learning* na tarefa de reconhecimento automático de modulação de sinal em duas bases de dados. Foi

constatado que a CNN cujos hiperparâmetros (número de neurônios / camadas convolucionais; tipo de função de ativação; e número de neurônios / camadas densas) foram ajustados pelo otimizador bayesiano, apresentou desempenho superior ao mesmo modelo ajustado por Algoritmo Genético e aos modelos sem qualquer ajuste automatizado.

Portanto, a reflexão que se pode fazer dos trabalhos desenvolvidos, tanto para pesquisa científica quanto para aplicações práticas, é a de que se utiliza tal otimizador em conjunto com CNNs principalmente para:

a) Reduzir o tempo gasto para encontrar um bom ajuste de hiperparâmetros;

b) Investigar um ajuste de hiperparâmetros que leve o atual desempenho da CNN ao estado-da-arte em sua respectiva área de aplicação;

c) Investigar a hibridização de processos de otimização, contemplando os conceitos da Teoria de Bayes ou de processos gaussianos, com outros *frameworks* já existentes.

# 4 Metodologia

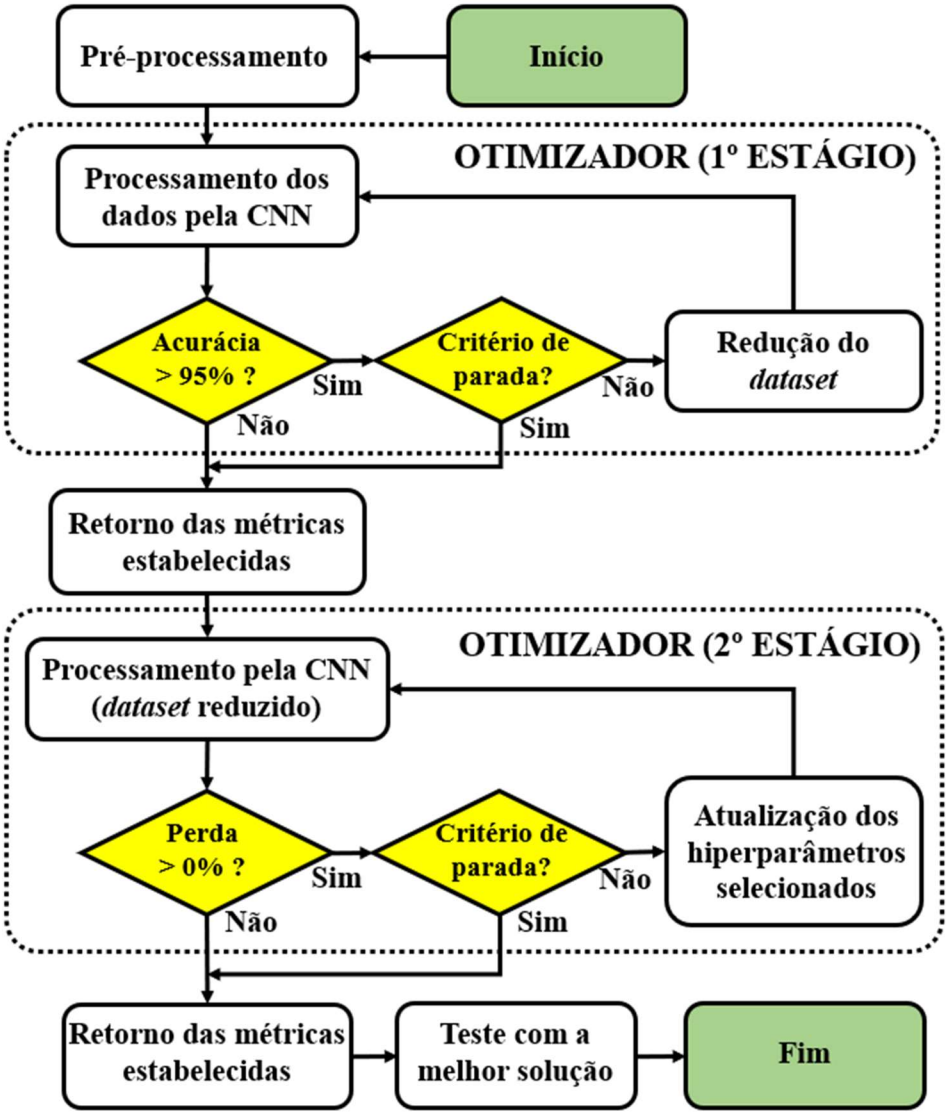
Neste capítulo é apresentada a arquitetura experimental e os procedimentos adotados para tratar o problema levantado e alcançar os objetivos propostos no Capítulo 1. O fluxograma da Figura 27 ajuda a compreender o denominado sistema de ajuste automatizado da CNN. Como pode-se observar na figura, o sistema é dividido em dois estágios de otimização.

O 1º estágio consiste na busca automática por um número mínimo de dados contidos em um *dataset*, que sejam necessários e suficientes para o treinamento adequado da CNN selecionada. Considera-se treinamento adequado de CNN a manutenção da acurácia de teste obtida ao treinar a rede com o conjunto de dados completo. O objetivo do 1º estágio é o de usar tal resultado durante a execução do 2º estágio de otimização, na tentativa de reduzir o tempo de convergência durante a busca por valores ótimos de um conjunto pré-definido de hiperparâmetros da arquitetura CNN. Esse ajuste automatizado de hiperparâmetros tem como meta entregar uma arquitetura de rede mais eficiente, substituindo o esforço e a expertise humana normalmente empregados de forma empírica em processos de construção de RNAs.

Sendo mais específico, o 1º estágio de otimização conta com um otimizador do tipo *Hill Climbing* e um conjunto de dados referentes a 600 exemplos (casos) de 16 classes a serem classificadas. A base de dados será, portanto, reduzida sucessivamente até que se atinja o objetivo, que corresponde à uma acurácia nos dados de teste igual a 95%, ou o critério de parada, que é estabelecido em 10% da quantidade total de exemplos, isto é, 60 casos para cada classe. Assim que um ou outro alvo é alcançado, o sistema finaliza o processo de otimização e fornece o histórico dos dados do experimento para análise. Cabe ressaltar que, tanto o limite de degradação da acurácia de classificação (95%) quanto o limite de redução de casos (10% do total), foram estabelecidos empiricamente, visando manter sob controle o desempenho da arquitetura CNN durante o 2º estágio de otimização.

Já o segundo estágio consiste em utilizar a solução encontrada após o término do 1º estágio, isto é, a quantidade de casos ótima, para que seja minimizada a nova função de custo através de um otimizador bayesiano. Tal função de custo tem como objetivo minimizar a perda (*loss*) de identificação através do ajuste dos hiperparâmetros selecionados.

Figura 27 – Fluxograma de implementação do ajuste automatizado da CNN.




Fonte: (Autor, 2021)

É utilizada a ferramenta *Google Colaboratory*, a qual possibilita o desenvolvimento colaborativo e o acesso remoto (*cloud*) às GPUs da *Google* (modelo Nvidia Tesla V100), fundamentais para se conseguir uma convergência a possíveis soluções dentro da restrição de tempo. Além disso, a plataforma utiliza como base o modelo de programação estilo *Jupyter Notebook*, que facilita a construção e análise de experimentos científicos. O micro computador utilizado para rodar o navegador de internet e, indiretamente, os experimentos, é um Dell Alienware 17 R3.

Devido a possíveis instabilidades na rede internet e às limitações em tempo de execução e recursos computacionais impostas pelos servidores do *Google Colab*, é implementado, em ambos os estágios de otimização, um recurso de regularização do modelo denominado parada

antecipada (*early stopping*). Este recurso é empiricamente configurado para ser acionado após 100 épocas sem que o modelo apresente alguma melhora na melhor acurácia obtida sobre às amostras de validação (KERAS, 2021b).

Cabe ressaltar que o otimizador empregado no primeiro estágio é de desenvolvimento do autor, enquanto que no segundo estágio, é empregado o *framework* denominado *Scikit-optimize* (HEAD et al., 2020), pela facilidade para implementação, pelas métricas fornecidas para investigação de tendências no espaço de busca e por estar fundamentado em importantes bibliotecas que, inclusive, foram utilizadas em outras partes do código, como o *NumPy* (NUMPY, 2021) e *Scikit-Learn* (SCIKIT-LEARN, 2021b).

## 4.1 Bases de Dados

As bases de dados utilizadas para simulação do processo de otimização heurística serão as mesmas coletadas e utilizadas em (FIRMES, 2020). O detalhamento do aparato utilizado e da metodologia aplicada podem ser consultados, em detalhes, no referido trabalho.

Representando as cargas altamente similares, do tipo *on*/*off*, invariantes no tempo, conectadas ao ponto comum de acoplamento (PCC), (FIRMES, 2020) menciona que foram utilizadas quatro lâmpadas fluorescentes, com 15W e 127V, do mesmo fabricante, compreendendo o conjunto de dados denominado banco $A_0$. O segundo conjunto de dados, denominado banco B, corresponde a quatro microcomputadores representando cargas de variação não determinística, cujo modelo é Dell Optiplex 960, Core 2 Duo 3 GHz, 4GB RAM e HD de 150GB.

Ambos os conjuntos de dados $A_0$ e B possuem 999600 amostras de 600 casos (ou exemplos), de modo que cada caso é representado por 1666 amostras consecutivas. De acordo com (FIRMES, 2020), os conjuntos de dados foram amostrados à uma taxa de 99960 amostras por segundo, com resolução de 16 bits.

A Tabela 2 apresenta a definição das classes associadas ao estado dos equipamentos que compõem as cargas registradas nos bancos $A_0$ e B. O detalhamento das condições de obtenção de tais dados é mostrado em (FIRMES, 2020).

Tabela 2 – Esquema de funcionamento das cargas.

| Classe | Q4 | Q3 | Q2 | Q1 | Estado |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | Todas desligadas |
| 1 | 0 | 0 | 0 | 1 | Q1 ligada |
| 2 | 0 | 0 | 1 | 0 | Q2 ligada |
| 3 | 0 | 0 | 1 | 1 | Q1 e Q2 ligadas |
| 4 | 0 | 1 | 0 | 0 | Q3 ligada |
| 5 | 0 | 1 | 0 | 1 | Q1 e Q3 ligadas |
| 6 | 0 | 1 | 1 | 0 | Q2 e Q3 ligadas |
| 7 | 0 | 1 | 1 | 1 | Q1, Q2 e Q3 ligadas |
| 8 | 1 | 0 | 0 | 0 | Q4 ligada |
| 9 | 1 | 0 | 0 | 1 | Q1 e Q4 ligadas |
| 10 | 1 | 0 | 1 | 0 | Q2 e Q4 ligadas |
| 11 | 1 | 0 | 1 | 1 | Q1, Q2 e Q4 ligadas |
| 12 | 1 | 1 | 0 | 0 | Q3 e Q4 ligadas |
| 13 | 1 | 1 | 0 | 1 | Q1, Q3 e Q4 ligadas |
| 14 | 1 | 1 | 1 | 0 | Q2, Q3 e Q4 ligadas |
| 15 | 1 | 1 | 1 | 1 | Todas ligadas |

Fonte: (FIRMES, 2020)

## 4.2 Definição das CNNs

De acordo com as razões expostas na Seção 1.3, as arquiteturas de redes neurais convolucionais utilizadas em (FIRMES, 2020) são utilizadas neste trabalho. Será necessário utilizar um método de escalonamento (também conhecido como pré-processamento) dos dados, isto é, padronizar a faixa de variação dos valores constituintes para que o modelo possa assimilá-los com maior eficácia (GÉRON, 2017).

O pré-processamento pode ser desenvolvido, por exemplo, através do método *Standard Scaler*, que ignora a forma da distribuição e faz a transformação dos dados para uma média próxima de zero e um desvio padrão próximo a um, evitando valores discrepantes no conjunto (SCIKIT-LEARN, 2021c).

Em (FIRMES, 2020), os treinamentos das redes foram realizados utilizando 64% dos casos (6144 amostras), e as validações das redes com outros 16% dos casos (1536 amostras). Os 20% dos casos restantes (1920 amostras) foram utilizados para testagem das redes treinadas e validadas. Tais etapas (treinamento e validação) foram repetidas a cada variação manual de hiperparâmetros da CNN. O conjunto de hiperparâmetros manualmente e empiricamente

ajustados em (FIRMES, 2020) foram o tamanho do *batch*, número de épocas, tipo de otimizador do gradiente descendente, quantidade de camadas e quantidade de neurônios.

Denomina-se *batch* a quantidade de amostras que será processada por vez, a cada época, na fase de treinamento. Já o número de épocas de treinamento representa a quantidade de vezes que um conjunto de amostras foi utilizado para ajustar os pesos e limiares da rede, conforme já apresentados no Capítulo 2.

Ademais, foram realizados testes com dois tipos de otimizadores do gradiente descendente, o SGD (*Stochastic Gradient Descent*) e o Adam (*Adaptive Moment Estimation*), de modo que, tanto o SGD quanto o Adam executaram atualizações de parâmetros para cada *batch* (RUDER, 2016). Entretanto, o SGD convergiu o gradiente descendente cerca de 20% mais rápido, motivo pelo qual foi utilizado na maioria das arquiteturas testadas em (FIRMES, 2020).

A Tabela 3 apresenta as arquiteturas implementadas no trabalho em questão, as quais reproduziram resultados que serão comparados e discutidos frente aos dados apresentados neste trabalho, a partir da implementação do ajuste automatizado de hiperparâmetros.

Tabela 3 – Arquiteturas de CNN implementadas.

| **Rede** | **Camadas** | **Neurônios** | **Tamanho e Passo** | ***Batch*** | **Épocas** |
|---|---|---|---|---|---|
| 1 | Convolucional: 2<br>*MaxPooling*: 2<br>Normalização: 2<br>Densas: 4 | Conv.: 32,16.<br>Densa: 22, 18, 8, 16. | Conv.: 2 e 1<br>MaxP.: 2 | 32 | 2000 |
| 2 | Convolucional: 2<br>*MaxPooling*: 2<br>Normalização: 2<br>Densas: 2 | Conv.: 32, 16.<br>Densa= 22, 18, 8, 16. | Conv.: 2 e 1<br>MaxP.: 2 | 32 | 1800 |
| 3 | Convolucional: 4<br>*MaxPooling*: 4<br>Normalização: 4<br>Densas: 4 | Conv.: 32, 64, 128, 256.<br>Densa: 512, 256, 128, 16 | Conv.: 4 e 1<br>MaxP.: 3 | 32 | 500 |
| 4 | Convolucional: 4<br>*MaxPooling*: 4<br>Normalização: 4<br>Densas: 4 | Conv.: 50, 70, 100, 200.<br>Densa: 500, 250, 175, 16 | Conv.: 4 e 1<br>MaxP.: 3 | 15 | 500 |

Fonte: (FIRMES, 2020)

Visando desenvolver um planejamento que permita a execução dos experimentos em tempo hábil e prevendo possíveis limitações ou problemas de ordem técnica durante os experimentos de otimização, é necessário que sejam estabelecidos critérios para a escolha de modelos de CNN que forneçam boas métricas com relativa rapidez de processamento. Neste trabalho, portanto, foi adotado o critério de prioridade, utilizando os seguintes parâmetros:

1. Acurácia superior a 95%;
2. Menor tempo de treinamento (*t*).

Portanto, conforme pode ser evidenciado na Tabela 4, para o conjunto de dados referente às lâmpadas, elege-se a Rede #3 para investigação.

Tabela 4 – Resultados de CNNs - Conjunto de dados $A_0$.

| Rede | Parâmetro (P) | Parâmetro (p) | Trein. (TT) | Trein. (t) | Acurácia (%) |
|---|---|---|---|---|---|
| 1 | 148.652 | 0,05 | 1732 | 1 | 96,61 |
| 2 | 148.844 | 0,05 | 332 | 0,19 | 93,07 |
| 3 | 2.962.800 | 1 | 45 | 0,03 | 99,90 |
| 4 | 2.296.791 | 0,78 | 106 | 0,06 | 100,00 |

Fonte: (FIRMES, 2020)

E, de acordo com as informações obtidas através da Tabela 5, para o conjunto de dados referente aos computadores, elege-se, igualmente, a Rede #3 para investigação.

Tabela 5 – Resultados de CNNs - Conjunto de dados B.

| Rede | Parâmetro (P) | Parâmetro (p) | Trein. (TT) | Trein. (t) | Acurácia (%) |
|---|---|---|---|---|---|
| 1 | 148.652 | 0,05 | 5158,9 | 1 | 92,45 |
| 2 | 148.844 | 0,05 | 1098 | 0,21 | 96,46 |
| 3 | 2.962.800 | 1 | 154 | 0,03 | 98,49 |
| 4 | 2.296.791 | 0,78 | 503 | 0,097 | 99,01 |

Fonte: (FIRMES, 2020)

## 4.3 Estrutura e Pré-processamento dos Dados

Com o objetivo de tornar os dados preparados para o treinamento supervisionado da CNN, isto é, cada caso tenha um rótulo (*label*) corretamente atribuído, são necessárias algumas operações para sua estruturação.

Os conjuntos de dados coletados e disponibilizados por (FIRMES, 2020) estão originalmente estruturados como dados no formato .csv. Após a importação para o *Google Colab*, através da biblioteca *Pandas* (PANDAS, 2021), o arquivo adquire o formato *Pandas dataframe*, de estrutura tabular. Em seguida, os dados são convertidos para vetores, através da biblioteca *Numpy*, que permite a manipulação matemática com maior flexibilidade e segurança, via algoritmo.

Tais manipulações, como evidenciado na Figura 28, envolvem a transposição dos dados, a fim de facilitar o processo de fatiamento automatizado de casos para cada atributo (1º estágio de otimização é detalhado na Seção 4.4); a modificação da estrutura dos dados (*reshape*) para uma matriz tridimensional, executada para a rotulação dos dados, conforme (FIRMES, 2020); o pré-processamento dos dados; e o seccionamento em amostras de treinamento (64%), validação (16%) e teste (20%), de acordo com o Princípio de Pareto (HARVEY; SOTARDI, 2018).

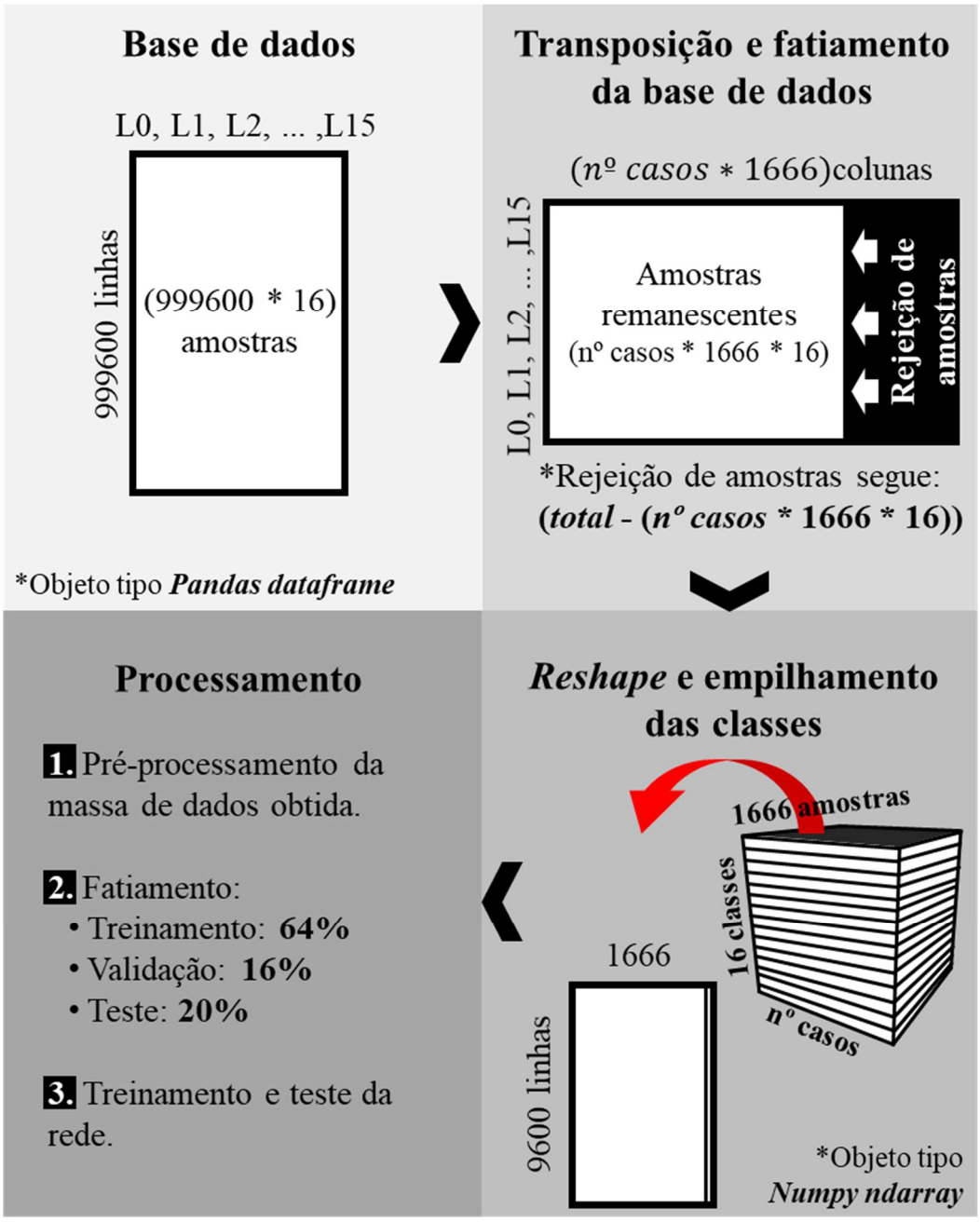
Além do tamanho da base de dados ou da arquitetura da rede neural utilizada, é de suma importância preocupar-se com o método de pré-processamento dos dados, uma vez que algoritmos de aprendizagem podem apresentar desempenhos diferentes, de acordo com o tratamento de dados discrepantes (*outliers*) utilizado no conjunto (SCIKIT-LEARN, 2021c).

São testados os métodos mais utilizados em aplicações práticas envolvendo modelos de *deep learning*, como o *StandardScaler*, que foi implementado em (FIRMES, 2020) e o *MinMaxScaler*, ambos disponíveis na biblioteca *Scikit-Learn*.

Contudo, uma vez que este trabalho aborda modificações relevantes no tamanho das bases de dados utilizadas, o que pode impactar na distribuição estatística de tais *outliers*, caso seja evidenciada instabilidade no processo de treinamento, à medida que se reduz a quantidade de casos, podem ser testadas outras ferramentas de pré-processamento de dados, também disponíveis na mesma biblioteca. A estratégia é encontrar, caso necessário, o método de pré-processamento de dados que melhor adapte o conjunto de dados às novas condições de distribuição, à medida que o 1º estágio de otimização executa as operações de fatiamento.

Além disso, a taxa de aprendizagem, já discutida na Seção 2.2, também possui relevante influência no modo como o algoritmo generaliza as informações, podendo causar instabilidade ou lentidão na convergência dos dados. Por esse motivo, tal hiperparâmetro será investigado manualmente, visando incrementar a estabilidade no treinamento e validação do modelo, para que o otimizador não fique preso em uma solução mínima local durante o 1º estágio.

Figura 28 – Modelagem dos dados.




Fonte: (Autor, 2021)

## 4.4 Primeiro Estágio de Otimização

O 1º estágio do experimento, baseado nas justificativas expostas na Seção 3.3, consiste em aplicar a heurística *Hill Climbing* como ferramenta de otimização. Contudo, ao invés de conduzir o custo a uma subida de encosta, como o próprio nome sugere, o algoritmo desenvolverá o sentido oposto, isto é, a redução sucessiva da quantidade de casos (exemplos) disponíveis para treinamento da CNN, até que se perceba que a RNA atingiu a acurácia designada como alvo, assim como detalhado no fluxograma da Figura 29.

Em suma, será investigado a influência da redução progressiva da base de dados na degradação da acurácia da rede. Portanto, foi estabelecido empiricamente, que o otimizador implementará sucessivas reduções até que seja identificada uma degradação em torno de 5% na acurácia de classificação das amostras de teste. Isto reduz a probabilidade de não ser possível encontrar, no 2º estágio, um bom conjunto de hiperparâmetros (dadas as limitações de *hardware*) que eleve a acurácia ao mesmo patamar estabelecido em (FIRMES, 2020).

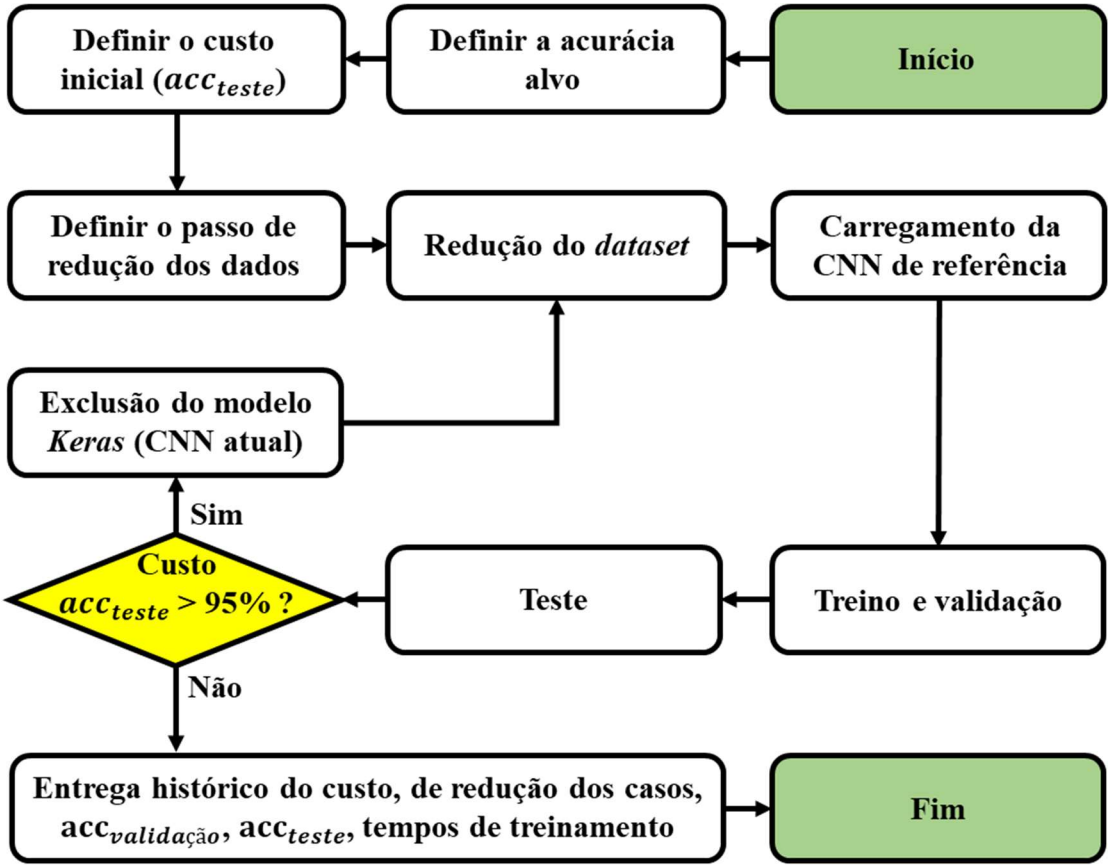
Contudo, o otimizador empregado durante o 1º estágio pode, eventualmente, permanecer preso em uma solução mínima local. Para lidar com esta situação, caso seja detectada, o experimento será novamente reproduzido com um "passo" maior de redução (Figura 29), isto é, a quantidade de casos que será reduzida a cada iteração do otimizador com a CNN. Deste modo, espera-se investigar se o decréscimo de acurácia (de uma iteração para outra) é realmente devido ao decréscimo da base de dados ou se é devido à uma característica estocástica do modelo de *deep learning*.

O algoritmo executará, portanto: a redução recursiva da base de dados baseada no passo de redução configurado; carregamento da CNN padrão; treinamento; validação e teste. Caso a acurácia nos dados de teste seja maior que 95%, o processo reinicia uma nova rodada, apagando a CNN atual, e recarregando uma nova, desta vez, com uma quantidade menor de casos. Caso contrário, verifica-se o critério de continuidade. Uma vez que ele não seja implementado, o otimizador finaliza o recurso e disponibiliza o histórico da função de custo (*fitness function*), da redução de casos, da maior acurácia de validação (encontrada em cada rodada), da acurácia nos dados de teste de cada rodada e, finalmente, do tempo de treinamento de cada rodada.

Cabe ressaltar que a implementação do processo de excluir o modelo *Keras* (CNN) e recarregá-lo a cada nova rodada, deve-se ao fato de evitar qualquer tipo de assimilação dos dados no *background* da CNN durante o processo de otimização. Isso se faz necessário pois pode ocorrer a evidência equivocada de uma boa acurácia de classificação mesmo com uma

baixíssima quantidade de dados quando, na verdade, o "mesmo" modelo *Keras* foi treinado em todas as rodadas do otimizador *Hill Climbing*. Tal condição deve ser observada pois pode impactar na confiabilidade dos resultados e não reproduz a hipotética situação em que o desenvolvedor fará um primeiro treinamento da rede com a solução encontrada no 1º estágio.

Figura 29 – Arquitetura do 1º estágio.




Fonte: (Autor, 2021)

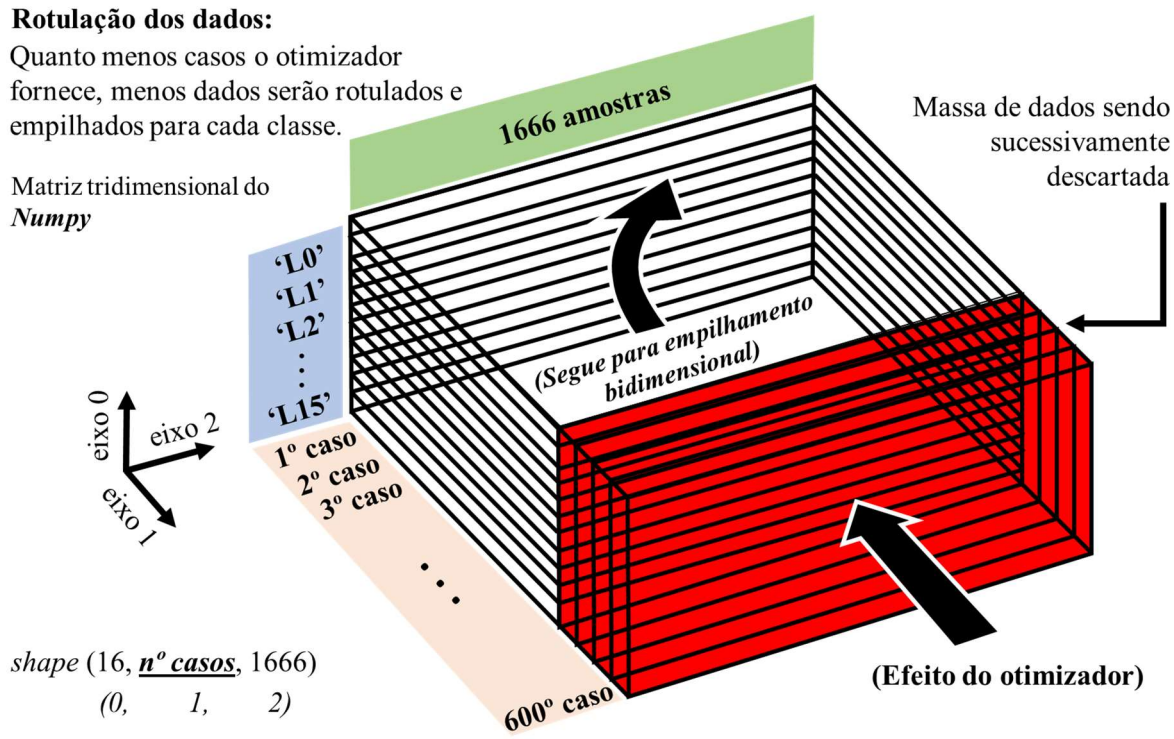
No 1º estágio, os dados, estruturados numa matriz virtual tridimensional, são empilhados em função das classes e dos casos, isto é, as amostras são organizadas em "lâminas virtuais", uma sobre a outra. Cada lâmina contém as amostras de uma classe (atributo) específica(o).

A Figura 30 indica que o efeito do otimizador será sobre o eixo 1 (*casos*), em que, a cada rodada, uma quantidade menor de casos estará disponível para rotulação e, consequentemente, para treinamento pela CNN. Após este processo, inicia-se a criação dos rótulos (*labels*), o que possibilitará que a rede desenvolva o treinamento supervisionado, finalizando com o empilhamento vertical dos dados agora agrupados em classes.

O processo de empilhamento vertical dos dados agrupados em classes é ilustrado em detalhes na Figura 31 que, sob um ponto de vista de continuidade na modelagem, indica que a

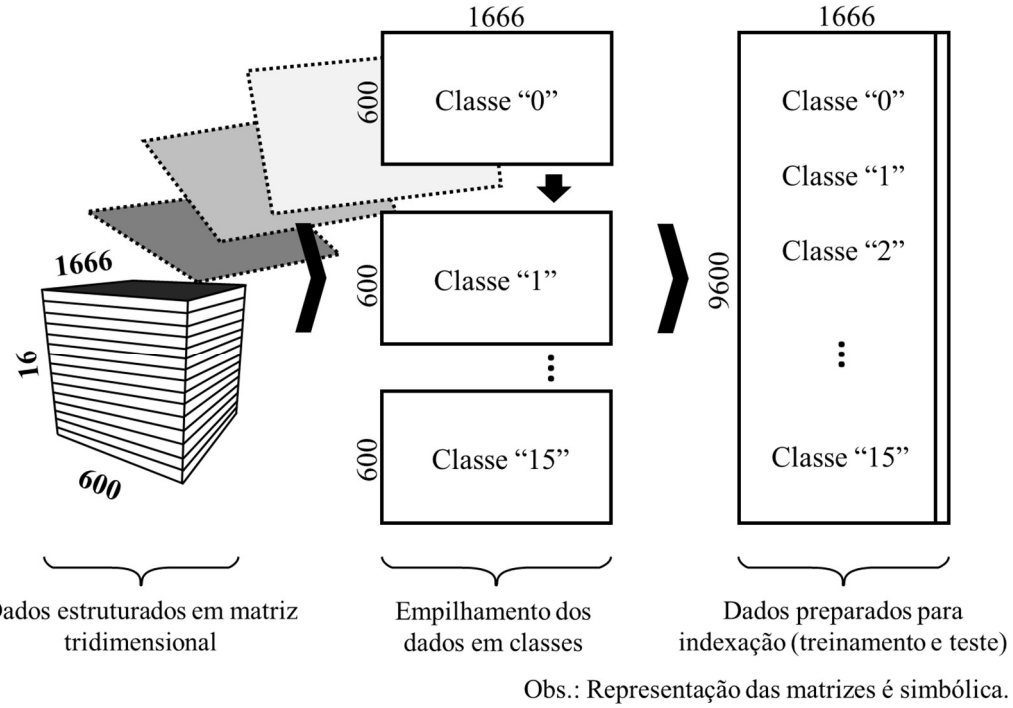
massa tridimensional de dados se reorganiza, recursivamente, em uma matriz bidimensional, partindo da Classe “15” (base), até a Classe “0” (topo).

Figura 30 – Mecanismo da redução automatizada de dados.




Fonte: (Autor, 2021)

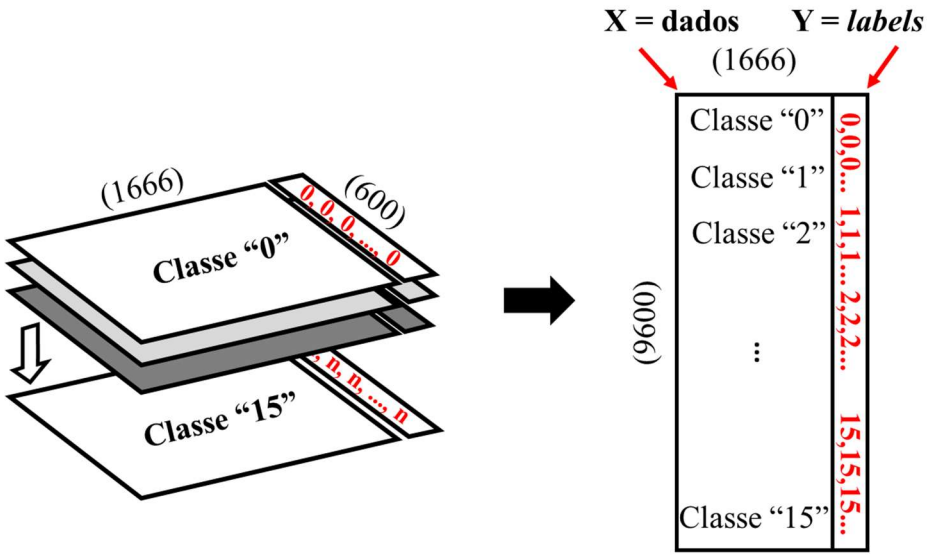
Figura 31 – Empilhamento bidimensional das classes de dados.




Fonte: (Autor, 2021)

Já a Figura 32 demonstra, de forma complementar à Figura 30, o processo de implementação dos *labels* através do recurso de empilhamento horizontal, no momento em que são "acoplados" às suas respectivas classes (atributos). Ao final deste processo, são instanciadas as variáveis "*X*" e "*Y*" para que recebam os dados e os *labels*, respectivamente.

Figura 32 – Rotulação e empilhamento das classes de dados.




Fonte: (Autor, 2021)

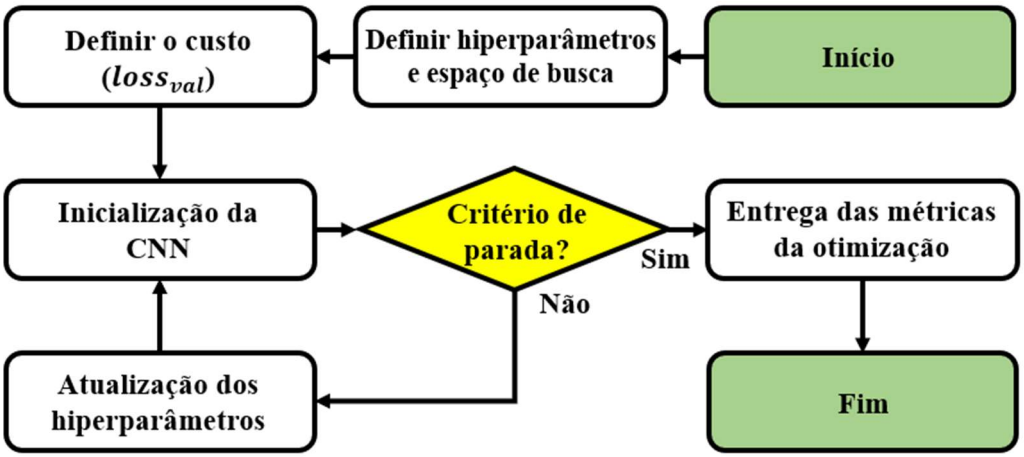
## 4.5 Segundo Estágio de Otimização

De acordo com o que foi apresentado na Seção 3.4, durante o 2º estágio de otimização é aplicado um algoritmo que implementa a otimização bayesiana para o ajuste dos hiperparâmetros. O *framework* utilizado é o *Scikit-optimize*.

A arquitetura desta etapa é apresentada no fluxograma da Figura 33. O processo é constituído pela determinação dos hiperparâmetros do modelo sob teste e a faixa de variação de cada um; e a definição da função de custo para avaliação do otimizador e inicialização do modelo neural. A cada rodada de treinamento, aqui designada como *call*, o algoritmo testa novos valores para os hiperparâmetros selecionados e computa a pontuação (*score*), de modo que este valor seja minimizado.

Assim que o número de *calls* for atingido, o sistema finaliza o processo e entrega o histórico da pesquisa: curva de *fitness*, exploração do espaço de busca e tempos de treinamento.

Figura 33 – Arquitetura do 2º estágio.




Fonte: (Autor, 2020)

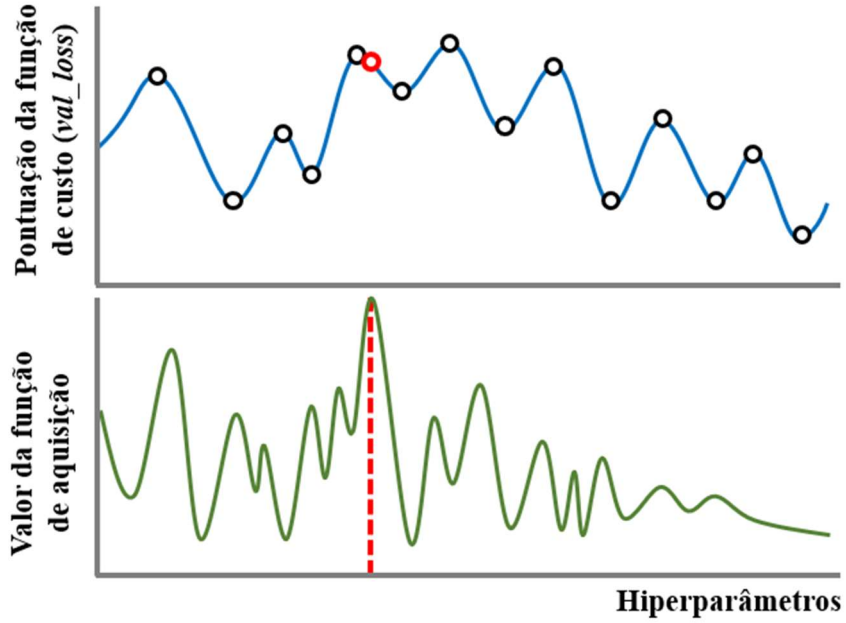
Este estágio, portanto, possui o objetivo de testar diferentes valores para os hiperparâmetros previamente selecionados, para que sejam autonomamente ajustados em função das métricas de desempenho do modelo, neste caso, o cálculo da perda nas amostras de validação (*val_loss*).

Mais especificamente, assim que o sistema é inicializado, o otimizador cria uma função substituta (*surrogate function*) da verdadeira função objetivo (*fitness function*). A *surrogate function* é uma representação probabilística para se atingir determinado *score* (a cada iteração) dado um valor do hiperparâmetro testado, no sentido de aproximá-la, cada vez mais, à

verdadeira *fitness function*. Após cada *call*, isto é, a cada treinamento da CNN, o otimizador atualizará os novos pontos da *surrogate function* em função do custo, que neste caso foi configurado como o *val_loss* da CNN.

Assim como apresentado na Seção 3.4, para que o otimizador determine qual o próximo ajuste no conjunto de hiperparâmetros, uma função de aquisição do tipo *Expected Improvement – EI* (Melhoria Esperada) é implementada. Tal função é construída a partir de um processo gaussiano (*GPminimize*), que calcula a probabilidade de uma pontuação (*score*) mais baixa ocorrer, dado o hiperparâmetro a ser testado. O ponto em que a função de aquisição é maximizada representa o valor do hiperparâmetro a ser testado na próxima rodada, conforme pode ser visualizado na Figura 34.

Figura 34 – Representação da seleção e hiperparâmetros para teste.




Fonte: (Autor, 2021)

Portanto, para cada hiperparâmetro (dimensão) selecionado, há uma respectiva função de aquisição calculada pelo otimizador, compondo o novo conjunto de soluções a ser testado na *call* seguinte.

## 4.6 Conjunto de Hiperparâmetros para o Ajuste Automatizado

Entre todos os hiperparâmetros existentes em uma rede neural convolucional, em geral, destacam-se aqueles que caracterizam sua arquitetura, tanto nas operações de convolução,

quanto nas de classificação dos dados, isto é, as camadas convolucionais e as camadas totalmente conectadas (ANDONIE; FLOREA, 2020).

A quantidade de camadas convolucionais e de neurônios em tais camadas, assim como a quantidade de camadas densas e seus respectivos neurônios, têm um impacto significativo na quantidade de parâmetros da rede, na estabilidade durante treinamento/validação, no tempo de processamento e na acurácia de classificação, como evidenciado em (FIRMES, 2020). Portanto, tais hiperparâmetros (número de camadas convolucionais, número de camadas densas, número de neurônios em camada convolucional e número de neurônios em camada densa), os quais foram manual e empiricamente ajustados em (FIRMES, 2020), são, neste trabalho, incluídos no conjunto de hiperparâmetros para ajuste automático. A Tabela 6 apresenta as respectivas faixas de variação.

Tabela 6 – Conjunto de hiperparâmetros e faixas de variação.

| **Hiperparâmetro** | **Designação** | **Faixa de variação** |
|---|---|---|
| Camadas convolucionais | *num_conv_layers* | (2:4) |
| Neurônios nas camadas convolucionais | *num_conv_nodes* | (32:256) |
| Camadas densas | *num_dense_layers* | (1:3) |
| Neurônios nas camadas densas | *num_dense_nodes* | (16:256) |

Fonte: (Autor, 2021)

Além disso, são impostas algumas restrições para diminuição do tempo de convergência do otimizador bayesiano:

- Todas as camadas convolucionais devem possuir o mesmo número de neurônios;
- As camadas densas (com exceção da camada de saída) devem possuir o mesmo número de neurônios.
- A camada de saída deve possuir 16 neurônios.

Observa-se que o número de camadas densas é propositalmente restringido para variação entre 1 e 3 camadas, considerando que a arquitetura CNN de referência possui um total de quatro camadas densas, sendo a última utilizada para a classificação das dezesseis classes dos conjuntos de dados $A_0$ e B, por isso ela não será ajustada.

## 4.7 Definição das Funções de Custo

De acordo com o que foi abordado nas Seções 3.2 e 3.4, os métodos de otimização utilizados neste trabalho necessitam de um parâmetro que sirva para avaliação (custo) das soluções contidas no espaço de busca. A função de custo, portanto, servirá como um critério matemático que conduzirá o otimizador no intento de identificar uma boa solução para o problema designado em cada estágio do experimento.

### 4.7.1 Função de Custo do 1º Estágio

O tamanho do *dataset* provoca um impacto direto na capacidade de treinamento da rede (GÉRON, 2017), de modo que, quanto maior o número de casos disponíveis, melhor será a chance de se alcançar um sistema que atenda aos requisitos. Por outro lado, também pode causar um impacto negativo, pois, quanto maior for o volume de dados a serem processados, maior será o custo computacional percebido, principalmente devido ao tempo e ao *hardware* empregados para se obter uma solução.

No intento de reduzir o esforço computacional atribuído ao experimento, percebe-se, no 1º estágio, a existência de um problema de otimização no sentido de minimizar, estrategicamente, o tamanho do *dataset*. Neste trabalho é designada a própria acurácia de classificação nos dados de teste como o custo do otimizador, por ser uma métrica amplamente empregada para avaliar o quão precisa e robusta é uma dada RNA.

É definido o espaço de busca através de um vetor de número de casos na forma

$$x = [x_{máx},\ x_{máx} - \Delta x,\ x_{máx} - (2\Delta x),\ x_{máx} - (3\Delta x), \dots,\ x_{min} + \Delta x,\ x_{min}] \qquad (4.1)$$

em que $x_{máx}$ é quantidade total de casos disponíveis na base de dados (600 casos para cada classe), $x_{min}$ é a quantidade mínima de casos disponíveis e $\Delta x$ (um número inteiro) é o passo de variação adotado, com $1 \leq \Delta x \leq (x_{máx} - x_{min})$. Estabeleceu-se empiricamente que um $\Delta x$ de 10 casos por rodada é suficiente para a execução do experimento, frente às possíveis limitações de tempo e de *hardware* (instabilidade de conexão e limitações do serviço em *cloud*).

O custo de otimização é definido como

$$f_{custo} = acc_{teste} = \frac{VP+VN}{VP+VN+FP+FN} \times 100\%, \qquad (4.2)$$

em que *VP* é um verdadeiro positivo (a classe real é positiva e foi prevista como positiva), *VN* é um verdadeiro negativo (a classe real é negativa e foi prevista como negativa), *FP* é um falso positivo (a classe real é negativa e foi prevista como positiva) e *FN* é um falso negativo (a classe real é positiva e foi prevista como negativa).

O objetivo, portanto, é minimizar $f(x_i)$, tal que $acc_{teste}(x_i) \geq 95\%$

$$f_{custo}(x_i) = \frac{VP(x_i)+VN(x_i)}{VP(x_i)+VN(x_i)+FP(x_i)+FN(x_i)} \times 100\%, \quad (4.3)$$

Em que $x_i$ é o i-*ésimo* número de casos do vetor de número de casos $\boldsymbol{x}$.

### 4.7.2 Função de Custo do 2º Estágio

Durante o processo de compilação de um modelo de RNA, o otimizador desempenha um importante papel no aprendizado dos dados. Como citado anteriormente, em (FIRMES, 2020) foi utilizado o otimizador SGD (*Stochastic Gradient Descent*). Tal otimizador utiliza a função de perda (*loss function*) como parâmetro numérico para guiar o processo de aprendizado (GÉRON, 2017).

Essencialmente, perda (*loss*) é uma penalidade para uma previsão ruim, isto é, um valor que indica o baixo desempenho de previsão de um modelo em um exemplo de treinamento. Caso a previsão não contenha erros, a perda será nula, caso contrário, assumirá algum valor. O objetivo, portanto, é que o otimizador minimize a função de perda (GOOGLE, 2021).

Uma variedade de *loss functions* estão disponíveis para compilação de RNAs. Contudo, em problemas de classificação com múltiplas classes, recomenda-se utilizar a função de perda denominada *categorical crossentropy* (entropia cruzada categórica) (KERAS, 2021c).

Segundo (BISHOP, 2006), o cálculo da perda através de entropia cruzada para tarefas de classificação com múltiplos *labels* é dado por

$$loss = -\sum_{i=1}^{N}\sum_{j=1}^{K} T_{i,j} \ln(Y_{i,j}), \quad (4.4)$$

em que *N* é o número de observações durante o treinamento, *K* é o número de classes, $T_{i,j}$ é o indicador de que a i-*ésima* amostra pertence à j-*ésima* classe, e $Y_{i,j}$ é a saída da amostra *i* para a classe *j*, que, neste caso, é o valor da função *Softmax*, conforme Equação 2.3.

Portanto, haja vista que para haver aprendizado a perda deve ser a menor possível, tem-se aí um problema de minimização, em que a função de custo do 2º estágio de otimização poderá ser representada como

$$f_{custo} = loss_{validation} = -\sum_{i=1}^{N}\sum_{j=1}^{K} T_{i,j}\ln\left(Y_{i,j}\right), \tag{4.5}$$

observando que a perda (*loss*) computada como custo, será designada como aquela calculada sobre as amostras de validação (*validation*), uma vez que possibilitam maior confiabilidade no monitoramento do desempenho da RNA (GÉRON, 2017).

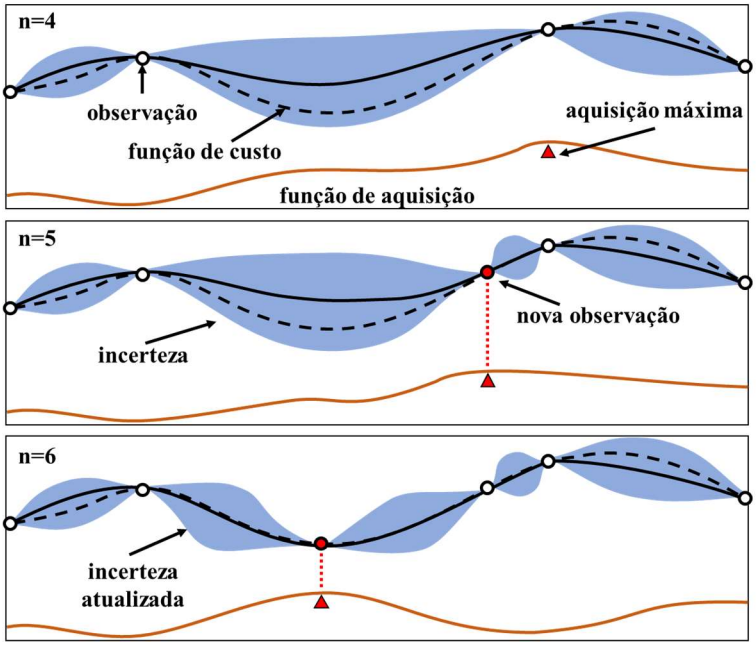
Ainda, oposto à função de custo do 1º estágio, em que o caminho a percorrer no espaço de busca é determinado por um vetor de número de casos, no 2º estágio o espaço de busca é baseado em um processo gaussiano e restringido por um limite inferior e superior, tal como apresentado na Tabela 6, e explorado através das operações matemáticas do otimizador bayesiano, já apresentadas na Seção 3.4 e ilustradas na Figura 35.

A função de aquisição é comumente maximizada nas regiões do espaço de busca em que a função substituta apresenta maior incerteza, tornando o otimizador apto a explorar regiões com potenciais soluções.

Contudo, haja vista que o processo de adequação integral da função substituta à verdadeira função de custo pode requerer um tempo e esforço computacional que torne o experimento extremamente custoso, faz-se necessário a consideração de restrições.

Prevendo possíveis instabilidades de conexão ou a restrição de recursos de GPU/memória RAM disponibilizados pelo servidor do *Google Colab*, a quantidade de iterações do otimizador com a CNN ($n_{calls}$) é limitada, empiricamente, para $n_{calls} = 100$ em ambas as bases de dados, $A_0$ e B. Será relevante, com essa restrição estabelecida, comparar o desempenho do otimizador com os dois tipos de cargas altamente similares, lâmpadas e computadores.

Figura 35 – Exploração do espaço de busca – 2º estágio.




Fonte: (Autor, 2021)

## 4.8 Avaliação Final da Arquitetura Otimizada

Com o objetivo de verificar a robustez das informações fornecidas através dos experimentos aqui detalhados, é implementada uma etapa final no processo automatizado de busca do tamanho da base de dados necessária e dos hiperparâmetros selecionados da CNN. Essa etapa consistirá num novo teste com os ajustes encontrados nos dois estágios de otimização.

Portanto, a arquitetura da rede será reconstruída, desta vez, com a quantidade de camadas convolucionais e densas (assim como seus respectivos neurônios) encontradas no 2º estágio de otimização e o tamanho da base de dados utilizada será em função do novo número de casos, encontrado durante o 1º estágio de otimização.

As métricas utilizadas para a avaliação deste modelo final serão semelhantes às usadas nas etapas anteriores, isto é:

- Percentual final do tamanho das bases de dados;
- Quantidade de parâmetros totais das redes;
- Quantidade de épocas necessárias para treinamento;
- Acurácia nas amostras de teste.

# 5 Resultados e Discussão

Neste Capítulo serão apresentados os resultados obtidos ao conduzir os experimentos tal como descrito na Metodologia. Optou-se por agrupar as informações em quatro subseções (5.1, 5.2, 5.3 e 5.4), sendo as duas primeiras para apresentar resultados do 1º e 2º estágio, respectivamente. A terceira subseção é dedicada à investigação de uma possível relação entre a redução de casos com os parâmetros totais da rede. Finalmente, a quarta subseção apresenta a discussão sobre a capacidade de generalização da CNN treinada com um número de casos reduzido utilizando, para isso, um conjunto de amostras de teste referente ao tamanho da base de dados completa.

## 5.1 Resultados do 1º Estágio

### 5.1.1 Otimização do Conjunto de Dados $A_0$

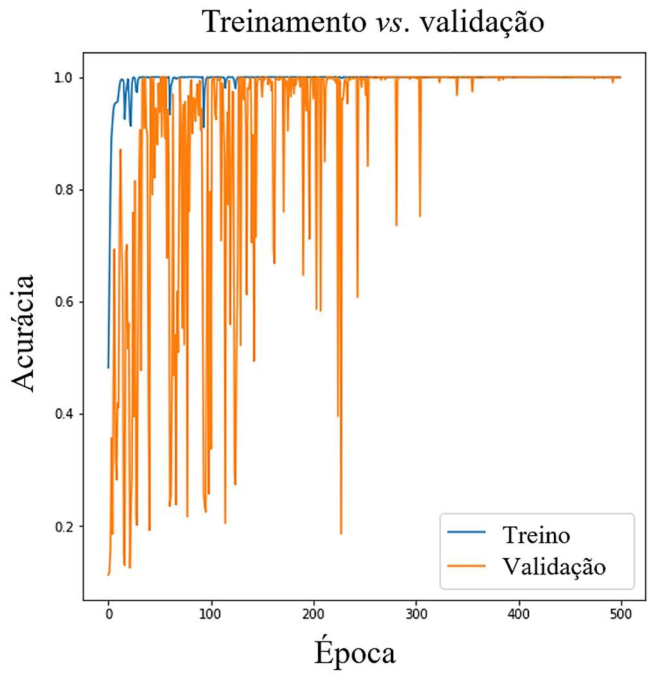
Para a redução automatizada do *dataset*, foi aplicado o otimizador denominado *Hill Climbing*, aqui designado como "descida de encosta". Visando varrer o espaço de busca balanceando tempo *vs.* pontos amostrados, foi estabelecido um passo de redução da base de dados em 10 casos. Após a inicialização do programa, o otimizador executou o modelo sucessivas vezes, reduzindo a quantidade de casos disponíveis para cada uma das 16 classes.

Entretanto, verificou-se uma instabilidade no processo de treinamento/validação conforme pode ser visto na Figura 36 o que, por vezes, culminava em uma baixa acurácia de teste, forçando o otimizador a ficar preso em uma solução mínima local. Para resolver tal problema, aplicou-se uma redução gradativa do hiperparâmetro denominado Taxa de Aprendizado, dada pela Equação 5.1, até que o sistema mostrasse uma estabilidade de treinamento satisfatória.

$$lr = lr_0 * decay\ rate^{\left(\frac{step}{decay\ steps}\right)}, \qquad (5.1)$$

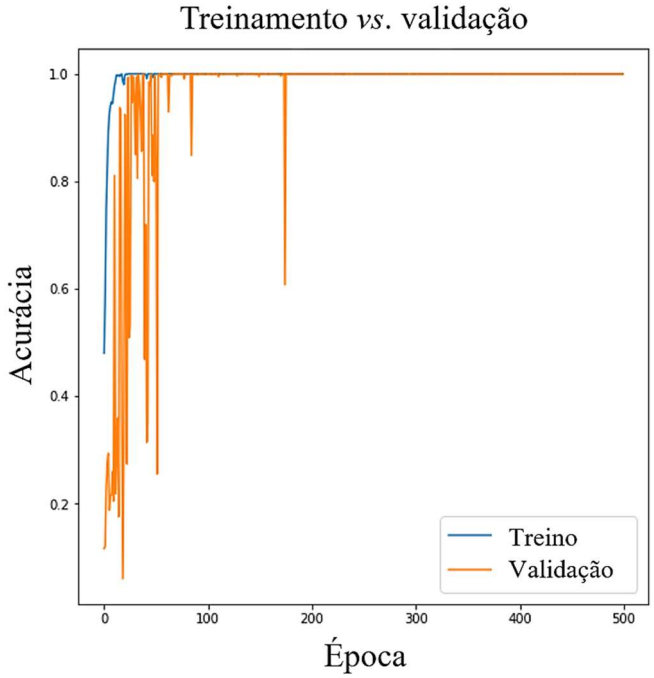
Em que *lr* é a taxa de aprendizado, $lr_0$ é a taxa de aprendizado inicial, *decay rate* é a taxa de decaimento, *step* é o passo que o otimizador SGD executa para minimizar a perda durante o aprendizado e *decay steps* é o tamanho do passo. A Figura 37 mostra o resultado do treinamento do sistema para uma taxa de aprendizagem ajustada utilizando $lr_0 = 0{,}01$; *decay rate* = 0,95; e *decay steps* = 50, em que se pode perceber um treinamento estável.

Figura 36 – Treinamento (sem ajuste da taxa de aprendizagem).




Fonte: (Autor, 2021)

Figura 37 – Treinamento (com ajuste da taxa de aprendizagem).



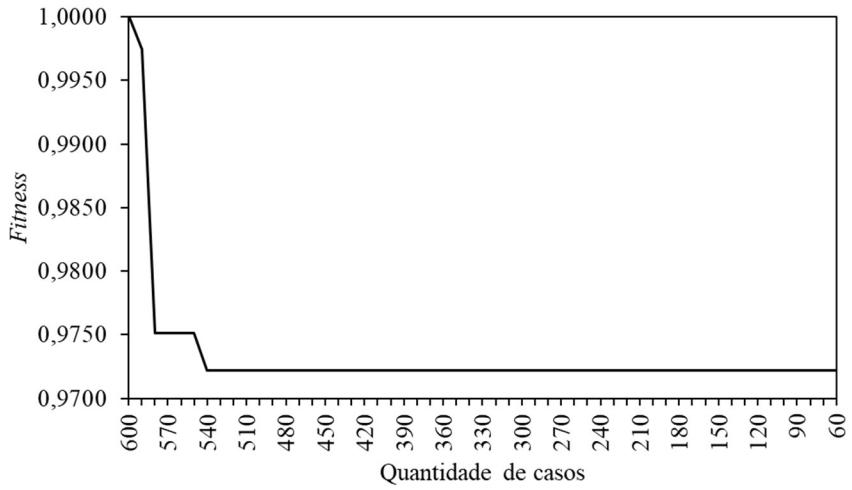

Fonte: (Autor, 2021)

A Figura 38 indica a curva de *fitness* (ou custo) apresentada pelo otimizador. Em outras palavras, este tipo de gráfico mostra para cada rodada de redução no número de casos, o menor valor encontrado desde o início do processo, evidenciando assim a minimização do custo, isto

é, da acurácia nas amostras de teste, tal como delineado durante a Metodologia. Nota-se que o otimizador foi capaz de alcançar o objetivo considerando as restrições impostas.

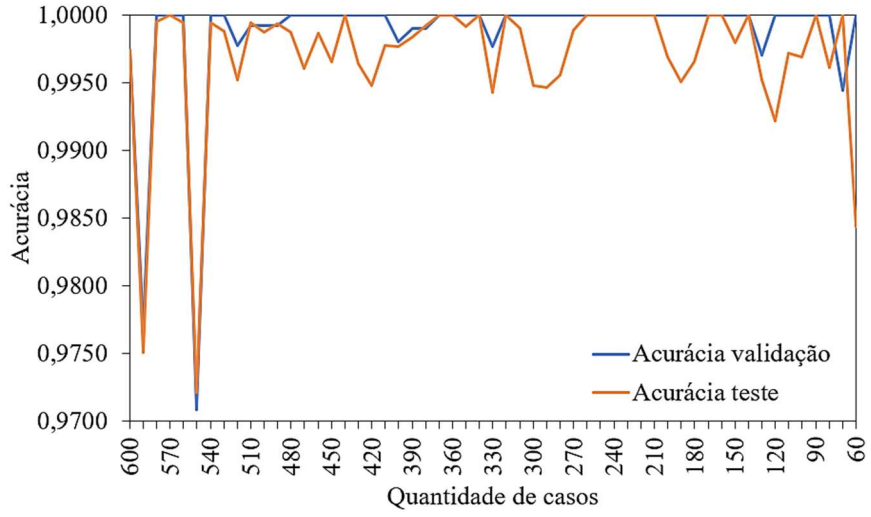
A parada desse processo iterativo ocorreu quando o critério de parada denominado "limite de casos" foi atingido, isto é, logo após o treinamento da rede com 60 casos. Percebe-se ainda que a acurácia de validação e teste se mantiveram acima do limite mínimo pré estabelecido de 95%, como mostra a Figura 39.

Figura 38 – Curva de *fitness* – base de dados $A_0$.




Fonte: (Autor, 2021)

Figura 39 – Acurácia de validação *vs.* teste durante a otimização do conjunto de dados $A_0$.

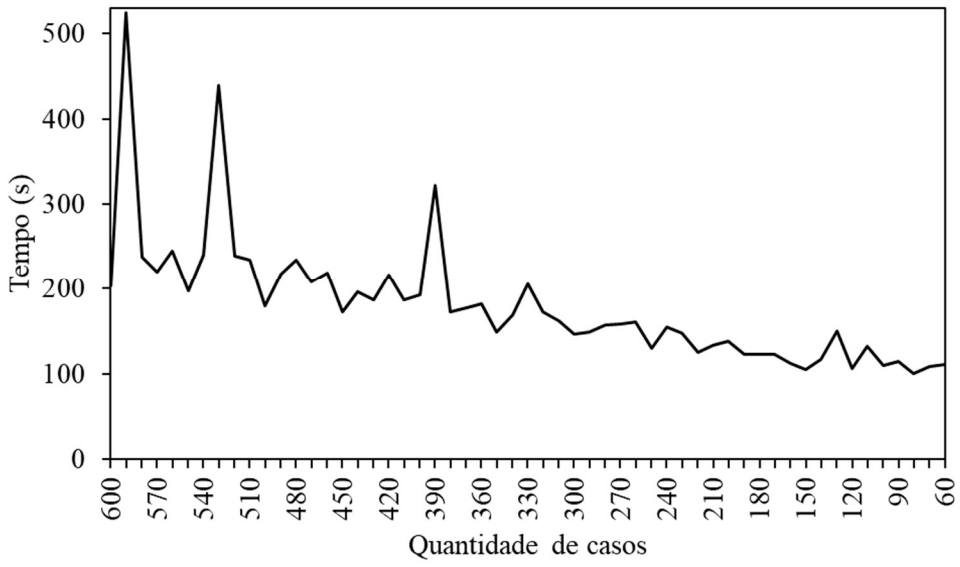



Fonte: (Autor, 2021)

Ainda na Figura 39, é possível perceber algumas variações mais acentuadas nos extremos do espaço de busca, o que pode ser justificado pela natureza estocástica do processo. Em termos globais de acurácia, o modelo obteve um desempenho bastante satisfatório (acima de 97%), mesmo com a redução agressiva do conjunto de dados.

Já a Figura 40 apresenta o histórico de tempos de treinamentos gastos em cada iteração do otimizador. Observa-se na figura alguns picos que podem ser justificados pela natureza estocástica do sistema sob treinamento. Contudo, fica clara a tendência de redução nos tempos de treinamento, conforme esperado, uma vez que o conjunto de dados foi sendo sucessivamente reduzido durante a ação do otimizador. O processo de descida de encosta foi concluído em, aproximadamente, 2h45m.

Figura 40 – Tempos de treinamentos durante a otimização do conjunto de dados $A_0$.



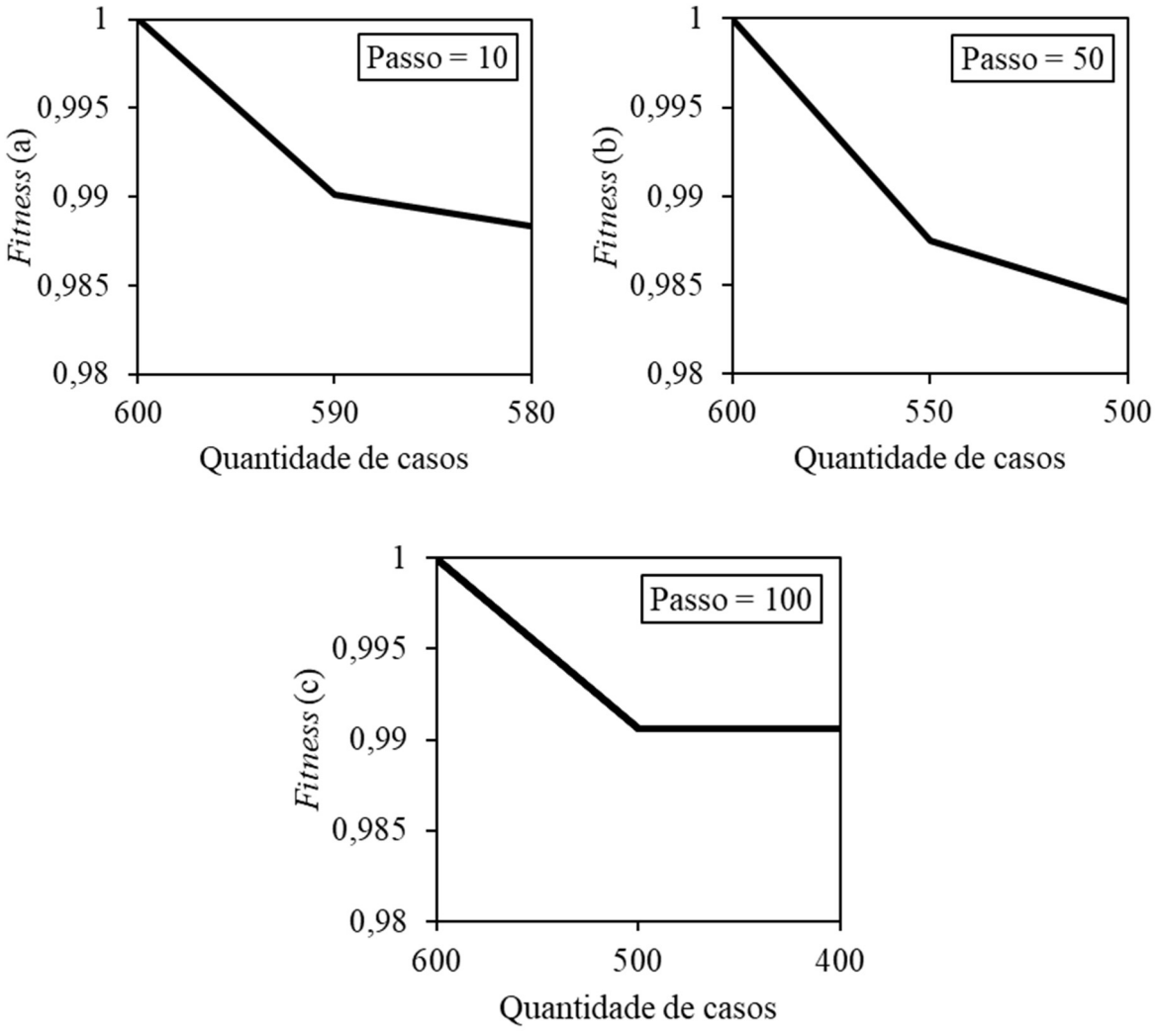

Fonte: (Autor, 2021)

## 5.1.2 Otimização do Conjunto de Dados B

No caso do conjunto de dados referente aos microcomputadores, representando as cargas de variação não determinística, foi detectado um comportamento distinto se comparado à base de dados $A_0$. Neste experimento não foi necessária a modificação do hiperparâmetro taxa de aprendizagem, pois percebeu-se que neste caso, tal procedimento provocava uma degradação da estabilidade durante o treinamento da CNN de referência. Assim, a taxa de aprendizagem foi mantida em 0,01.

O passo de redução, $\Delta x$, foi inicialmente ajustado para 10 casos, pois percebeu-se que o otimizador ficava constantemente preso numa região do espaço de busca que compreende as iterações referentes à faixa de 580, conforme mostrado na Figura 41a. Assim, o $\Delta x$ foi aumentado para 50 casos, o que melhorou o espaço de busca para 500 casos, como pode ser visto na Figura 41b. Por fim, o $\Delta x$ foi ajustado para 100 casos, resultando em mais uma melhora no espaço de busca, como mostra a Figura 41c. Já a Figura 42 permite uma melhor comparação do desempenho do otimizador com o $\Delta x$ assumindo diferentes valores.

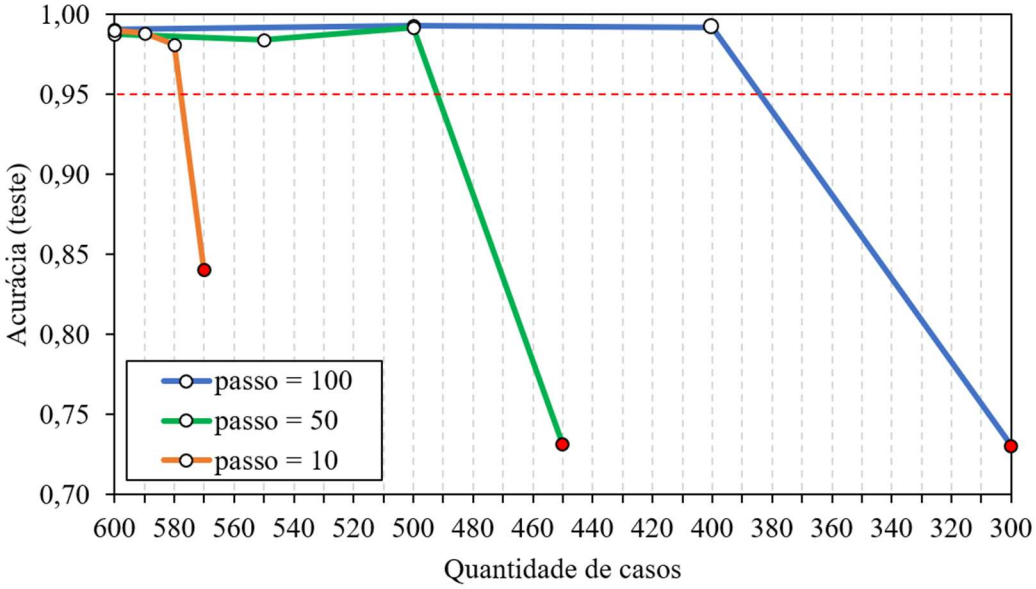
Figura 41 – Curvas de *fitness* – base de dados B.




Fonte: (Autor, 2021)

Através do histórico das acurácias de teste, é possível comparar os experimentos realizados com $\Delta x$ de 10, 50 e 100 casos, sendo o último o que se mostrou mais promissor na tarefa de encontrar a maior redução possível, neste caso, para 400 casos (66,6% do tamanho total), com acurácia de classificação em 99,21%, conforme mostrado na Figura 42.

Figura 42 – Acurácias de teste em função do *passo* de redução – base de dados B.

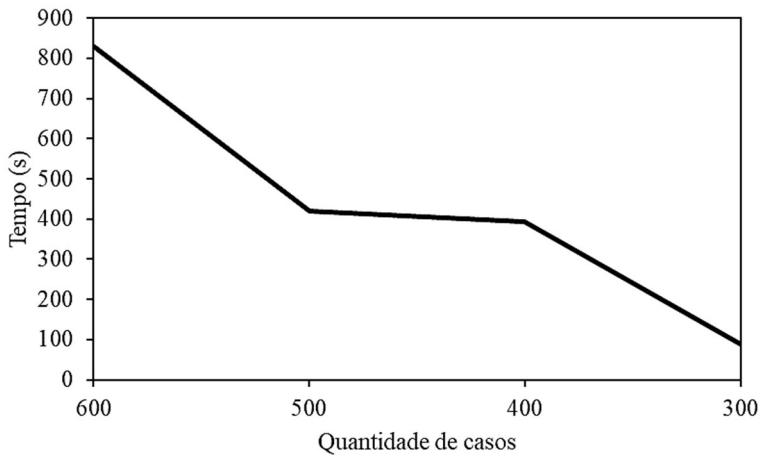



Fonte: (Autor, 2021)

Finalmente, a Figura 43 apresenta os tempos gastos durante o processo de otimização do conjunto de dados B, usando-se um $\Delta x$ igual a 100 casos. Percebe-se uma diminuição do conjunto de dados B de 600 para 400 casos (suficientes para garantir uma acurácia superior ao mínimo estabelecido de 95%, conforme pode ser visto na Figura 42), isto é, uma redução de cerca de 33%, capaz de reduzir pela metade o tempo de treinamento do sistema.

Por fim, cabe ressaltar que todo o procedimento de otimização do conjunto de dados B, considerando o experimento configurado com $\Delta x$ = 100, foi concluído em um total de aproximadamente 0h29m.

Figura 43 – Tempos de treinamentos durante a otimização do conjunto de dados B.




Fonte: (Autor, 2021)

## 5.2 Resultados do 2º Estágio

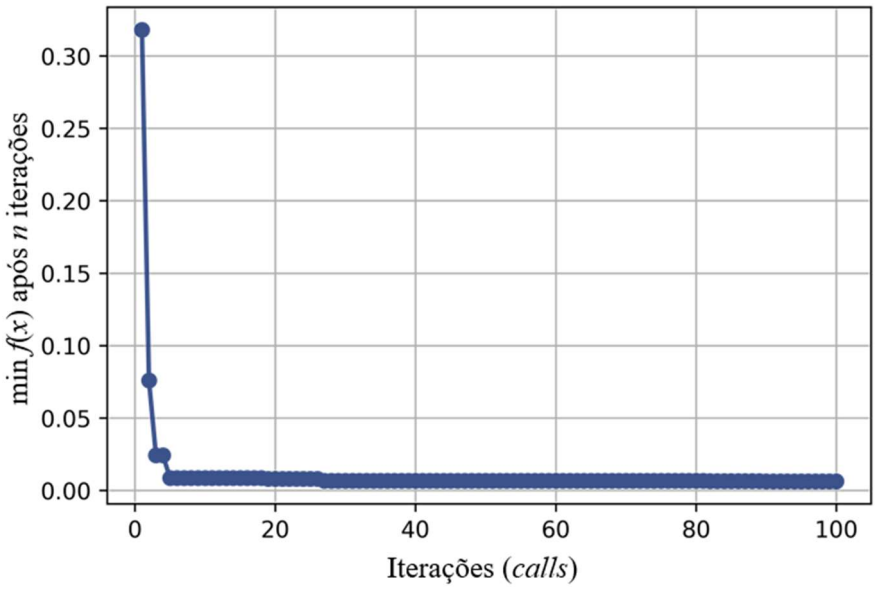
Conforme detalhado no Capítulo 4, o ajuste automatizado de alguns hiperparâmetros da arquitetura CNN #3, é feito utilizando um otimizador bayesiano. Os hiperparâmetros selecionados para tal demonstração são o número de camadas convolucionais, o número de neurônios nas camadas convolucionais, a quantidade de camadas densas e o número de neurônios nas camadas densas.

Para fins de comparação de desempenho do otimizador quando usando dois conjuntos de dados distintos, $A_0$ e B, e para possibilitar a execução do experimento em tempo não proibitivo, é estabelecido como critério de parada um valor máximo de 100 iterações ($n_{calls} = 100$).

### 5.2.1 Otimização de Hiperparâmetros Usando o Conjunto de Dados $A_0$ Reduzido

A evolução da função de custo implementada quando aplicado o otimizador bayesiano é apresentada na Figura 44. Constata-se que o otimizador conseguiu reduzir o custo de forma bastante acentuada antes das 10 primeiras iterações tendendo exponencialmente para zero à medida em que se avança no número de iterações.

Figura 44 – Curva de *fitness* durante otimização de hiperparâmetros de CNN quando usando o conjunto de dados $A_0$ reduzido ao valor ótimo.




Fonte: (Autor, 2021)

Esta otimização de hiperparâmetros foi realizada em aproximadamente 3h18m. A Tabela 7 apresenta os valores ajustados para os hiperparâmetros selecionados.

Tabela 7 – Hiperparâmetros de CNN otimizados usando o conjunto de dados $A_0$.

| Hiperparâmetro | Ajuste |
|---|---|
| Nº de camadas convolucionais | 3 |
| Nº de neurônios nas camadas convolucionais | 72 |
| Nº de camadas densas (exceto a de saída) | 3 |
| Nº de neurônios nas camadas densas (exceto na de saída) | 256 |

Fonte: (Autor, 2021)

O Apêndice C apresenta o histórico do processo de otimização e dos respectivos hiperparâmetros autonomamente ajustados. Os valores estão em ordem crescente, tal como disponibilizados pelo *framework*.

A Figura 45 demonstra o histórico de exploração realizado pelo otimizador bayesiano no interior do espaço de busca. Os pontos amostrados estão correlacionados baseados no cálculo da dependência parcial (*partial dependence*) (FRIEDMAN, 2001). Resumidamente, o objetivo é visualizar como o valor de uma variável *x* influencia a função substituta após calcular a média da influência de todas as outras variáveis. Os gráficos de linha, empilhados da diagonal superior, são unidimensionais e indicam a influência de cada um dos quatro hiperparâmetros previamente selecionados na função substituta, enquanto que os gráficos de contorno indicam a mesma influência, porém, ao comparar um conjunto de dois hiperparâmetros (SCIKIT-LEARN, 2021a).

Cabe ressaltar que o método de dependências parciais corresponde a uma estimativa probabilística capaz de demonstrar tendências de comportamento da função substituta que, por sua vez, é um modelo probabilístico da verdadeira função de custo. Tais observações sugerem que este parâmetro deve ser avaliado com cautela (HEAD et al., 2020).

Referente aos gráficos de contorno, as regiões da base (azul escuro) são aquelas que foram inicialmente exploradas pelo otimizador, enquanto que as do topo (amarelo) correspondem às iterações mais atuais. Os pontos escuros representam os valores amostrados pelo otimizador, e a estrela, destacada em vermelho, indica o melhor ponto encontrado, comparando dois hiperparâmetros diferentes em cada gráfico. Já nos gráficos de linha, tal informação está contida no ponto em que se situa a linha tracejada em vermelho.

Quanto à disposição dos dados, é importante observar que a altura dos pontos amostrados (eixo *y* nos gráficos de linha e eixo *z* nas superfícies de contorno) não representam,

necessariamente, a melhor solução encontrada e sim, os valores que tiveram maior influência (seja positiva ou negativa) no sentido de minimizar a função substituta.

A disposição dos pontos amostrados nos gráficos de contorno evidencia que o otimizador executou as iterações conforme as restrições previamente impostas ao espaço de busca, as quais culminaram na solução de ajuste apresentada na Tabela 7.

Considerando os valores de dependências parciais e comparando os gráficos de linha com os gráficos de contorno, é possível fazer as seguintes observações de comportamento do otimizador:

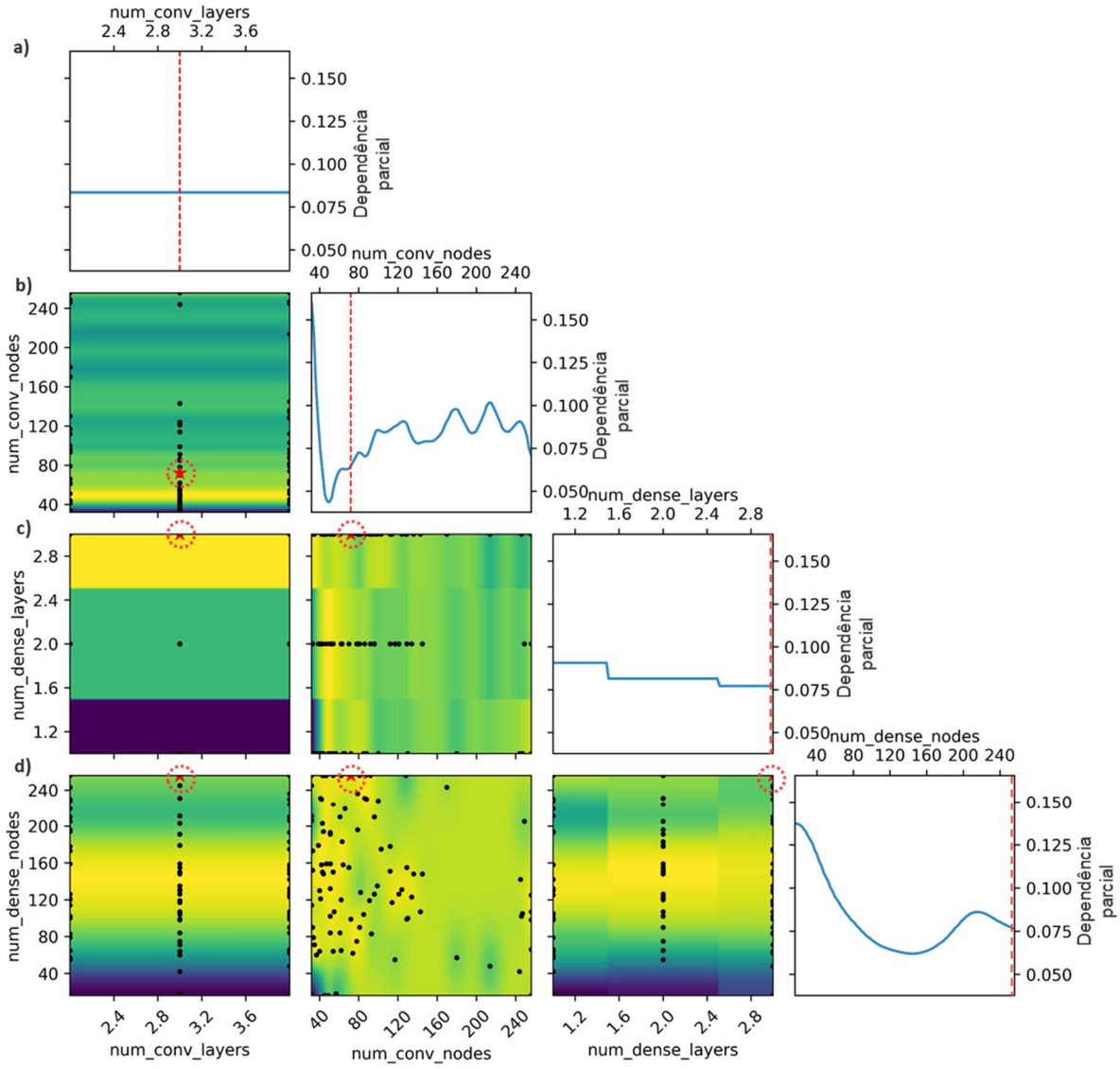
a) Avaliando-se de maneira unidimensional, o número de camadas convolucionais pouco influenciou a função substituta, além disso, a variação desse número não resultou em variações na dependência parcial.

b) A quantidade de neurônios testados nas camadas convolucionais apresentou grande variação de influência na função substituta, como pode ser visto no gráfico unidimensional (*num_conv_nodes*), contudo, ao comparar o conjunto (*num_conv_layers*; *num_conv_nodes*), o ajuste que proporcionou menor custo foi 3 camadas convolucionais e 72 neurônios em cada uma destas camadas.

c) O gráfico unidimensional mostra que a quantidade de camadas densas provocou pouca influência na função substituta, apresentando ainda um pequeno decréscimo desta influência à medida que o número de camadas aumentou. No entanto, ao comparar os conjuntos (*num_conv_layers*; *num_dense_layers*) e (*num_conv_nodes*; *num_dense_layers*), os ajustes que proporcionaram menor custo foram, respectivamente, (3 camadas convolucionais; 3 camadas densas) e (72 neurônios nas camadas convolucionais; 3 camadas densas);

d) A quantidade de neurônios testados nas camadas densas apresentou grande variação na dependência parcial, conforme apresentado no gráfico unidimensional. Além disso, o aumento deste hiperparâmetro causou, em linhas gerais, a redução de sua influência na função substituta. Quanto aos três últimos gráficos de contorno, destaca-se o do conjunto (*num_conv_nodes*; *num_dense_nodes*), em que se percebe um padrão de aglomeração de pontos testados nas regiões do espaço de busca com baixas quantidades de neurônios nas camadas de convolução, culminando no ajuste (72 neurônios nas camadas convolucionais; 256 neurônios nas camadas densas).

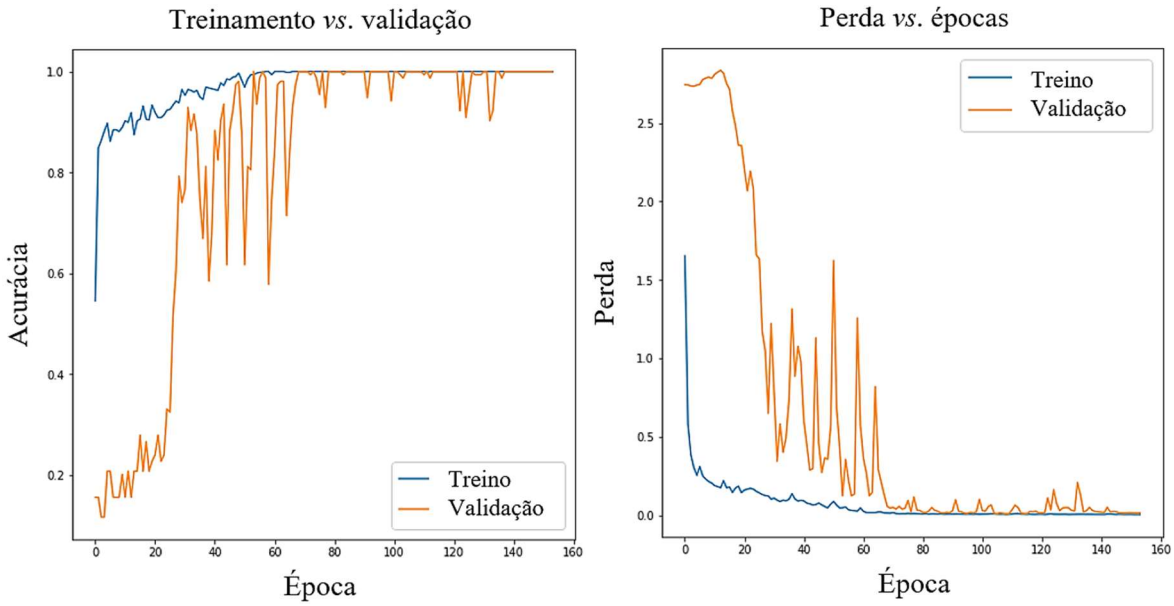
Figura 45 – Exploração do espaço de busca do otimizador bayesiano operando com o conjunto de dados $A_0$ reduzido.




Fonte: (Autor, 2021)

Seguindo o planejamento do experimento, conforme proposto no Capítulo 4, modificou-se a arquitetura da CNN conforme os novos hiperparâmetros encontrados. Os dados apresentados na Figura 46 mostram o desempenho final do modelo utilizado junto ao conjunto de dados $A_0$ reduzido a 60 casos.

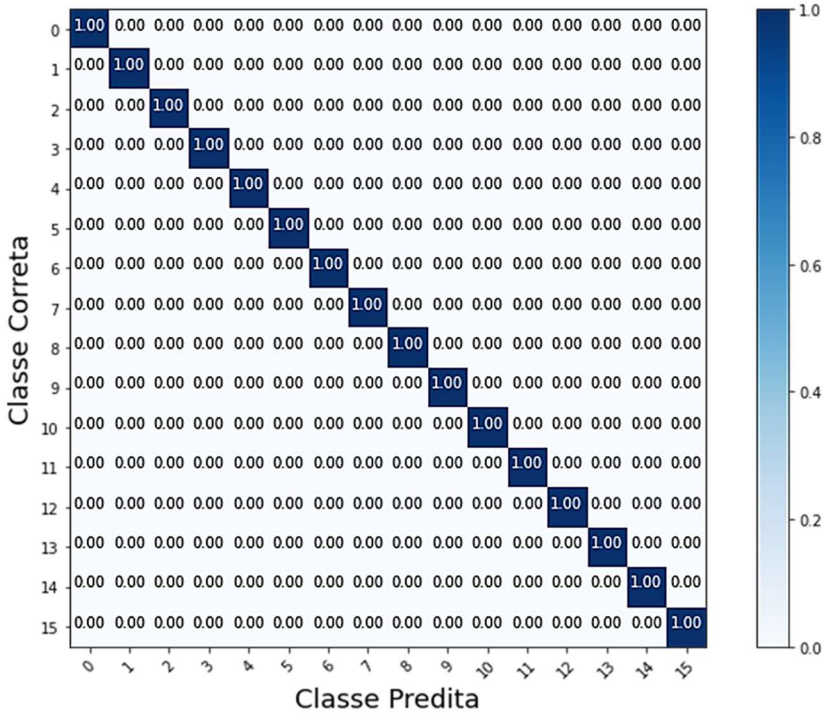
Figura 46 – Desempenho da CNN otimizada a partir do conjunto de dados $A_0$ reduzido.




Fonte: (Autor, 2021)

A Figura 47 apresenta a matriz de confusão criada a partir das amostras de teste do conjunto de dados $A_0$ reduzido. É possível observar que a arquitetura foi capaz de classificar corretamente todas as dezesseis classes.

Figura 47 – Matriz de confusão do teste realizado no conjunto de dados $A_0$ reduzido.




Fonte: (Autor, 2021)

Constata-se que a rede CNN otimizada foi treinada, validada e testada apresentado um excelente desempenho, com uma acurácia de 100% após 154 épocas de treinamento. Outro fato importante a ser destacado é que a otimização de apenas 4 hiperparâmetros da rede CNN resultou em uma redução considerável da quantidade de parâmetros totais da rede em questão, quando comparada com a rede CNN de referência proposta em (FIRMES, 2020), a qual possui 2.962.800 parâmetros totais, contra 1.303.144 parâmetros totais, o que representa uma redução em torno de 56,02% (Figura 48). Tal redução é de extrema importância, pois provoca um impacto positivo ao reduzir a complexidade do *hardware* necessário para tratar um dado problema.

A Tabela 8 apresenta algumas métricas relativas as CNNs de referência e otimizada, permitindo uma comparação direta entre ambas. A primeira é treinada com o conjunto de dados $A_0$ e a segunda com o conjunto de dados $A_0$ reduzido.

Tabela 8 – *Benchmark* CNN referência *vs.* otimizada.

| **Arquitetura da CNN** | **Amostras de corrente no *dataset*** | **Parâmetros da rede** | **Épocas de treinamento** | **Acurácia do teste (%)** |
|---|---|---|---|---|
| Referência | 15.993.600 | 2.962.800 | 500 | 99,90 |
| Otimizada | 1.599.360 | 1.303.144 | 154 | 100,00 |
| **redução (%)** | 90,00 | 56,02 | 69,20 | -0,10 |

Fonte: (Autor, 2021)

Figura 48 – Arquitetura da rede CNN otimizada.

| **Nome da camada e Tipo** | **Formato da Saída** | **# Parâmetros** |
|---|---|---|
| conv1d (*Conv1D*) | (1666, 72) | 360 |
| max_pooling1d (*MaxPooling1D*) | (555, 72) | 0 |
| batch_normalization (*BatchNormalization*) | (555, 72) | 288 |
| conv1d_1 (*Conv1D*) | (555, 72) | 20808 |
| max_pooling1d_1 (*MaxPooling1D*) | (185, 72) | 0 |
| batch_normalization_1 (*BatchNormalization*) | (185, 72) | 288 |
| conv1d_2 (*Conv1D*) | (185, 72) | 20808 |
| max_pooling1d_2 (*MaxPooling1D*) | (61, 72) | 0 |
| batch_normalization_2 (*BatchNormalization*) | (61, 72) | 288 |
| flatten (*Flatten*) | (4392) | 0 |
| dense (*Dense*) | (256) | 1124608 |
| dense_1 (*Dense*) | (256) | 65792 |
| dense_2 (*Dense*) | (256) | 65792 |
| dense_3 (*Dense*) | (16) | 4112 |

**Total de parâmetros:** 1.303.144
**Parâmetros treináveis:** 1.302.712
**Parâmetros não treináveis:** 432

Fonte: (Autor, 2021)

### 5.2.2 Otimização de Hiperparâmetros Usando o Conjunto de Dados B Reduzido

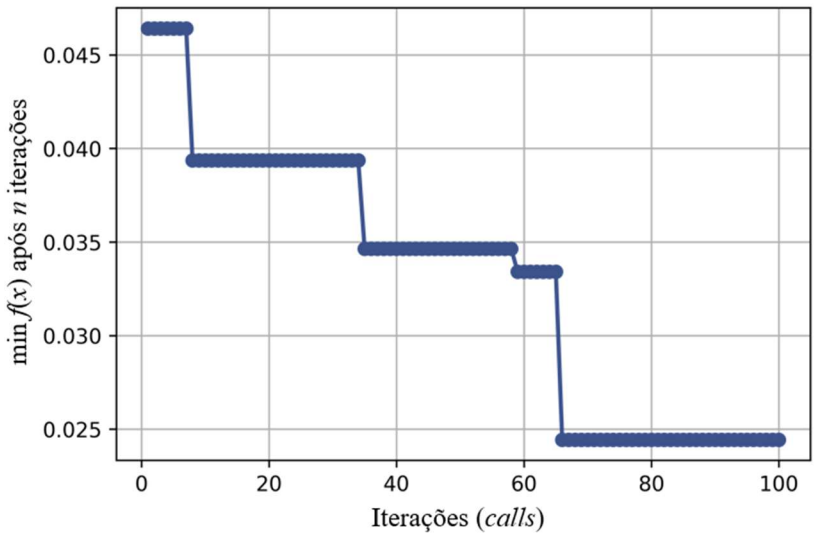
A Figura 49 apresenta a curva de *fitness* devido à otimização de parte dos hiperparâmetros da CNN de referência proposta em (FIRMES, 2020). Observa-se na figura uma redução do custo durante a validação da CNN sob otimização quando treinada com o conjunto de dados B reduzido. Pode-se observar que, ao contrário do que foi constatado em $A_0$, o otimizador só foi capaz de encontrar um valor próximo de zero após a 60ª iteração.

Este processo de otimização de hiperparâmetros de CNN limitado a 100 iterações levou cerca de 15h13m. Trata-se de um tempo maior quando comparado com os resultados da otimização de hiperparâmetros de CNN apresentados na Subseção 5.2.1. Tal aumento pode ser justificado principalmente por dois fatores:

a) O conjunto de dados B reduzido possui um número superior de casos, que reflete em um custo computacional também superior;

b) O conjunto de dados B reduzido apresentou pequenos incrementos na acurácia de validação mais bem distribuídos em cada treinamento. Como este foi o gatilho de acionamento da regularização do tipo *Early Stopping*, as rodadas de treinamento continham mais épocas do que se comparado com o conjunto de dados $A_0$ reduzido.

Figura 49 – Curva de *fitness* durante otimização de hiperparâmetros de CNN quando usando o conjunto de dados B reduzido ao valor ótimo.




Fonte: (Autor, 2021)

A Tabela 9 apresenta os valores ajustados para os hiperparâmetros selecionados no segundo estágio do experimento.

Tabela 9 – Hiperparâmetros de CNN otimizados usando o conjunto de dados B reduzido.

| Hiperparâmetro | Ajuste |
|---|---|
| Nº de camadas convolucionais | 4 |
| Nº de neurônios nas camadas convolucionais | 64 |
| Nº de camadas densas (exceto a de saída) | 1 |
| Nº de neurônios nas camadas densas (exceto na de saída) | 180 |

Fonte: (Autor, 2021)

A Figura 50 demonstra o histórico de exploração do espaço de busca tal como realizado na Subseção 5.2.1. Considerando novamente os valores de dependências parciais e, comparando os gráficos de linha com os gráficos de contorno, é possível fazer as seguintes observações de comportamento do otimizador:

a) Assim como observado na Subseção 5.2.1, a quantidade de camadas convolucionais testadas pouco influenciaram a função substituta, não havendo, inclusive, variação na sua influência. Observando de forma unidimensional, o valor que proporcionou melhor ajuste foi de 4 camadas convolucionais;

b) As quantidades de neurônios testadas nas camadas de convolução demonstraram um pequeno ruído com baixa variação de sua influência na função substituta, exceto pelo pico demonstrado próximo dos 240 neurônios. Considerando o conjunto (*num_conv_layers*; *num_conv_nodes*), a melhor solução encontrada foi de 4 camadas convolucionais com 64 neurônios em cada camada;

c) De forma unidimensional, a quantidade de camadas densas, assim como observado na Subseção 5.2.1, provocou pouca influência na função substituta e o aumento deste valor provocou leve decréscimo de sua influência. Considerando-se os pares (*num_conv_layers*; *num_dense_layers*) e (*num_conv_nodes*; *num_dense_layers*), os melhores ajustes foram identificados com as soluções (3 camadas convolucionais e 1 camada densa – sem contar a de saída) e (64 neurônios nas camadas convolucionais e 1 camada densa – sem contar a de saída), respectivamente;

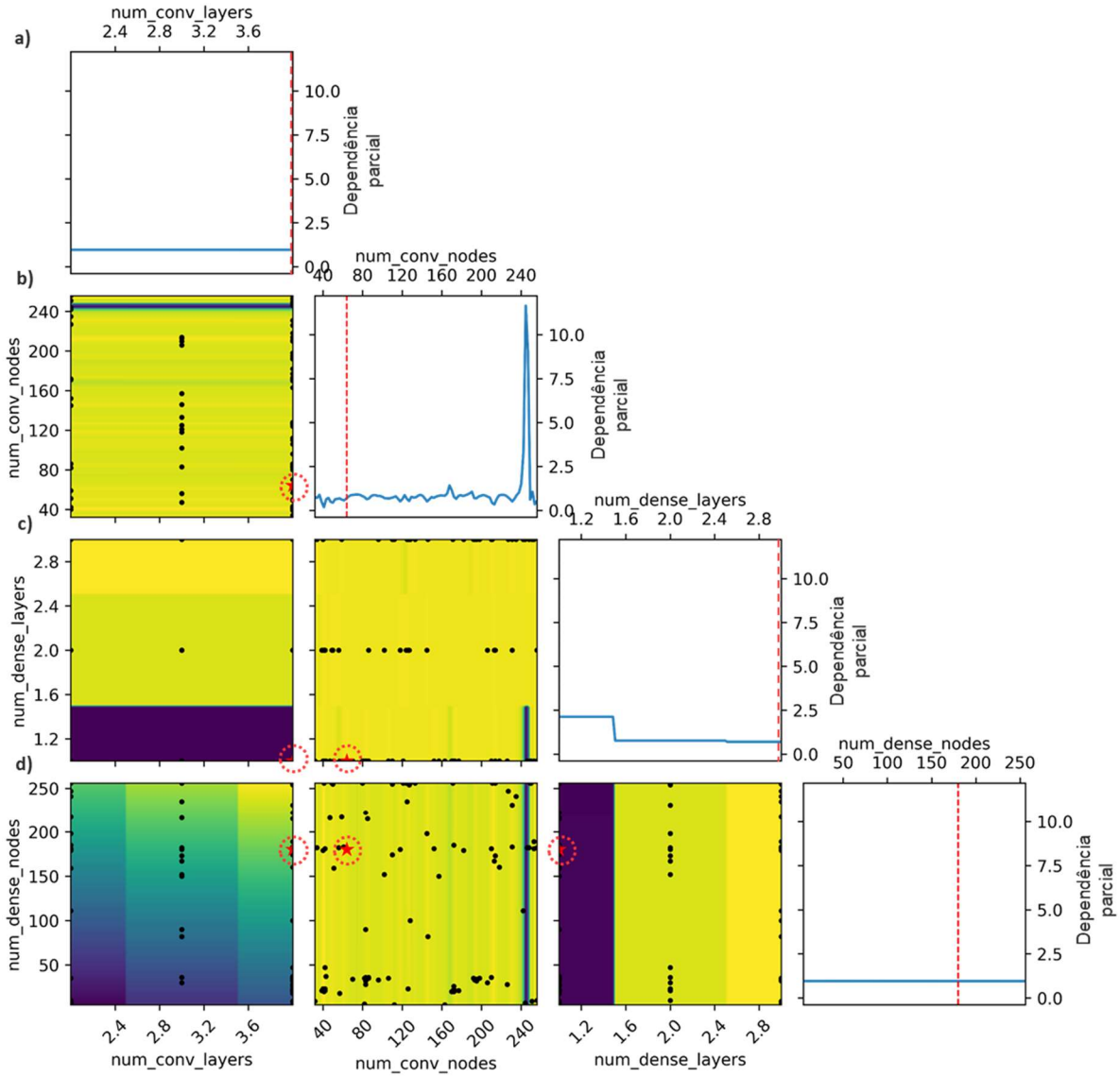
d) As quantidades de neurônios nas camadas densas, por si só, pouco influenciaram a função substituta, não havendo variações unidimensionais com a modificação deste hiperparâmetro. Diferentemente da Subseção 5.2.1, cabe observar que o conjunto (*num_conv_nodes*; *num_dense_nodes*) não apresentou um padrão na dispersão de pontos amostrados, o que não permite estabelecer uma relação concreta entre as quantidades de

neurônios utilizadas nas camadas de convolução e densas a partir da solução encontrada (64 neurônios nas camadas convolucionais; 180 neurônios na camada densa – exceto a de saída).

Cabe observar ainda que, numa comparação global dos valores das dependências parciais obtidas nos dois experimentos, o conjunto B apresentou uma faixa de variação entre 0 e 12,5, já o conjunto $A_0$ apresentou variação entre 0 e 0,175. Levando em consideração as restrições impostas ao espaço de busca, tal comparação sugere que os quatro hiperparâmetros previamente selecionados têm maior influência na função substituta quando utilizado o conjunto de dados B do que quando utilizado o conjunto de dados $A_0$.

O Apêndice D detalha histórico do processo de otimização e dos respectivos hiperparâmetros testados. Os valores estão em ordem crescente, tal como disponibilizados pelo *framework*.

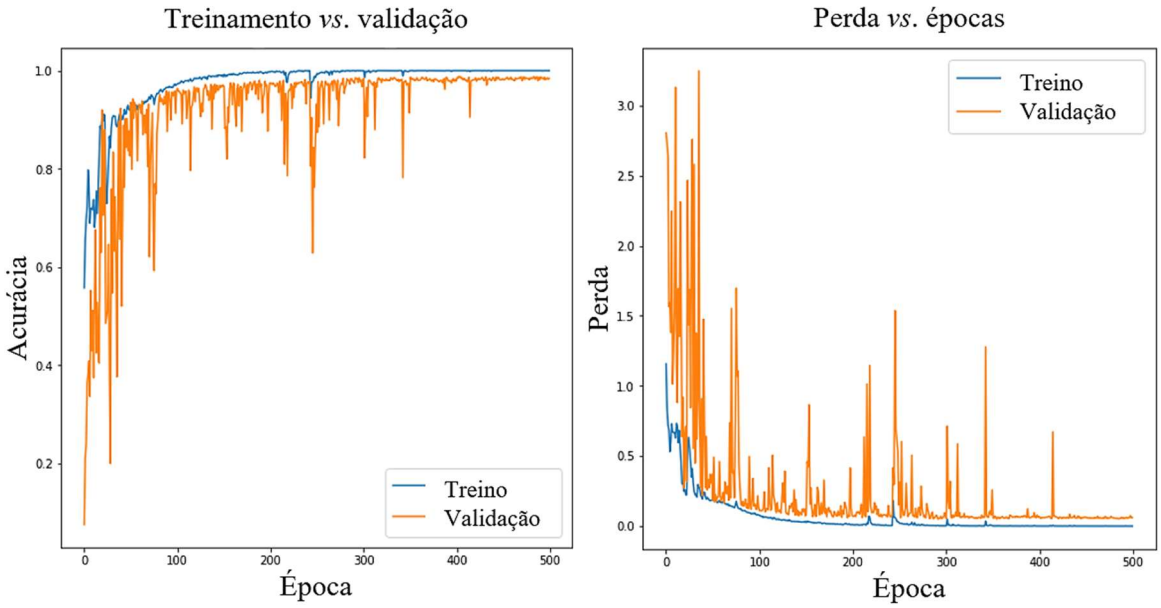
Figura 50 – Exploração do espaço de busca do otimizador bayesiano operando com o conjunto de dados B reduzido.




Fonte: (Autor, 2021)

De forma análoga ao que foi feito na Subseção 5.2.1, a arquitetura CNN com os hiperparâmetros ajustados conforme os valores ótimos descritos na Tabela 9 foi treinada, validada e testada usando o conjunto de dados B reduzido. A Figura 51 apresenta os gráficos de tais procedimentos, onde se percebe que a CNN otimizada apresentou um ótimo desempenho, exibindo uma acurácia de 98,28% nas amostras de teste, após 500 épocas de treinamento.

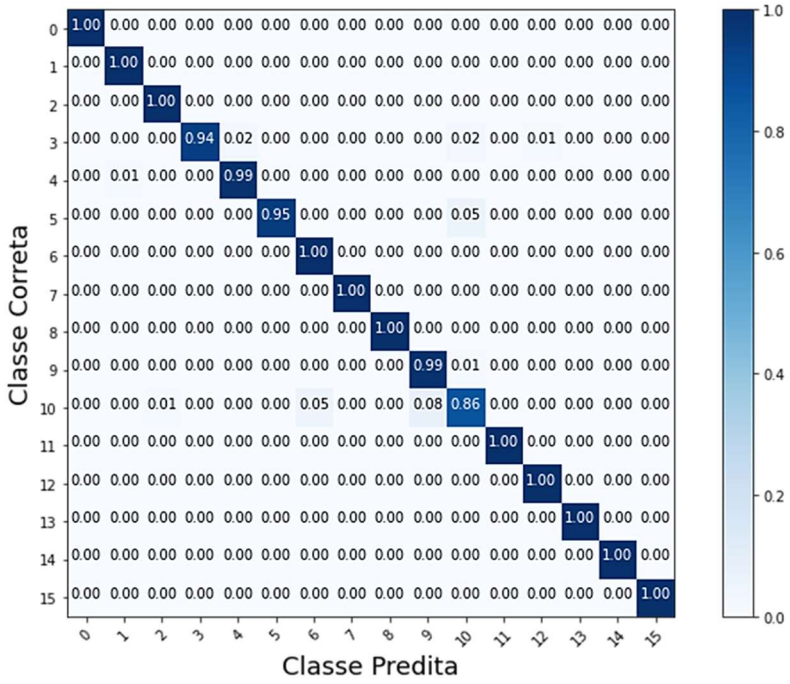
Figura 51 – Desempenho da CNN otimizada a partir do conjunto de dados B reduzido.




Fonte: (Autor, 2021)

A Figura 52 apresenta a matriz de confusão criada a partir das amostras de teste do conjunto de dados B reduzido. É possível observar que a arquitetura foi capaz de classificar corretamente a maioria das dezesseis classes, com baixa incidência de confusão identificada nas classes 3,4,5,9 e 10.

Figura 52 – Matriz de confusão do teste realizado no conjunto de dados B reduzido.




Fonte: (Autor, 2021)

A Tabela 10 apresenta algumas métricas relativas às CNNs de referência e otimizada, permitindo uma comparação direta entre ambas. A primeira é treinada com o conjunto de dados B e a segunda com o conjunto de dados B reduzido.

Tabela 10 – *Benchmark* CNN referência *vs.* otimizada.

| **Arquitetura da CNN** | **Amostras de corrente no *dataset*** | **Parâmetros da rede** | **Épocas de treinamento** | **Acurácia do teste (%)** |
|---|---|---|---|---|
| Referência | 15.993.600 | 2.962.800 | 500 | 98,49 |
| Otimizada | 10.662.400 | 284.164 | 500 | 98,28 |
| **redução (%)** | 33,33 | 90,41 | 0,00 | 0,21 |

Fonte: (Autor, 2021)

Com relação aos parâmetros totais da rede, o otimizador pôde encontrar uma arquitetura bastante enxuta. Como já informado, em (FIRMES, 2020), a arquitetura da rede #3 continha 2.962.800 parâmetros totais, já o 2º estágio de otimização foi capaz de entregar uma arquitetura com apenas 284.164 parâmetros totais, o que representa uma redução em torno de 90,41% (Figura 53).

Figura 53 – Arquitetura da rede CNN otimizada.

| **Nome da camada e Tipo** | **Formato da Saída** | **# Parâmetros** |
|---|---|---|
| conv1d (*Conv1D*) | (1666, 64) | 320 |
| max_pooling1d (*MaxPooling1D*) | (555, 64) | 0 |
| batch_normalization (*BatchNormalization*) | (555, 64) | 256 |
| conv1d_1 (*Conv1D*) | (555, 64) | 16448 |
| max_pooling1d_1 (*MaxPooling1D*) | (185, 64) | 0 |
| batch_normalization_1 (*BatchNormalization*) | (185, 64) | 256 |
| conv1d_2 (*Conv1D*) | (185, 64) | 16448 |
| max_pooling1d_2 (*MaxPooling1D*) | (61, 64) | 0 |
| batch_normalization_2 (*BatchNormalization*) | (61, 64) | 256 |
| conv1d_3 (*Conv1D*) | (61, 64) | 16448 |
| max_pooling1d_3 (*MaxPooling1D*) | (20, 64) | 0 |
| batch_normalization_3 (*BatchNormalization*) | (20, 64) | 256 |
| flatten (*Flatten*) | (1280) | 0 |
| dense (*Dense*) | (180) | 230580 |
| dense_1 (*Dense*) | (16) | 2896 |

**Total de parâmetros:** 284.164
**Parâmetros treináveis:** 283.652
**Parâmetros não treináveis:** 512

Fonte: (Autor, 2021)

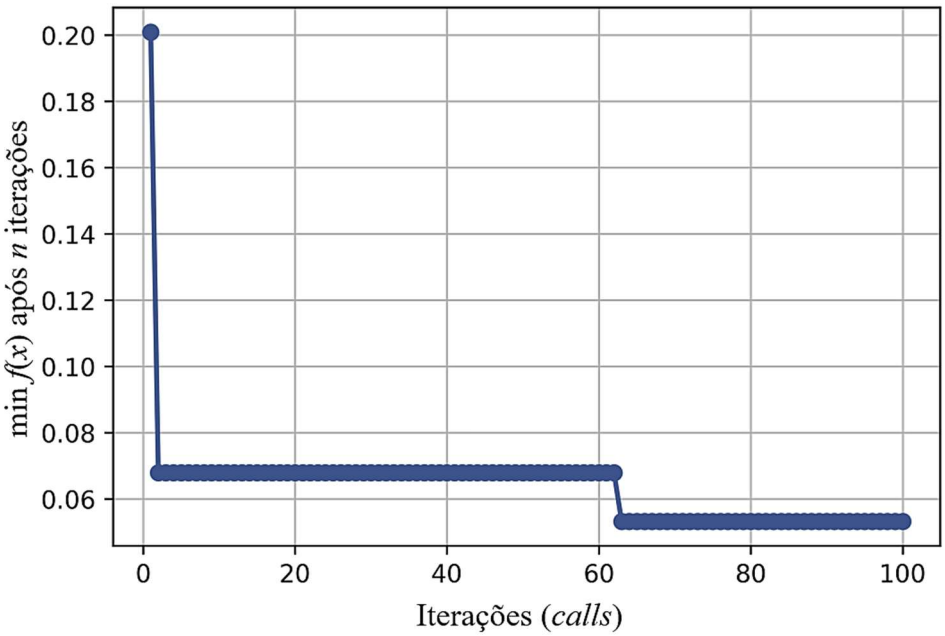
## 5.3 Relação casos *vs.* parâmetros da rede

Conforme apresentado anteriormente, foi possível alcançar uma redução do número total de parâmetros treináveis em 56,02% (Tabela 8) para a rede treinada com a base de dados $A_0$ e de 90,41% (Tabela 10) para a rede treinada com a base B.

Visando investigar o motivo de tal diferença, isto é, se o uso de uma quantidade reduzida de casos no conjunto de dados $A_0$ (em relação ao conjunto B) teve influência sobre isso, foi conduzido mais um experimento para o 2º estágio, desta vez usando 400 casos no conjunto $A_0$, equiparando à quantidade de casos identificada no 1º estágio do conjunto B.

A Figura 54 ilustra a curva de *fitness* fornecida pelo otimizador. O experimento durou aproximadamente 7h00m, representando um tempo 254,5% superior se comparado ao experimento descrito na Subseção 5.1.1, o que era esperado devido à uma quantidade 666,6% superior de dados disponíveis para o treinamento e validação.

Figura 54 – Curva de *fitness* durante otimização de hiperparâmetros de CNN quando usando o conjunto de dados $A_0$ reduzido a 400 casos.




Fonte: (Autor, 2021)

Como é possível observar na figura, o otimizador convergiu para o valor mais próximo de zero somente após a 60ª iteração. A Tabela 11 mostra a solução encontrada pelo otimizador.

Tabela 11 – Hiperparâmetros de CNN otimizados usando o conjunto de dados $A_0$ reduzido a 400 casos.

| Hiperparâmetro | Ajuste |
|---|---|
| Nº de camadas convolucionais | 4 |
| Nº de neurônios nas camadas convolucionais | 256 |
| Nº de camadas densas (exceto a de saída) | 3 |
| Nº de neurônios nas camadas densas (exceto na de saída) | 184 |

Fonte: (Autor, 2021)

Comparando os dois experimentos utilizando o conjunto de dados $A_0$ (400 casos contra 60 casos), foi evidenciada uma quantidade 38,57% maior de parâmetros totais utilizando 400 casos se comparado ao experimento com 60 casos.

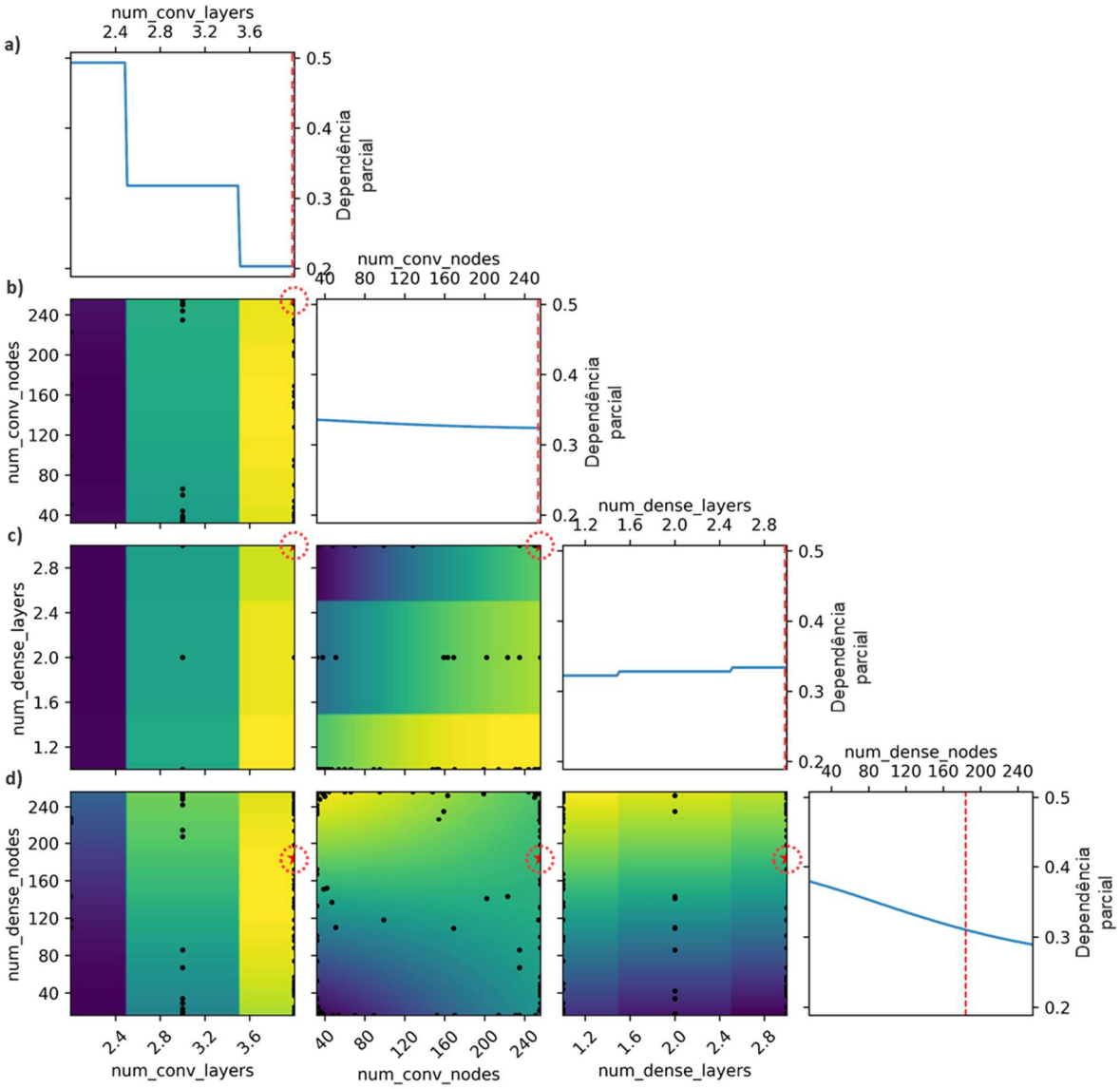
Com relação aos pontos amostrados e às dependências parciais apresentadas nos gráficos da Figura 55, foram constatados comportamentos diferentes do experimento com 60 casos. Sugere-se que tal fenômeno tenha ocorrido em detrimento do tamanho do conjunto de dados utilizado e da natureza probabilística do otimizador.

Com relação à avaliação das dependências parciais unidimensionais em conjunto com a função substituta, evidencia-se um decréscimo de influência à medida em que se aumenta o número de camadas convolucionais (Figura 55a); baixa variação na influência do número de neurônios nas camadas convolucionais e, também, do número de camadas densas (Figura 55b, c); e um decréscimo de influência, praticamente linear, à medida em que se aumenta os neurônios nas camadas densas (Figura 55d).

Quanto às dependências parciais bidimensionais, especificamente o conjunto (*num_conv_nodes*; *num_dense_nodes*) (Figura 55d), percebe-se uma distribuição mais esparsa dos pontos amostrados, o que diverge do experimento apresentado na Subseção 5.1.1, não sendo possível constatar nenhum padrão de concentração de pontos amostrados nas regiões próximas à melhor solução.

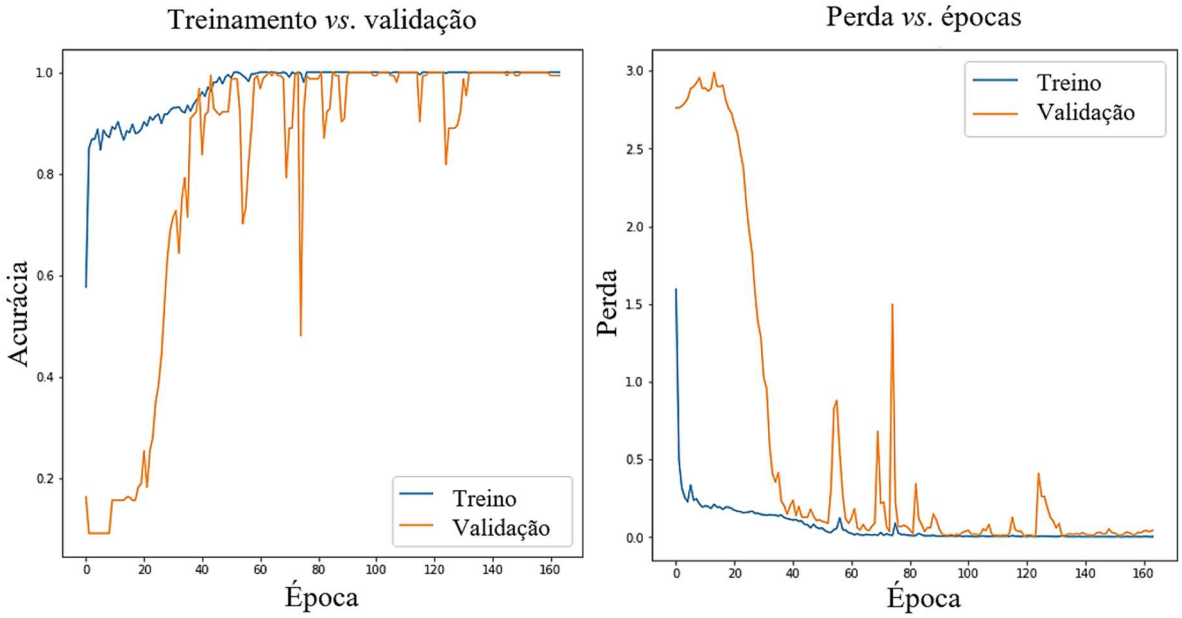
As Figuras 56 e 57 apresentam o desempenho da arquitetura CNN ao utilizar 400 casos no conjunto de dados $A_0$. Com relação às etapas de treinamento e validação (Figura 56), observa-se que a rede CNN apresentou um desempenho similar ao apresentado com a utilização de 60 casos. Com relação à matriz de confusão (Figura 57), observa-se que a rede classificou corretamente todas as dezesseis classes, apresentando desempenho igualmente similar ao que foi discutido na Subseção 5.1.1.

Figura 55 – Exploração do espaço de busca do otimizador bayesiano operando com o conjunto de dados $A_0$ reduzido a 400 casos.



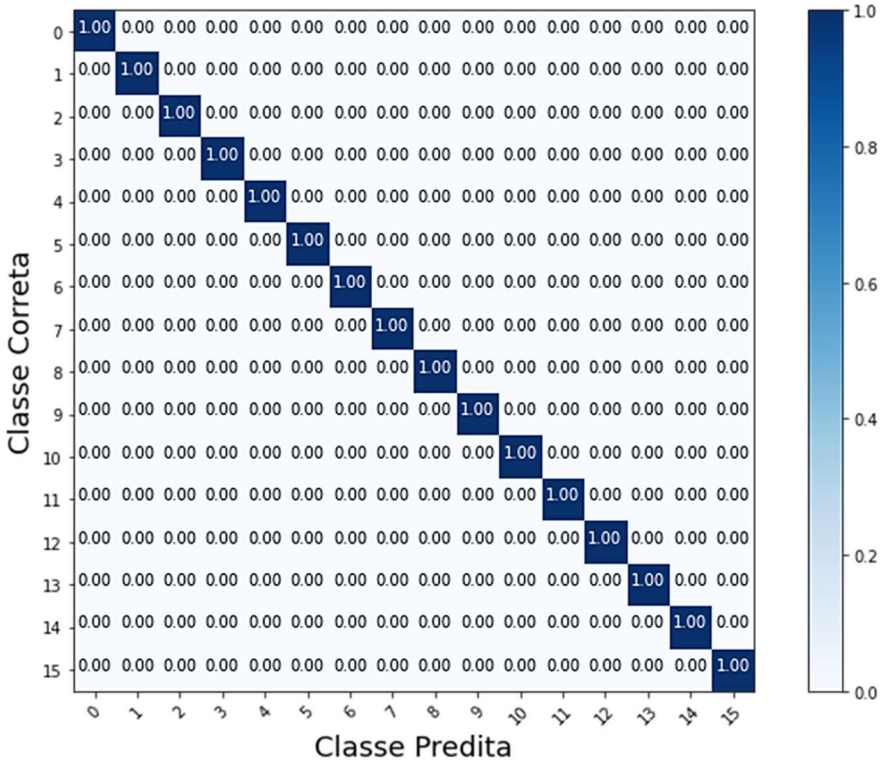

Fonte: (Autor, 2021)

Figura 56 – Desempenho da CNN otimizada a partir do conjunto de dados $A_0$ reduzido a 400 casos.




Fonte: (Autor, 2021)

Figura 57 – Matriz de confusão do teste realizado no conjunto de dados B reduzido a 400 casos.




Fonte: (Autor, 2021)

Em linhas gerais, com 60 casos, foi identificada uma redução de 56,02% dos parâmetros totais em relação à arquitetura original de (FIRMES, 2020), já com 400 casos, constatou-se redução de 60,96% dos parâmetros totais, conforme mostra a Figura 58.

Figura 58 – Arquitetura da rede CNN otimizada utilizando 400 casos.

| Nome da camada e Tipo | Formato da Saída | # Parâmetros |
|---|---|---|
| conv1d (*Conv1D*) | (1666, 256) | 1280 |
| max_pooling1d (*MaxPooling1D*) | (555, 256) | 0 |
| batch_normalization (*BatchNormalization*) | (555, 256) | 1024 |
| conv1d_1 (*Conv1D*) | (555, 256) | 262400 |
| max_pooling1d_1 (*MaxPooling1D*) | (185, 256) | 0 |
| batch_normalization_1 (*BatchNormalization*) | (185, 256) | 1024 |
| conv1d_2 (*Conv1D*) | (185, 256) | 262400 |
| max_pooling1d_2 (*MaxPooling1D*) | (61, 256) | 0 |
| batch_normalization_2 (*BatchNormalization*) | (61, 256) | 1024 |
| conv1d_3 (*Conv1D*) | (61, 256) | 262400 |
| max_pooling1d_3 (*MaxPooling1D*) | (20, 256) | 0 |
| batch_normalization_3 (*BatchNormalization*) | (20, 256) | 1024 |
| flatten (*Flatten*) | (5120) | 0 |
| dense (*Dense*) | (184) | 942264 |
| dense_1 (*Dense*) | (184) | 34040 |
| dense_2 (*Dense*) | (184) | 34040 |
| dense_3 (*Dense*) | (16) | 2960 |

**Total de parâmetros:** 1.805.880
**Parâmetros treináveis:** 1.803.832
**Parâmetros não treináveis:** 2048

Fonte: (Autor, 2021)

Isso significa que ao utilizar o conjunto de dados $A_0$, a taxa de redução dos parâmetros aumentou com a redução de casos, mas ficou longe dos 90% reduzidos no experimento de 400 casos com o conjunto de dados B.

## 5.4 Avaliação das CNNs Treinadas com Diferentes Amostras de Teste

Após constatação do alto desempenho de classificação das cargas elétricas, mesmo após a redução drástica do número de casos disponíveis para separação em conjuntos de treinamento, validação e teste (como observado no conjunto de dados $A_0$), suspeitou-se que tais reduções durante o 1º estágio possam limitar a capacidade de generalização da rede CNN ao pô-la em prova perante à amostra de teste referente ao tamanho completo da base de dados original (20% de 600 casos). Levou-se em consideração que, ao reduzir a base dados original, aplicou-se uma

redução relativa à quantidade das amostras de treinamento, validação e teste, implicando em perda de informações importantes para o processo de treinamento da CNN.

Portanto, para averiguar o desempenho das redes treinadas com redução ótima dos casos, reservou-se a mesma quantidade de casos para teste utilizada em (FIRMES, 2020), a qual foi aplicada durante três novos experimentos, conforme descrito:

a) Treinamento e validação da CNN com 80% de 60 casos do conjunto de dados $A_0$ (conforme executado na Subseção 5.1.1) e teste com uma nova amostra contendo 20% dos 600 casos totais de $A_0$;

b) Treinamento e validação da CNN com 80% de 400 casos do conjunto de dados $A_0$ (conforme executado na Subseção 5.3) e teste com uma nova amostra contendo 20% dos 600 casos totais de $A_0$;

c) Treinamento e validação da CNN com 80% de 400 casos do conjunto de dados B (conforme executado na Subseção 5.1.2) e teste com uma nova amostra contendo 20% dos 600 casos totais de B.

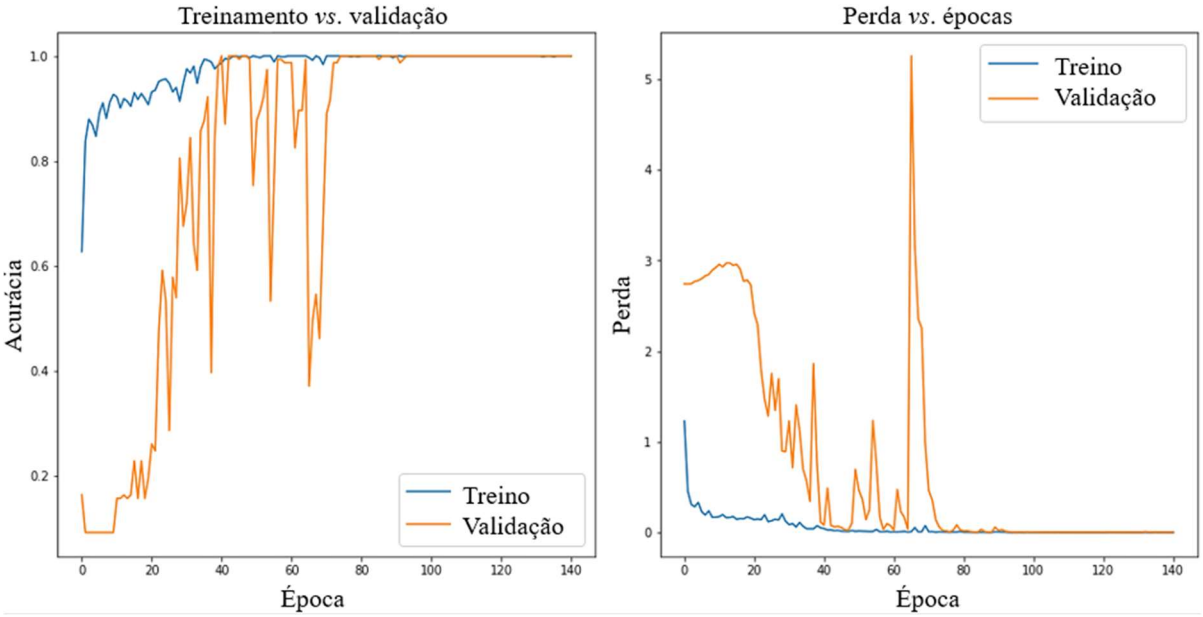
Os resultados são apresentados nas Figuras 59, 60, 61, 62, 63 e 64, as quais correspondem, respectivamente, à obtenção da acurácia e perda nas amostras de treinamento/validação, assim como as respectivas matrizes de confusão para cada um dos três experimentos adicionais. As seguintes observações podem ser feitas a partir das figuras obtidas:

a) Ao treinar o modelo com o conjunto de dados $A_0$ contendo 80% de 60 casos, o comportamento das acurácias de treinamento e validação evidenciaram-se suficientemente altas (Figura 59), tal como observado na Subseção 5.1.1. Entretanto, ao submeter o mesmo modelo ao conjunto de dados de teste referente à massa de dados total (20% de 600 casos), foi obtida uma acurácia de classificação de 27,55%, o que representa uma redução de 72,45% na métrica anteriormente obtida (Figura 60);

b) Ao treinar o modelo com o conjunto de dados $A_0$ contendo 80% de 400 casos, o comportamento das acurácias de treinamento e validação evidenciaram-se suficientemente altas (Figura 61), tal como observado na Subseção 5.3. Entretanto, ao submeter o mesmo modelo ao conjunto de dados de teste referente à massa de dados total (20% de 600 casos), foi obtida uma acurácia de classificação de 74,94%, o que representa uma redução de 25,06% na métrica anteriormente obtida (Figura 62);

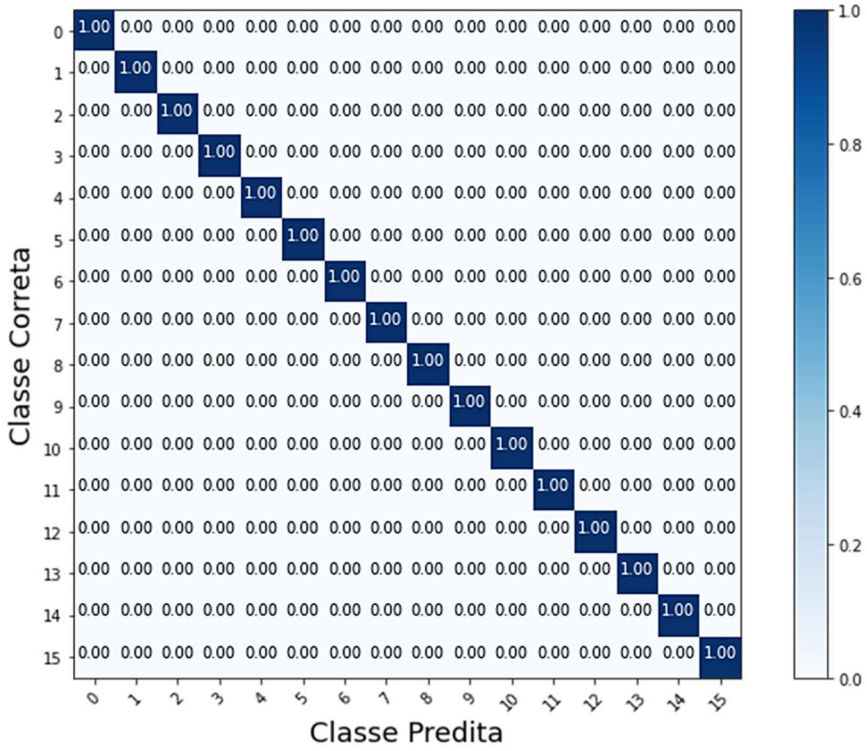
c) Ao treinar o modelo com o conjunto de dados B contendo 80% de 400 casos, o comportamento das acurácias de treinamento e validação evidenciaram-se suficientemente altas (Figura 63), tal como observado na Subseção 5.1.2. Entretanto, ao submeter o mesmo modelo ao conjunto de dados de teste referente à massa de dados total (20% de 600 casos), foi obtida uma acurácia de classificação de 78,69%, o que representa uma redução de redução de 19,59% na métrica anteriormente obtida (Figura 64).

Figura 59 – Desempenho da CNN otimizada a partir do conjunto de dados $A_0$ reduzido a 60 casos.



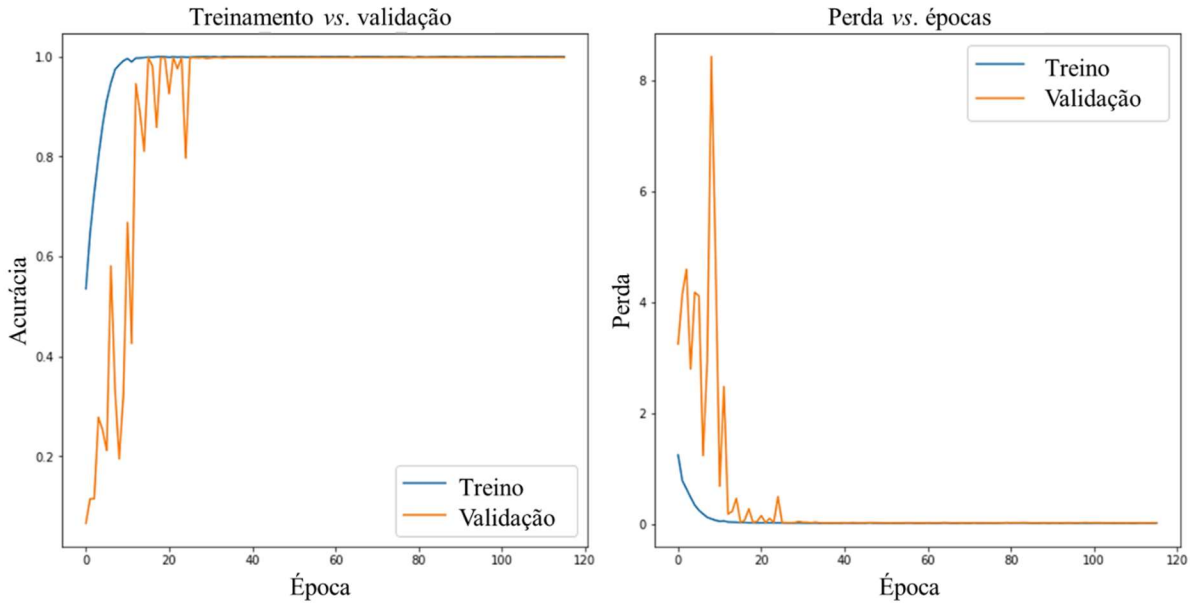

Fonte: (Autor, 2021)

Figura 60 – Matriz de confusão obtida pela CNN treinada com 60 casos e testada com o conjunto de testes $A_0$ original (20% de 600 casos).



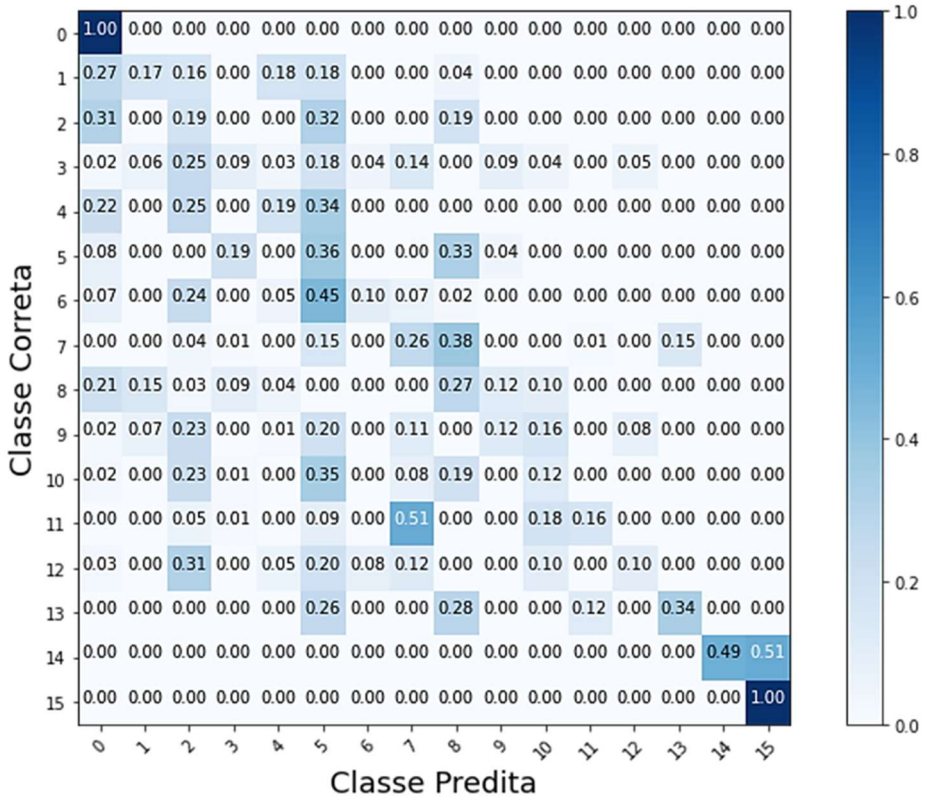

Fonte: (Autor, 2021)

Figura 61 – Desempenho da CNN otimizada a partir do conjunto de dados $A_0$ reduzido a 400 casos.



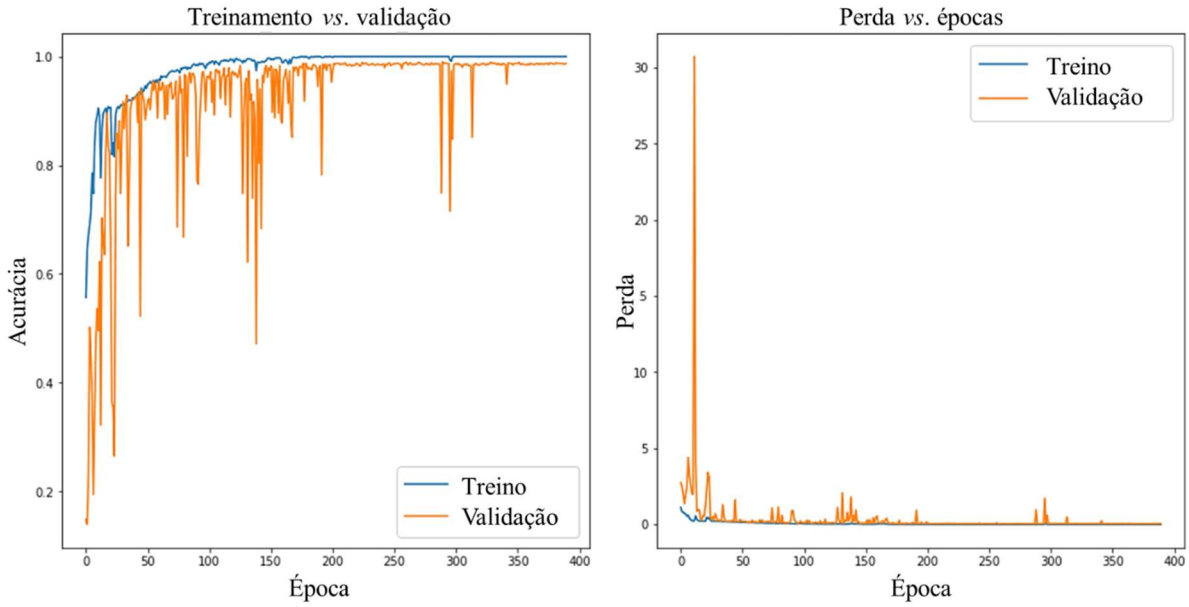

Fonte: (Autor, 2021)

Figura 62 – Matriz de confusão obtida pela CNN treinada com 400 casos e testada com o conjunto de testes $A_0$ original (20% de 600 casos).



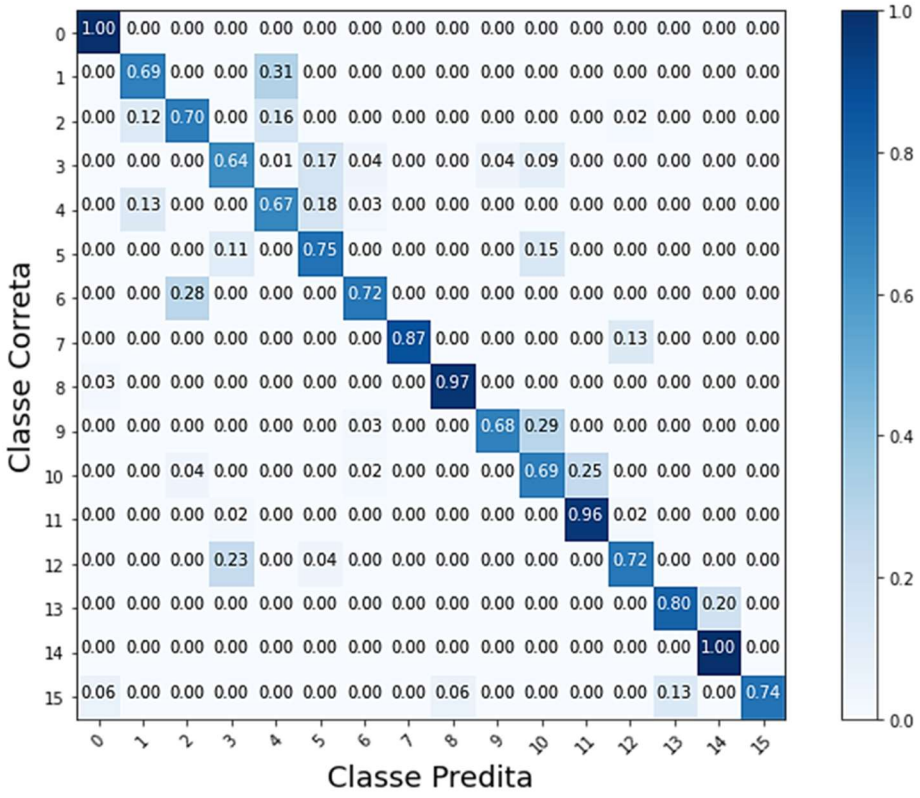

Fonte: (Autor, 2021)

Figura 63 – Desempenho da CNN otimizada a partir do conjunto de dados B reduzido a 400 casos.




Fonte: (Autor, 2021)

Figura 64 – Matriz de confusão obtida pela CNN treinada com 400 casos e testada com o conjunto de testes B original (20% de 600 casos).




Fonte: (Autor, 2021)

Finalmente, a Tabela 12 apresenta o comparativo final dos experimentos executados. O que se pode concluir é que a quantidade de casos de teste utilizada na análise de desempenho da CNN afeta consideravelmente o resultado. Reduzir o conjunto de dados e depois subdividi-lo em conjuntos de treinamento, validação e teste, de acordo com o Princípio de Pareto, não é suficiente para que os parâmetros do modelo sejam suficientemente treinados para realizar a classificação das cargas em um contexto mais realista, isto é, com uma quantidade maior de novos casos.

As matrizes de confusão, em especial a da Figura 62, deixam claro que a grande redução implementada no conjunto de dados $A_0$, assim como no conjunto B, culminou no descarte de dados importantes para o teste da rede numa grande quantidade de dados ainda não vistos. Percebe-se que a CNN não obteve problemas para classificar as classes 0 e 15, que representam todas as cargas desligadas e todas as cargas ligadas, respectivamente. Contudo, houve dificuldade para proceder com a classificação de cargas combinadas, o que pode ser justificado pelo descarte supracitado, implementado durante o 1º estágio de otimização.

Tabela 12 – Comparativo dos experimentos após 2º estágio.

| **Base de dados** | **Arquitetura da CNN** | **Quantidade de casos** | **Tamanho da amostra de teste** | **Amostras de corrente no *dataset*** | **Parâmetros da rede** | **Acurácia (%)** |
|---|---|---|---|---|---|---|
| 100% de $A_0$ | original | 600 | 20% de 600 casos | 15.993.600 | 2.962.800 | 99,90 |
| 66,6% de $A_0$ | ajustada | 400 | 20% de 400 casos | 10.662.400 | 1.805.880 | 100,00 |
| 66,6% de $A_0$ | ajustada | 400 | 20% de 600 casos | 10.662.400 | 1.805.880 | 74,94 |
| 10% de $A_0$ | ajustada | 60 | 20% de 60 casos | 1.599.360 | 1.303.144 | 100,00 |
| 10% de $A_0$ | ajustada | 60 | 20% de 600 casos | 1.599.360 | 1.303.144 | 27,55 |
| 100% de B | original | 600 | 20% de 600 casos | 15.993.600 | 2.962.800 | 98,49 |
| 66,6% de B | ajustada | 400 | 20% de 400 casos | 10.662.400 | 284.164 | 98,28 |
| 66,6% de B | ajustada | 400 | 20% de 600 casos | 10.662.400 | 284.164 | 78,69 |

Fonte: (Autor, 2021)

# 6 Considerações Finais

Este trabalho descreveu, em detalhes, o processo de ajuste automatizado de redes neurais convolucionais dedicadas à identificação de cargas elétricas altamente similares representadas em dois conjuntos de dados distintos, o $A_0$ e o B.

Tal ajuste foi implementado através da utilização de dois otimizadores, o que exigiu o planejamento dos experimentos em dois estágios, sendo:

1. Encontrar um tamanho ótimo para os conjuntos de dados utilizados em função da quantidade de casos disponíveis para cada uma das 16 classes, através da implementação de uma adaptação do otimizador *Hill Climbing*;
2. Implementar o ajuste automatizado de um conjunto de hiperparâmetros previamente estabelecido, através da implementação de um otimizador bayesiano.

Para o caso do conjunto de dados $A_0$, foi constatada a possibilidade de redução de 90% no tamanho original utilizado por (FIRMES, 2020), mantendo, ainda, um desempenho similar na sua acurácia de classificação. Já no caso do conjunto de dados B, foi possível estabelecer uma redução de 33,4% no tamanho original utilizado no trabalho de referência.

Este primeiro estágio foi essencial, pois permitiu identificar uma condição em que o custo computacional fosse o menor possível durante o ajuste de hiperparâmetros no estágio seguinte.

Ao analisar os resultados, constatou-se que a acurácia de classificação no conjunto de dados $A_0$ permaneceu suficientemente alta mesmo com a redução acentuada de casos. Isso, por outro lado, não ocorreu com o banco B. Sabendo que o banco $A_0$ representa cargas de comportamento determinístico enquanto que o banco B representa cargas de comportamento não determinístico, supõe-se que o aumento na complexidade comportamental limita o nível de redução do tamanho do banco necessário para um treinamento efetivo da arquitetura CNN. Entretanto, por fugir dos objetivos originalmente traçados para este trabalho, é aqui proposto que tal efeito seja investigado em trabalhos futuros.

Baseado nas observações das duas bases de dados, conclui-se que, apesar da quantidade de casos ter tido um efeito na quantidade de parâmetros da rede, não é possível afirmar que o tamanho do *dataset* seja um fator preponderante quanto à redução do número de parâmetros treináveis.

Quanto à aplicação das heurísticas para abordagem do problema, foi possível identificar que o otimizador *Hill Climbing* convencional (1º estágio), apesar de proporcionar soluções satisfatórias, ficou exposto à influência de soluções mínimas locais para o conjunto de dados $A_0$, evidenciado a partir do treinamento instável pela CNN (Figura 36). Tal comportamento culminou em diversas paradas inesperadas do processo e exigiu o ajuste manual do hiperparâmetro denominado taxa de aprendizagem.

Os dados obtidos pelo otimizador bayesiano não foram suficientes para investigar, em detalhes, o modo como é fornecida uma solução de hiperajuste que acarrete em *x* parâmetros totais na arquitetura da rede CNN ao invés de uma solução de *y* parâmetros totais. Levanta-se, porém, a hipótese de que como o custo otimizado, advindo de uma RNN estocástica, variou na ordem de centésimos/milésimos (quando comparados os experimentos), a probabilidade de o otimizador oferecer uma solução com soluções de hiperajustes bastante distintos e que ainda assim minimizem o custo é relativamente alta, impactando a quantidade final de parâmetros totais da arquitetura CNN.

Além disso, outro ponto que há de se avaliar está contido nas características intrínsecas das duas bases de dados. Como informado no Capítulo 4, o conjunto de dados $A_0$ é composto por cargas invariantes no tempo, já o conjunto B, por cargas de variação não determinística. Os padrões contidos na estrutura dos *datasets*, assim como a distribuição dos possíveis *outliers*, são fatores igualmente importantes de se avaliar. Ainda que técnicas de pré-processamento existentes atenuem possíveis efeitos relacionados aos algoritmos de aprendizado de máquina, tais características intrínsecas podem se propagar ao restante do sistema, causando impactos durante o treinamento e validação dos dados.

Tais observações sugerem que a questão envolvendo a relação entre o decréscimo de casos e o efeito sobre a quantidade de parâmetros segue em aberto, o que está evidenciado nos comportamentos tão distintos entre os conjuntos de dados $A_0$ e B.

Ainda sobre o segundo estágio, baseado nos trabalhos discutidos na revisão de literatura, determinou-se ajustar os hiperparâmetros denominados: número de camadas convolucionais, número de neurônios nas camadas convolucionais, número de camadas densas e número de neurônios nas camadas densas. O otimizador desempenhou os experimentos conforme esperado, encontrando um ajuste ótimo em função do custo que lhe foi atribuído.

Ficou nítido que o tempo gasto pelo otimizador para convergir à uma solução de hiperparâmetros ao utilizar o conjunto de dados B foi superior ao tempo gasto utilizando o

conjunto de dados $A_0$. Além disso, é importante ressaltar que a otimização de apenas 4 hiperparâmetros da rede CNN resultou em uma grande redução na quantidade de parâmetros totais da rede em questão quando comparada com a rede CNN de referência proposta em (FIRMES, 2020), a qual possui 2.962.800 parâmetros totais, contra 284.164 parâmetros totais encontrados, o que representa uma redução em torno de 90%. Como já comentado na Subseção 5.2.1, tal redução é de extrema importância, pois provoca um impacto positivo ao reduzir a complexidade do hardware necessário para tratar um dado problema.

Adicionalmente, ao implementar os experimentos da Seção 5.4, constatou-se que o tamanho da amostra de testes utilizada para medição do desempenho de classificação da CNN possui influência sobre a confiabilidade dos resultados. Conforme discutido, reduzir a massa total de dados (em função de casos) e depois subdividi-la em conjuntos de treinamento, validação e teste não significa que os parâmetros do modelo estarão suficientemente treinados para realizar a classificação das cargas em um contexto com uma quantidade maior de dados não vistos. Tais constatações exigem que este fenômeno seja levado em consideração nos futuros experimentos acerca do problema de classificação de cargas elétricas similares.

Como contribuições, é possível elencar:

a) O desenvolvimento de um relatório técnico (Apêndice A), contendo uma taxonomia das principais metaheurísticas utilizadas em otimização para *deep learning*, em que se espera ser útil para futuras consultas;

b) A exploração do comportamento da taxa de aprendizado da CNN configurada junto ao conjunto de dados $A_0$, visando estabilizar o treinamento e aumentar a robustez dos resultados;

c) A aplicação da técnica de regularização denominada *Early Stopping* (parada antecipada), diminuindo o tempo total do experimento;

d) A implementação da redução automatizada dos conjuntos de dados, com o objetivo de analisar o comportamento das CNNs sob tais circunstâncias e reduzir o esforço computacional durante o 2º estágio de otimização;

e) A constatação de um número de casos ótimo para os conjuntos de dados $A_0$ e B, usados para resolver o problema de identificação de cargas elétricas similares em *smart grid*;

f) A implementação de um ajuste automatizado para o conjunto de hiperparâmetros da CNN de referência, representando a principal evolução do trabalho realizado em (FIRMES, 2020);

g) A redução da quantidade total de parâmetros da rede ao otimizar quatro dos principais hiperparâmetros de uma arquitetura CNN, resultando em um menor esforço humano e computacional sem abrir mão do desempenho de classificação;

h) A constatação de que o tamanho do conjunto de teste deve ser tratado com cautela devido sua importância ao evidenciar o desempenho da rede em um contexto real.

Sobre os empecilhos e desafios enfrentados durante os experimentos, é importante salientar que a plataforma *Google Colab* possui mecanismos de restrição quanto ao tempo de execução ininterrupta (diferente do tempo de execução máxima por sessão), e quanto aos recursos virtuais de memória RAM e de GPU. Isso exigiu a subscrição ao *Colab Pro*, o que atribuiu gastos com mensalidades referentes ao serviço, além do delineamento de uma estratégia computacional que possibilitasse a execução dos códigos, tal como apresentado no Capítulo 4.

A execução dos experimentos através de um navegador de internet culminou em um consumo altíssimo de memória RAM, o que causou eventuais *crashs*, mesmo em uma máquina considerada “de ponta”. Recomenda-se, no mínimo, a utilização de um processador Intel Core I7 (ou equivalente) e 16 Gb de memória RAM para um experimento desta envergadura. Foram testados os navegadores Google Chrome, Mozilla Firefox e Microsoft Edge, sendo que este último aparentou desempenhar um melhor gerenciamento de recursos da máquina.

Outro problema identificado foi a instabilidade do fornecimento de internet o que, de certa forma, está fora do controle do desenvolvedor. Mesmo utilizando conexão via cabo *Ethernet*, houveram *crashs* no processamento do algoritmo, levando à incontáveis situações de retrabalho.

Por tais motivos, fica a recomendação de que seja utilizada uma máquina com memória RAM e GPUs dedicadas para experimentos com *deep learning*, não dependendo exclusivamente do compartilhamento de recursos computacionais via *cloud*.

Finalmente, baseando-se na discussão levantada a partir dos experimentos realizados, como trabalhos futuros recomenda-se:

a) Investigar potenciais diferenças nas características intrínsecas dos dados correspondentes às bases de dados $A_0$ e B;

b) Investigar a influência que os métodos de pré-processamento (escalonamento) existentes podem exercer nas bases de dados utilizadas;

c) Investigar o modo como ocorre a degradação da acurácia em função do decréscimo de casos nos conjuntos de dados;

d) Investigar a possibilidade de estabelecimento de uma relação entre o decréscimo de casos e a quantidade de parâmetros totais necessários para o treinamento das redes CNN;

e) Implementar a rede CNN em um contexto real e investigar seu comportamento quando submetida a uma quantidade maior de novos casos para classificação.

# Referências Bibliográficas

ABDECHIRI, M.; MEYBODI, M. R.; BAHRAMI, H. Gases Brownian Motion Optimization: An Algorithm for Optimization (GBMO). **Applied Soft Computing**, v. 13, n. 5, p. 2932–2946, 1 maio 2013.

ABDEL-BASSET, M.; ABDEL-FATAH, L.; SANGAIAH, A. K. Metaheuristic Algorithms: A Comprehensive Review. In: SANGAIAH, A. K.; SHENG, M.; ZHANG, Z. (Org.). . **Comput. Intell. Multimed. Big Data Cloud Eng. Appl.** Intelligent Data-Centric Systems. [S.l.]: Academic Press, 2018. p. 185–231. Disponível em: <http://www.sciencedirect.com/science/article/pii/B9780128133149000104>. Acesso em: 20 mar. 2020.

ABDELHAFIEZ, E. A.; ALTURKI, F. A. A Shaking Optimization Algorithm for Solving Job Shop Scheduling Problem. **Industrial Engineering and Management Systems**, v. 10, n. 1, p. 7–14, 2011.

ABDEL-RAOUF, O.; METWALLY, M. A. A Survey of Harmony Search Algorithm. **International Journal of Computer Applications**, v. 70, n. 28, p. 17–26, 31 maio 2013.

ABDULHAMID, S. M. et al. A Survey of League Championship Algorithm: Prospects and Challenges. **Indian Journal of Science and Technology**, arXiv: 1603.09728, v. 8, n. S3, p. 101, 1 fev. 2015.

AGARWAL, S.; SINGH, A. P.; ANAND, N. Evaluation performance study of Firefly algorithm, particle swarm optimization and artificial bee colony algorithm for non-linear mathematical optimization functions. In: 2013 FOURTH INTERNATIONAL CONFERENCE ON COMPUTING, COMMUNICATIONS AND NETWORKING TECHNOLOGIES (ICCCNT), jul. 2013, [S.l: s.n.], jul. 2013. p. 1–8.

AHANGARAN, M.; RAMEZANI, P. Harmony Search Algorithm: Strengths and Weaknesses. **Journal of Computer Engineering & Information Technology**, v. 2, 2013. Disponível em: <https://www.scitechnol.com/harmony-search-algorithm-strengths-and-weaknesses-yuSN.php?article_id=557>. Acesso em: 26 mar. 2020.

AHMED, M.; VIRIRI, S. Deep Learning Using Bayesian Optimization for Facial Age Estimation. Lecture Notes in Computer Science, 2019, Cham. *Anais*... Cham: Springer International Publishing, 2019. p. 243–254.

ALAHAKOON, D.; YU, X. Smart Electricity Meter Data Intelligence for Future Energy Systems: A Survey. **IEEE Transactions on Industrial Informatics**, v. 12, n. 1, p. 425–436, fev. 2016.

ALAM, M. R.; REAZ, M. B. I.; ALI, M. A. M. A Review of Smart Homes—Past, Present, and Future. **IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)**, v. 42, n. 6, p. 1190–1203, nov. 2012.

ALATAS, B. Chaotic Harmony Search Algorithms. **Applied Mathematics and Computation**, v. 216, n. 9, p. 2687–2699, 1 jul. 2010.

ALBA, E. **Parallel Metaheuristics**. [S.l.]: John Wiley & Sons, Inc., 2005. Disponível em: <https://www.wiley.com/en-us/Parallel+Metaheuristics%3A+A+New+Class+of+Algorithms-p-9780471678069>. Acesso em: 15 abr. 2020. (Wiley Series on Parallel and Distributed Computing).

ALBA, E.; LUQUE, G.; NESMACHNOW, S. Parallel Metaheuristics: Recent Advances and New Trends. **International Transactions in Operational Research**, v. 20, n. 1, p. 1–48, 2013.

ALBELWI, S.; MAHMOOD, A. Analysis of instance selection algorithms on large datasets with Deep Convolutional Neural Networks. In: 2016 IEEE LONG ISLAND SYSTEMS, APPLICATIONS AND TECHNOLOGY CONFERENCE (LISAT), abr. 2016, [S.l: s.n.], abr. 2016. p. 1–5.

AL-BETAR, M. A. β-Hill Climbing: An Exploratory Local Search. **Neural Computing and Applications**, v. 28, n. 1, p. 153–168, 1 dez. 2017.

ALWESHAH, M. et al. β-Hill Climbing Algorithm with Probabilistic Neural Network for Classification Problems. **Journal of Ambient Intelligence and Humanized Computing**, v. 11, n. 8, p. 3405–3416, 1 ago. 2020.

ANDONIE, R.; FLOREA, A.-C. Weighted Random Search for CNN Hyperparameter Optimization. **INTERNATIONAL JOURNAL OF COMPUTERS COMMUNICATIONS & CONTROL**, v. 15, n. 2, 28 mar. 2020. Disponível em: <http://univagora.ro/jour/index.php/ijccc/article/view/3868>. Acesso em: 29 dez. 2020.

APERGIS, N.; PAYNE, J. E. A Dynamic Panel Study of Economic Development and the Electricity Consumption-Growth Nexus. **Energy Economics**, v. 33, n. 5, p. 770–781, set. 2011.

BASHA, S. H. S.; VINAKOTA, S. K.; DUBEY, S. R.; et al. AutoFCL: Automatically Tuning Fully Connected Layers for Handling Small Dataset. **Neural Computing and Applications**, 4 jan. 2021. Disponível em: <http://arxiv.org/abs/2001.11951>. Acesso em: 28 jan. 2021.

BASHA, S. H. S.; VINAKOTA, S. K.; PULABAIGARI, V.; et al. AutoTune: Automatically Tuning Convolutional Neural Networks for Improved Transfer Learning. **Neural Networks**, v. 133, p. 112–122, jan. 2021.

BASHEER, I. A.; HAJMEER, M. Artificial Neural Networks: Fundamentals, Computing, Design, and Application. **Journal of Microbiological Methods**, Neural Computting in Micrbiology. v. 43, n. 1, p. 3–31, 1 dez. 2000.

BEAN, J. C. Genetic Algorithms and Random Keys for Sequencing and Optimization. **ORSA Journal on Computing**, v. 6, n. 2, p. 154–160, 1 maio 1994.

BENGIO, Y. Practical Recommendations for Gradient-Based Training of Deep Architectures. In: MONTAVON, G.; ORR, G. B.; MÜLLER, K.-R. (Org.). . **Neural Netw. Tricks Trade Second Ed.** Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2012. p. 437–478. Disponível em: <https://doi.org/10.1007/978-3-642-35289-8_26>. Acesso em: 3 fev. 2021.

BENT, R.; VAN HENTENRYCK, P. Spatial, Temporal, and Hybrid Decompositions for Large-Scale Vehicle Routing with Time Windows. Lecture Notes in Computer Science, 2010, Berlin, Heidelberg. *Anais*... Berlin, Heidelberg: Springer, 2010. p. 99–113.

BERGSTRA, J. et al. Algorithms for hyper-parameter optimization. NIPS'11, 12 dez. 2011, Red Hook, NY, USA. *Anais*... Red Hook, NY, USA: Curran Associates Inc., 12 dez. 2011. p. 2546–2554. . Acesso em: 27 jan. 2021.

BERGSTRA, J.; BENGIO, Y. Random search for hyper-parameter optimization. **The Journal of Machine Learning Research**, v. 13, n. null, p. 281–305, 1 fev. 2012.

BINGOL, H.; ALATAS, B. Chaotic League Championship Algorithms. **Arabian Journal for Science and Engineering**, v. 41, n. 12, p. 5123–5147, 1 dez. 2016.

BINU, D.; SELVI, M.; GEORGE, A. MKF-Cuckoo: Hybridization of Cuckoo Search and Multiple Kernel-Based Fuzzy C-Means Algorithm. **AASRI Procedia**, 2013 AASRI Conference on Intelligent Systems and Control. v. 4, p. 243–249, 1 jan. 2013.

BISHOP, C. **Pattern Recognition and Machine Learning**. New York: Springer-Verlag, 2006. Disponível em: <https://www.springer.com/gp/book/9780387310732>. Acesso em: 25 jan. 2021. (Information Science and Statistics).

BLUM, C. Ant Colony Optimization: Introduction and Recent Trends. **Physics of Life Reviews**, v. 2, n. 4, p. 353–373, 1 dez. 2005.

BORGLI, R. J. et al. Automatic Hyperparameter Optimization for Transfer Learning on Medical Image Datasets Using Bayesian Optimization. In: 2019 13TH INTERNATIONAL SYMPOSIUM ON MEDICAL INFORMATION AND COMMUNICATION TECHNOLOGY (ISMICT), maio 2019, [S.l: s.n.], maio 2019. p. 1–6.

BRAGA, A. de P.; LUDERMIR, T. B.; CARVALHO, A. C. P. de L. F. **Redes neurais artificiais**. [S.l: s.n.], 2000. Disponível em: <https://repositorio.usp.br/single.php?_id=001080665>. Acesso em: 10 mar. 2020.

BROWNLEE, J. Stochastic Hill Climbing. **Clever Algorithms Nat.-Inspired Program. Recipes**. 2. ed. Melbourne, Australia: [s.n.], 2012. . Disponível em: <http://www.cleveralgorithms.com/nature-inspired/stochastic/variable_neighborhood_search.html>. Acesso em: 14 abr. 2020.

BUKATA, L. et al. Energy Optimization of Robotic Cells. **IEEE Transactions on Industrial Informatics**, v. 13, n. 1, p. 92–102, fev. 2017.

BURNET, F. M. **The clonal selection theory of acquired immunity**. Nashville,: Vanderbilt University Press, 1959. p. 1–232Disponível em: <https://www.biodiversitylibrary.org/bibliography/8281>.

CAI, T.; PAN, F.; CHEN, J. Adaptive particle swarm optimization algorithm. In: FIFTH WORLD CONGRESS ON INTELLIGENT CONTROL AND AUTOMATION (IEEE CAT. NO.04EX788), jun. 2004, [S.l: s.n.], jun. 2004. p. 2245- 2247 Vol.3.

ČANGALOVIĆ, M. M. et al. Tabu Search: A Brief Survey and Some Real-Life Applications. **The Yugoslav Journal of Operations Research**, v. 6, n. 11, p. 5–17, 1996.

CASTRO, L. N. D.; ZUBEN, F. J. V. The Clonal Selection Algorithm with Engineering Applications. 2002, [S.l.]: Morgan Kaufmann, 2002. p. 36–37.

CHANG, H.-H. et al. Feature Extraction-Based Hellinger Distance Algorithm for Nonintrusive Aging Load Identification in Residential Buildings. **IEEE Transactions on Industry Applications**, v. 52, n. 3, p. 2031–2039, maio 2016.

CHEN, H.; WANG, Y.-H.; FAN, C.-H. A Convolutional Autoencoder-Based Approach with Batch Normalization for Energy Disaggregation. **The Journal of Supercomputing**, 10 jul. 2020. Disponível em: <https://doi.org/10.1007/s11227-020-03375-y>. Acesso em: 26 dez. 2020.

CHEN, K. et al. Convolutional Sequence to Sequence Non-Intrusive Load Monitoring. **The Journal of Engineering**, v. 2018, n. 17, p. 1860–1864, 1 nov. 2018.

CHEN, L.; CHEN, C. L. P.; LU, M. A Multiple-Kernel Fuzzy C-Means Algorithm for Image Segmentation. **IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)**, v. 41, n. 5, p. 1263–1274, out. 2011.

CHEN, M.-R. et al. An Improved Shuffled Frog-Leaping Algorithm for Job-Shop Scheduling Problem. In: 2011 SECOND INTERNATIONAL CONFERENCE ON INNOVATIONS IN BIO-INSPIRED COMPUTING AND APPLICATIONS, dez. 2011, [S.l: s.n.], dez. 2011. p. 203–206.

CHENG, C.-C.; LIU, K.-W. Optimizing Energy Savings of the Injection Molding Process by Using a Cloud Energy Management System. **Energy Efficiency**, v. 11, n. 2, p. 415–426, fev. 2018.

CHEUNG, N. J.; DING, X.-M.; SHEN, H.-B. Adaptive Firefly Algorithm: Parameter Analysis and Its Application. **PLoS ONE**, v. 9, n. 11, p. e112634, 14 nov. 2014.

CORUS, D. et al. Level-Based Analysis of Genetic Algorithms and Other Search Processes. Lecture Notes in Computer Science, 2014, Cham. *Anais*... Cham: Springer International Publishing, 2014. p. 912–921.

CRAINIC, T. G.; TOULOUSE, M. (Org.). Parallel Metaheuristics. **Fleet Manag. Logist.** Centre for Research on Transportation. [S.l.]: Springer US, 1998. . Disponível em: <https://www.springer.com/gp/book/9780792381617>. Acesso em: 15 abr. 2020.

CUI, H.; BAI, J. A New Hyperparameters Optimization Method for Convolutional Neural Networks. **Pattern Recognition Letters**, v. 125, p. 828–834, 1 jul. 2019.

DARWISH, A.; EZZAT, D.; HASSANIEN, A. E. An Optimized Model Based on Convolutional Neural Networks and Orthogonal Learning Particle Swarm Optimization Algorithm for Plant Diseases Diagnosis. **Swarm and Evolutionary Computation**, v. 52, p. 100616, 1 fev. 2020.

DASGUPTA, J.; SIKDER, J.; MANDAL, D. Modeling and Optimization of Polymer Enhanced Ultrafiltration Using Hybrid Neural-Genetic Algorithm Based Evolutionary Approach. **Applied Soft Computing**, v. 55, p. 108–126, 1 jun. 2017.

DE CASTRO, L. N.; TIMMIS, J. An artificial immune network for multimodal function optimization. In: PROCEEDINGS OF THE 2002 CONGRESS ON EVOLUTIONARY COMPUTATION. CEC'02 (CAT. NO.02TH8600), maio 2002, [S.l: s.n.], maio 2002. p. 699–704 vol.1.

DEB, K. et al. A fast and elitist multiobjective genetic algorithm: NSGA-II. **IEEE Transactions on Evolutionary Computation**, v. 6, n. 2, p. 182–197, abr. 2002.

DOKE, P. et al. Using CNN with Bayesian Optimization to Identify Cerebral Micro-Bleeds. **Machine Vision and Applications**, v. 31, n. 5, p. 36, 30 maio 2020.

DOMHAN, T.; SPRINGENBERG, J. T.; HUTTER, F. Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. IJCAI'15, 25 jul. 2015, Buenos Aires, Argentina. *Anais*... Buenos Aires, Argentina: AAAI Press, 25 jul. 2015. p. 3460–3468. . Acesso em: 20 mar. 2020.

DORIGO, M. **Optimization, Learning and Natural Algorithms**. 1992. Politecnico di Milano, 1992. Disponível em: <https://www.bibsonomy.org/bibtex/2981e8eba5bb9a64363d706f90d6067df/dalbem>. Acesso em: 21 mar. 2020.

DORIGO, M.; MANIEZZO, V.; COLORNI, A. Ant system: optimization by a colony of cooperating agents. **IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)**, v. 26, n. 1, p. 29–41, fev. 1996.

DORIGO, M.; STUTZLE, T. **Ant Colony Optimization**. [S.l.]: MIT Press, 2004. Disponível em: <https://mitpress.mit.edu/books/ant-colony-optimization>. Acesso em: 22 mar. 2020.

DOWNEY, A. B. **Think Bayes: Bayesian Statistics in Python**. 1. ed. Needham, Massachusetts: O'Reilly Media, Inc., 2013. Disponível em: <https://www.oreilly.com/library/view/think-bayes/9781491945407/>. Acesso em: 27 jan. 2021.

DU, R. et al. Identification of COPD From Multi-View Snapshots of 3D Lung Airway Tree via Deep CNN. **IEEE Access**, v. 8, p. 38907–38919, 2020.

DUAN, C.; ZHANG, T. Two-Stream Convolutional Neural Network Based on Gradient Image for Aluminum Profile Surface Defects Classification and Recognition. **IEEE Access**, v. 8, p. 172152–172165, 2020.

EBERHART, R.; KENNEDY, J. A new optimizer using particle swarm theory. In: MHS'95. PROCEEDINGS OF THE SIXTH INTERNATIONAL SYMPOSIUM ON MICRO MACHINE AND HUMAN SCIENCE, out. 1995, [S.l: s.n.], out. 1995. p. 39–43.

EIBEN, A. E.; AARTS, E. H. L.; VAN HEE, K. M. Global Convergence of Genetic Algorithms: A Markov Chain Analysis. Lecture Notes in Computer Science, 1991, Berlin, Heidelberg. *Anais*... Berlin, Heidelberg: Springer, 1991. p. 3–12.

EKICI, S. et al. Power Quality Event Classification Using Optimized Bayesian Convolutional Neural Networks. **Electrical Engineering**, 20 jul. 2020. Disponível em: <https://doi.org/10.1007/s00202-020-01066-8>. Acesso em: 28 jan. 2021.

ELOLA, A. et al. Deep Neural Networks for ECG-Based Pulse Detection during Out-of-Hospital Cardiac Arrest. **Entropy**, v. 21, n. 3, p. 305, mar. 2019.

ELSHAWI, R.; MAHER, M.; SAKR, S. Automated Machine Learning: State-of-The-Art and Open Challenges. **ArXiv**, arXiv: 1906.02287, 11 jun. 2019. Disponível em: <http://arxiv.org/abs/1906.02287>. Acesso em: 11 mar. 2020.

ENERDATA. **World Power consumption - Electricity consumption**. Disponível em: <https://yearbook.enerdata.net/electricity/electricity-domestic-consumption-data.html>. Acesso em: 3 fev. 2021.

EPE, E. de P. E. **Anuário Estatístico de Energia Elétrica 2020**. Disponível em: <https://www.epe.gov.br/pt/publicacoes-dados-abertos/publicacoes/anuario-estatistico-de-energia-eletrica>. Acesso em: 3 fev. 2021.

ESCALANTE, H. J.; MONTES, M.; SUCAR, L. E. Particle Swarm Model Selection. **Journal of Machine Learning Research**, p. 36, 2009.

FARAHANI, Sh. M. et al. A Gaussian Firefly Algorithm. **International Journal of Machine Learning and Computing**, p. 448–453, 2011.

FARIA, E. L. de. **Redes neurais convolucionais e máquinas de aprendizado extremo aplicadas ao mercado financeiro brasileiro**. 2018. Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2018. Disponível em: <http://pantheon.ufrj.br/handle/11422/11418>. Acesso em: 3 fev. 2021.

FERREIRA, L. et al. Galaxy Merger Rates up to z Similar to 3 Using a Bayesian Deep Learning Model: A Major-Merger Classifier Using IllustrisTNG Simulation Data. **The Astrophysical Journal**, v. 895, n. 2, p. 115, jun. 2020.

FEURER, M.; HUTTER, F. Hyperparameter Optimization. In: HUTTER, F.; KOTTHOFF, L.; VANSCHOREN, J. (Org.). . **Autom. Mach. Learn. Methods Syst. Chall.** The Springer Series on Challenges in Machine Learning. Cham: Springer International Publishing, 2019. p. 3–33. Disponível em: <https://doi.org/10.1007/978-3-030-05318-5_1>. Acesso em: 17 abr. 2020.

FIRMES, V. P. **Identificação não-intrusiva de cargas similares em Smart Grid usando rede neural convolucional**. 2020. Universidade Federal do Espírito Santo, São Mateus, 2020. Disponível em: <https://energia.ufes.br/pt-br/pos-graduacao/PGEN/detalhes-da-tese?id=14024>. Acesso em: 3 fev. 2021.

FOGEL, L.; OWENS, A.; WALSH, M. **Artificial intelligence through simulated evolution**. 1. ed. [S.l.]: John Wiley & Sons, Inc., 1966. v. 1. Disponível em: <https://www.bibsonomy.org/bibtex/246129d2e7486b0c75a2a4d420ba10921/danfunky>. Acesso em: 21 mar. 2020.

FRAZIER, P. I. A Tutorial on Bayesian Optimization. **arXiv:1807.02811 [cs, math, stat]**, arXiv: 1807.02811, 8 jul. 2018. Disponível em: <http://arxiv.org/abs/1807.02811>. Acesso em: 27 jan. 2021.

FRIEDMAN, J. H. Greedy Function Approximation: A Gradient Boosting Machine. **The Annals of Statistics**, v. 29, n. 5, p. 1189–1232, 2001.

FUKUSHIMA, K. Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position. **Biological Cybernetics**, v. 36, n. 4, p. 193–202, 1 abr. 1980.

FURTADO, M. I. V. **Redes Neurais Artificiais**. [S.l.]: Atena Editora, 2019. Disponível em: <https://www.atenaeditora.com.br/arquivos/ebooks/redes-neurais-artificiais-uma-abordagem-para-sala-de-aula>. Acesso em: 10 mar. 2020.

GANDIBLEUX, X. et al. (Org.). **Metaheuristics for Multiobjective Optimisation**. Berlin Heidelberg: Springer-Verlag, 2004. Disponível em: <https://www.springer.com/gp/book/9783540206378>. Acesso em: 15 abr. 2020. (Lecture Notes in Economics and Mathematical Systems).

GANGALE, F.; MENGOLINI, A.; ONYEJI, I. Consumer Engagement: An Insight from Smart Grid Projects in Europe. **Energy Policy**, v. 60, p. 621–628, set. 2013.

GEEM, Z. W. (Org.). **Music-Inspired Harmony Search Algorithm**. Berlin Heidelberg: Springer-Verlag, 2009. Disponível em: <https://www.springer.com/gp/book/9783642001840>. Acesso em: 26 mar. 2020. (Studies in Computational Intelligence).

GEEM, Z. W.; KIM, J. H.; LOGANATHAN, G. V. A New Heuristic Optimization Algorithm: Harmony Search. **SIMULATION**, 1 fev. 2001. Disponível em: <https://journals.sagepub.com/doi/10.1177/003754970107600201>. Acesso em: 26 mar. 2020.

GÉRON, A. **Hands-On Machine Learning with Scikit-Learn and TensorFlow**. [S.l: s.n.], 2017. Disponível em: <https://www.oreilly.com/library/view/hands-on-machine-learning/9781491962282/>. Acesso em: 10 mar. 2020.

GILLIS, J. M.; CHUNG, J. A.; MORSI, W. G. Designing New Orthogonal High-Order Wavelets for Nonintrusive Load Monitoring. **IEEE Transactions on Industrial Electronics**, v. 65, n. 3, p. 2578–2589, mar. 2018.

GLOVER, F. **Tabu Search Fundamentals and Uses**. Boulder: Graduate School of Business, University of Colorado, 1995.

GLOVER, F.; LAGUNA, M.; MARTÍ, R. Principles of Tabu Search. **Handb. Approx. Algorithms Metaheuristics**. 1. ed. [S.l.]: Chapman and Hall/CRC, 2007. p. 1432. Disponível em: <https://www.taylorfrancis.com/books/e/9780429143793>. Acesso em: 14 abr. 2020.

GLOVER, F.; MCMILLAN, C. The General Employee Scheduling Problem. An Integration of MS and AI. **Computers & Operations Research**, Applications of Integer Programming. v. 13, n. 5, p. 563–573, 1 jan. 1986.

GLOVER, F.; TAILLARD, E.; TAILLARD, E. A User's Guide to Tabu Search. **Annals of Operations Research**, v. 41, n. 1, p. 1–28, 1 mar. 1993.

GODFREY, L. B. **Parameterizing and Aggregating Activation Functions in Deep Neural Networks**. 2018. Arkansas University, Arkansas, 2018.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. [S.l.]: MIT Press, 2016. Disponível em: <https://www.deeplearningbook.org/>. Acesso em: 10 mar. 2020.

GOOGLE. **Descending into ML**. Disponível em: <https://developers.google.com/machine-learning/crash-course/descending-into-ml/training-and-loss?hl=pt-br>. Acesso em: 25 jan. 2021.

GU, J. et al. Recent Advances in Convolutional Neural Networks. **Pattern Recognition**, v. 77, p. 354–377, 2017.

GU, X. **The behavior of simulated annealing in stochastic optimization**. 2008. 2008. Disponível em: <https://lib.dr.iastate.edu/rtd/15383>.

HAN, D.; LIU, Q.; FAN, W. A New Image Classification Method Using CNN Transfer Learning and Web Data Augmentation. **Expert Systems with Applications**, v. 95, p. 43–56, 1 abr. 2018.

HANAFI, S. On the Convergence of Tabu Search. **Journal of Heuristics**, v. 7, n. 1, p. 47–58, 1 jan. 2001.

HANSEN, P.; MLADENOVIĆ, N. A Tutorial on Variable Neighborhood Search. **Les Cahiers Du Gerad**, 2003.

_____. Developments of Variable Neighborhood Search. In: RIBEIRO, C. C.; HANSEN, P. (Org.). . **Essays Surv. Metaheuristics**. Operations Research/Computer Science Interfaces Series. Boston, MA: Springer US, 2002. p. 415–439. Disponível em: <https://doi.org/10.1007/978-1-4615-1507-4_19>. Acesso em: 14 abr. 2020.

_____. Variable Neighborhood Search: Principles and Applications. **European Journal of Operational Research**, v. 130, n. 3, p. 449–467, 1 maio 2001.

HARVEY, H. B.; SOTARDI, S. T. The Pareto Principle. **Journal of the American College of Radiology**, v. 15, n. 6, p. 931, 1 jun. 2018.

HASHIMOTO, K. **Técnicas de otimização combinatória multiobjetivo aplicadas na estimação do desempenho elétrico de redes de distribuição.** 2004. text–Universidade de São Paulo, São Paulo, 2004. Disponível em: <http://www.teses.usp.br/teses/disponiveis/3/3143/tde-19112004-165342/>. Acesso em: 20 mar. 2020.

HAYKIN, S. **Redes Neurais**. 2. ed. Porto Alegre: Bookman, 2001.

HE, K. et al. Non-Intrusive Load Disaggregation Using Graph Signal Processing. **IEEE Transactions on Smart Grid**, v. 9, n. 3, p. 1739–1747, 2016.

HEAD, T. et al. **Scikit-optimize**. [S.l: s.n.], 2020. Disponível em: <https://zenodo.org/record/4014775#.YAhkfehKjIU>. Acesso em: 20 jan. 2021.

HEBB, D. O. **The Organization of Behavior: A Neuropsychological Theory**. [S.l: s.n.], 1949. . Acesso em: 10 mar. 2020.

HEFNY, H. A.; AZAB, S. S. Chaotic particle swarm optimization. In: 2010 THE 7TH INTERNATIONAL CONFERENCE ON INFORMATICS AND SYSTEMS (INFOS), mar. 2010, [S.l: s.n.], mar. 2010. p. 1–8.

HINTON, G. E. et al. Improving neural networks by preventing co-adaptation of feature detectors. **ArXiv**, arXiv: 1207.0580, 3 jul. 2012. Disponível em: <http://arxiv.org/abs/1207.0580>. Acesso em: 10 mar. 2020.

HIZUKURI, A. et al. Computer-Aided Diagnosis Scheme for Distinguishing Between Benign and Malignant Masses on Breast DCE-MRI Images Using Deep Convolutional Neural Network with Bayesian Optimization. **Journal of Digital Imaging**, 6 nov. 2020. Disponível em: <https://europepmc.org/article/med/33159279>. Acesso em: 4 fev. 2021.

HOLLAND, J. H. **Adaptation in natural and artificial systems**. Oxford, England: U Michigan Press, 1975. p. viii, 183(Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence).

HOLLAND, John Henry. **Adaptation in Natural and Artificial Systems**. 1. ed. Cambridge, Massachusetts: The MIT Press, 1992. Disponível em: <https://mitpress.mit.edu/books/adaptation-natural-and-artificial-systems>. Acesso em: 26 dez. 2020.

HU, J.; HUANG, M.-C.; YU, X. Efficient Mapping of Crash Risk at Intersections with Connected Vehicle Data and Deep Learning Models. **Accident Analysis & Prevention**, v. 144, p. 105665, 1 set. 2020.

HUANG, F.; WANG, L.; HE, Q. An Effective Co-Evolutionary Differential Evolution for Constrained Optimization. **Applied Mathematics and Computation**, v. 186, n. 1, p. 340–356, 1 mar. 2007.

HUANG, K. W.; CHEN, J. L.; YANG, C. S. A Memetic Gravitation Search Algorithm for Solving Permutation Flow Shop Scheduling. **Advanced Materials Research**, DOI: 10.4028/www.scientific.net/AMR.1079-1080.626, v. 1079–1080, p. 626–630, 2015. Disponível em: </AMR.1079-1080.626>. Acesso em: 15 abr. 2020.

HUBEL, D. H.; WIESEL, T. N. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. **The Journal of Physiology**, 1962. Disponível em: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1359523/>. Acesso em: 10 mar. 2020.

HUSSEINZADEH KASHAN, A. League Championship Algorithm (LCA): An Algorithm for Global Optimization Inspired by Sport Championships. **Applied Soft Computing**, v. 16, p. 171–200, 1 mar. 2014.

HUTTER, F.; HOOS, H. H.; LEYTON-BROWN, K. Sequential Model-Based Optimization for General Algorithm Configuration. Lecture Notes in Computer Science, 2011, Berlin, Heidelberg. *Anais*... Berlin, Heidelberg: Springer, 2011. p. 507–523.

HUTTER, F.; KOTTHOFF, L.; VANSCHOREN, J. (Org.). **Automated Machine Learning**. Cham: Springer International Publishing, 2019. Disponível em: <http://link.springer.com/10.1007/978-3-030-05318-5>. Acesso em: 11 mar. 2020. (The Springer Series on Challenges in Machine Learning).

ITAKURA, K. et al. Estimation of Citrus Maturity with Fluorescence Spectroscopy Using Deep Learning. **Horticulturae**, v. 5, n. 1, p. 2, mar. 2019.

JACK, M. W.; STEPHENSON, J.; KHAN, I. Analysis of Greenhouse Gas Emissions in Electricity Systems Using Time-Varying Carbon Intensity. Accepted: 2018-05-03T22:47:18Z, 2018. Disponível em: <https://ourarchive.otago.ac.nz/handle/10523/8032>. Acesso em: 9 mar. 2020.

JACOBSON, S. H.; YÜCESAN, E. Analyzing the Performance of Generalized Hill Climbing Algorithms. **Journal of Heuristics**, v. 10, n. 4, p. 387–405, 1 jul. 2004.

JAIN, A. K.; JIANCHANG MAO; MOHIUDDIN, K. M. Artificial neural networks: a tutorial. **Computer**, v. 29, n. 3, p. 31–44, mar. 1996.

JI, Z.; DASGUPTA, D. Revisiting Negative Selection Algorithms. **Evolutionary Computation**, v. 15, n. 2, p. 223–251, 29 maio 2007.

JIN, Y.; LI, S.; JUNG, O. Prediction of Flow Properties on Turbine Vane Airfoil Surface From 3D Geometry With Convolutional Neural Network. In: ASME TURBO EXPO 2019: TURBOMACHINERY TECHNICAL CONFERENCE AND EXPOSITION, 5 nov. 2019, Phoenix, Arizona, USA. *Anais*... Phoenix, Arizona, USA: American Society of Mechanical Engineers Digital Collection, 5 nov. 2019. Disponível em: <https://asmedigitalcollection.asme.org/GT/proceedings/GT2019/58585/V02DT46A007/1066599>. Acesso em: 4 fev. 2021.

JONES, D. F.; MIRRAZAVI, S. K.; TAMIZ, M. Multi-Objective Meta-Heuristics: An Overview of the Current State-of-the-Art. **European Journal of Operational Research**, v. 137, n. 1, p. 1–9, 16 fev. 2002.

JONES, D. R. A Taxonomy of Global Optimization Methods Based on Response Surfaces. **Journal of Global Optimization**, v. 21, n. 4, p. 345–383, 1 dez. 2001.

JONES, D. R.; SCHONLAU, M.; WELCH, W. J. Efficient Global Optimization of Expensive Black-Box Functions. **Journal of Global Optimization**, v. 13, n. 4, p. 455–492, 1 dez. 1998.

JUNIOR, H. A. e O. et al. **Stochastic Global Optimization and Its Applications with Fuzzy Adaptive Simulated Annealing**. Berlin Heidelberg: Springer-Verlag, 2012. Disponível em: <https://www.springer.com/gp/book/9783642274787>. Acesso em: 14 abr. 2020. (Intelligent Systems Reference Library).

KALOURIS, G.; ZACHARAKI, E. I.; MEGALOOIKONOMOU, V. Improving CNN-based activity recognition by data augmentation and transfer learning. In: 2019 IEEE 17TH INTERNATIONAL CONFERENCE ON INDUSTRIAL INFORMATICS (INDIN), jul. 2019, [S.l: s.n.], jul. 2019. p. 1387–1394.

KARABOGA, D.; BASTURK, B. Artificial Bee Colony (ABC) Optimization Algorithm for Solving Constrained Optimization Problems. Lecture Notes in Computer Science, 2007, Berlin, Heidelberg. *Anais*... Berlin, Heidelberg: Springer, 2007. p. 789–798.

KATEHAKIS, M. N.; VEINOTT, A. F. Jr. The Multi-Armed Bandit Problem: Decomposition and Computation. **Mathematics of Operations Research**, 1 maio 1987. Disponível em: <https://pubsonline.informs.org/doi/abs/10.1287/moor.12.2.262>. Acesso em: 20 mar. 2020.

KAUR, E. A.; KAUR, E. M. A Comprehensive Survey of Test Functions for Evaluating the Performance of Particle Swarm Optimization Algorithm. **International Journal of Hybrid Information Technology**, v. 8, n. 5, p. 97–104, 22 mar. 2020.

KELSEY, J.; TIMMIS, J. Immune Inspired Somatic Contiguous Hypermutation for Function Optimisation. Lecture Notes in Computer Science, 2003, Berlin, Heidelberg. *Anais*... Berlin, Heidelberg: Springer, 2003. p. 207–218.

KENNEDY, J. Bare bones particle swarms. In: PROCEEDINGS OF THE 2003 IEEE SWARM INTELLIGENCE SYMPOSIUM. SIS'03 (CAT. NO.03EX706), abr. 2003, [S.l: s.n.], abr. 2003. p. 80–87.

KENNEDY, J.; EBERHART, R. Particle swarm optimization. In: PROCEEDINGS OF ICNN'95 - INTERNATIONAL CONFERENCE ON NEURAL NETWORKS, nov. 1995, [S.l: s.n.], nov. 1995. p. 1942–1948 vol.4.

KERAS. **Keras Documentation**. Disponível em: <https://keras.io/api/layers/convolution_layers/convolution1d/>. Acesso em: 3 fev. 2021a.

_____. **Keras Documentation**. Disponível em: <https://keras.io/api/callbacks/early_stopping/>. Acesso em: 3 fev. 2021b.

_____. **Keras Documentation**. Disponível em: <https://keras.io/api/losses/>. Acesso em: 25 jan. 2021c.

KHANESAR, M. A.; TESHNEHLAB, M.; SHOOREHDELI, M. A. A novel binary particle swarm optimization. In: 2007 MEDITERRANEAN CONFERENCE ON CONTROL AUTOMATION, jun. 2007, [S.l: s.n.], jun. 2007. p. 1–6.

KIM, yeon; LEE, J.; CHOE, Y. Bayesian Optimization-Based Global Optimal Rank Selection for Compression of Convolutional Neural Networks. **IEEE Access**, v. 8, p. 17605–17618, 2020.

KING, R. D.; FENG, C.; SUTHERLAND, A. STATLOG: Comparison of Classification Algorithms on Large Real-World Problems. **Applied Artificial Intelligence**, v. 9, n. 3, p. 289–333, maio 1995.

KIRCHGÄSSNER, W.; WALLSCHEID, O.; BÖCKER, J. Deep Residual Convolutional and Recurrent Neural Networks for Temperature Estimation in Permanent Magnet Synchronous Motors. In: 2019 IEEE INTERNATIONAL ELECTRIC MACHINES DRIVES CONFERENCE (IEMDC), maio 2019, [S.l: s.n.], maio 2019. p. 1439–1446.

KIRKPATRICK, S.; GELATT, C. D.; VECCHI, M. P. Optimization by Simulated Annealing. **Science**, v. 220, n. 4598, p. 671–680, 13 maio 1983.

KOBER, T. et al. Global Energy Perspectives to 2060 – WEC's World Energy Scenarios 2019. **Energy Strategy Reviews**, v. 31, p. 100523, 1 set. 2020.

KOHAVI, R.; JOHN, G. H. Automatic Parameter Selection by Minimizing Estimated Error. **Mach. Learn. Proc. 1995**. [S.l.]: Elsevier, 1995. p. 304–312. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/B9781558603776500451>. Acesso em: 19 mar. 2020.

KOJIMA, R. et al. kGCN: a graph-based deep learning framework for chemical structures. **Journal of Cheminformatics**, v. 12, n. 1, p. 32, 12 maio 2020.

KONG, W. et al. A Practical Solution for Non-Intrusive Type II Load Monitoring Based on Deep Learning and Post-Processing. **IEEE Transactions on Smart Grid**, v. 11, n. 1, p. 148–160, jan. 2020.

KOSTERS, W. A.; KOK, J. N.; FLORÉEN, P. Fourier Analysis of Genetic Algorithms. **Theoretical Computer Science**, v. 229, n. 1, p. 143–175, 6 nov. 1999.

KOUGIAS, I.; THEODOSSIOU, N. A New Music-Inspired Harmony Based Optimization Algorithm. Theory and Applications. **International Conference on Protection and Restoration of the Environment X**, 2010.

KRAUSE, J. et al. A Survey of Swarm Algorithms Applied to Discrete Optimization Problems. In: YANG, X.-S. et al. (Org.). . **Swarm Intell. Bio-Inspired Comput.** Oxford: Elsevier, 2013. p. 169–191. Disponível em: <http://www.sciencedirect.com/science/article/pii/B9780124051638000077>. Acesso em: 15 abr. 2020.

KUMAR, V.; KUMAR, D.; CHHABRA, J. K. Improved Grey Wolf Algorithm for Optimization Problems. In: INTERNATIONAL SYMPOSIUM ON "FUSION OF SCIENCE & TECHNOLOGY", 2016, Nova Deli, Índia. *Anais*... Nova Deli, Índia: [s.n.], 2016. p. 18–22.

KWASIGROCH, A.; GROCHOWSKI, M.; MIKOŁAJCZYK, A. Neural Architecture Search for Skin Lesion Classification. **IEEE Access**, v. 8, p. 9061–9071, 2020.

LAM, A. Y. S.; LI, V. O. K. Chemical-Reaction-Inspired Metaheuristic for Optimization. **IEEE Transactions on Evolutionary Computation**, v. 14, n. 3, p. 381–399, jun. 2010.

LAM, A. Y. S.; LI, V. O. K.; YU, J. J. Q. Real-Coded Chemical Reaction Optimization. **IEEE Transactions on Evolutionary Computation**, v. 16, n. 3, p. 339–353, jun. 2012.

LANDA BECERRA, R.; COELLO, C. A. C. Cultured Differential Evolution for Constrained Optimization. **Computer Methods in Applied Mechanics and Engineering**, v. 195, n. 33, p. 4303–4322, 1 jul. 2006.

LE, H. T. et al. Human Motion Classification with Micro-Doppler Radar and Bayesian-Optimized Convolutional Neural Networks. In: 2018 IEEE INTERNATIONAL CONFERENCE ON ACOUSTICS, SPEECH AND SIGNAL PROCESSING (ICASSP), abr. 2018, [S.l: s.n.], abr. 2018. p. 2961–2965.

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

LI, Dan et al. Acoustic Emission Wave Classification for Rail Crack Monitoring Based on Synchrosqueezed Wavelet Transform and Multi-Branch Convolutional Neural Network. **Structural Health Monitoring**, p. 1475921720922797, 1 jun. 2020.

LI, Ding; DICK, S. Residential Household Non-Intrusive Load Monitoring via Graph-Based Multi-Label Semi-Supervised Learning. **IEEE Transactions on Smart Grid**, v. 10, n. 4, p. 4615–4627, jul. 2019.

LI, J.; HE, D. A Bayesian Optimization AdaBN-DCNN Method With Self-Optimized Structure and Hyperparameters for Domain Adaptation Remaining Useful Life Prediction. **IEEE Access**, v. 8, p. 41482–41501, 2020.

LI, Y. et al. Self-Learning Perfect Optical Chirality via a Deep Neural Network. **Physical Review Letters**, v. 123, n. 21, p. 213902, 19 nov. 2019.

LIANG, J. et al. Load Signature Study - Part I: Basic Concept, Structure, and Methodology. **IEEE Transactions on Power Delivery**, v. 25, n. 2, p. 551–560, abr. 2010.

LIANG, J. J.; SUGANTHAN, P. N. Dynamic multi-swarm particle swarm optimizer with local search. In: 2005 IEEE CONGRESS ON EVOLUTIONARY COMPUTATION, set. 2005, [S.l: s.n.], set. 2005. p. 522- 528 Vol.1.

LIANG, X. Image-Based Post-Disaster Inspection of Reinforced Concrete Bridge Systems Using Deep Learning with Bayesian Optimization. **Computer-Aided Civil and Infrastructure Engineering**, v. 34, n. 5, p. 415–430, 2019.

LIN, C.-C. et al. Key design of driving industry 4.0: joint energy-efficient deployment and scheduling in group-based industrial wireless sensor networks. **IEEE Communications Magazine**, v. 54, n. 10, p. 46–52, out. 2016.

LOHOKARE, M. et al. Optimal load dispatch using Accelerated Biogeography-Based Optimization. In: 2010 JOINT INTERNATIONAL CONFERENCE ON POWER ELECTRONICS, DRIVES AND ENERGY SYSTEMS 2010 POWER INDIA, dez. 2010, [S.l: s.n.], dez. 2010. p. 1–5.

LOUIS, S. J.; RAWLINS, G. J. E. Predicting Convergence Time for Genetic Algorithms. 1992, [S.l.]: Morgan Kaufmann, 1992. p. 141–161.

LU, Y. et al. Bayesian Optimized Deep Convolutional Network for Bearing Diagnosis. **The International Journal of Advanced Manufacturing Technology**, v. 108, n. 1, p. 313–322, 1 maio 2020.

_____. Bayesian Optimized Deep Convolutional Network for Electrochemical Drilling Process. **Journal of Manufacturing and Materials Processing**, v. 3, n. 3, p. 57, set. 2019.

LUJAN-MORENO, G. A. et al. Design of Experiments and Response Surface Methodology to Tune Machine Learning Hyperparameters, with a Random Forest Case-Study. **Expert Systems with Applications**, v. 109, p. 195–205, 1 nov. 2018.

LUQUE, G.; ALBA, E. **Parallel Genetic Algorithms**. Berlin Heidelberg: Springer-Verlag, 2011. Disponível em: <https://www.springer.com/gp/book/9783642220838>. Acesso em: 15 abr. 2020. (Studies in Computational Intelligence).

MA, L.; CUI, J.; YANG, B. Deep Neural Architecture Search with Deep Graph Bayesian Optimization. WI '19, 14 out. 2019, New York, NY, USA. *Anais*... New York, NY, USA:

Association for Computing Machinery, 14 out. 2019. p. 500–507. Disponível em: <https://doi.org/10.1145/3350546.3360740>. Acesso em: 4 fev. 2021.

MAHDAVI, M.; FESANGHARY, M.; DAMANGIR, E. An Improved Harmony Search Algorithm for Solving Optimization Problems. **Applied Mathematics and Computation**, v. 188, n. 2, p. 1567–1579, 15 maio 2007.

MANJARRES, D. et al. A Survey on Applications of the Harmony Search Algorithm. **Engineering Applications of Artificial Intelligence**, v. 26, n. 8, p. 1818–1831, 1 set. 2013.

MANTOVANI, R. G. et al. Hyper-Parameter Tuning of a Decision Tree Induction Algorithm. In: 2016 5TH BRAZILIAN CONFERENCE ON INTELLIGENT SYSTEMS (BRACIS), out. 2016, [S.l: s.n.], out. 2016. p. 37–42.

MARICHELVAM, M. K.; PRABAHARAN, T.; YANG, X. S. A Discrete Firefly Algorithm for the Multi-Objective Hybrid Flowshop Scheduling Problems. **IEEE Transactions on Evolutionary Computation**, v. 18, n. 2, p. 301–305, abr. 2014.

MASTROLILLI, M.; GAMBARDELLA, L. M. Maximum Satisfiability: How Good Are Tabu Search and Plateau Moves in the Worst-Case? **European Journal of Operational Research**, Metaheuristics and Worst-Case Guarantee Algorithms: Relations, Provable Properties and Applications. v. 166, n. 1, p. 63–76, 1 out. 2005.

MATHWORKS. **MatLab Documentation**. Disponível em: <https://www.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.flattenlayer.html>. Acesso em: 3 fev. 2021.

MCCULLOCH, W. S.; PITTS, W. A Logical Calculus of the Ideas Immanent in Nervous Activity. **The Bulletin of Mathematical Biophysics**, v. 5, n. 4, p. 115–133, 1 dez. 1943.

METROPOLIS, N. et al. Equation of State Calculations by Fast Computing Machines. **The Journal of Chemical Physics**, v. 21, n. 6, p. 1087–1092, 1 jun. 1953.

MEZURA-MONTES, E.; COELLO, C. A. C. A simple multimembered evolution strategy to solve constrained optimization problems. **IEEE Transactions on Evolutionary Computation**, v. 9, n. 1, p. 1–17, fev. 2005.

MICHIE, D. et al. (Org.). **Machine learning, neural and statistical classification**. USA: Ellis Horwood, 1995.

MICROSOFT. **Ajuste de hiperparâmetro de um modelo - Azure Machine Learning**. Disponível em: <https://docs.microsoft.com/pt-br/azure/machine-learning/how-to-tune-hyperparameters>. Acesso em: 3 fev. 2021.

MIRJALILI, S. SCA: A Sine Cosine Algorithm for Solving Optimization Problems. **Knowledge-Based Systems**, v. 96, p. 120–133, 15 mar. 2016.

MLADENOVIĆ, N.; HANSEN, P. Variable Neighborhood Search. **Computers & Operations Research**, v. 24, n. 11, p. 1097–1100, 1 nov. 1997.

MONTEIRO, T. G.; SKOURUP, C.; ZHANG, H. Optimizing CNN Hyperparameters for Mental Fatigue Assessment in Demanding Maritime Operations. **IEEE Access**, v. 8, p. 40402–40412, 2020.

MOORE, J.; CHAPMAN, R. C. Application of Particle Swarm to Multiobjective Optimization. In: DEPARTMENT OF COMPUTER SCIENCE AND SOFTWARE ENGINEERING, 1999, Auburn University. *Anais*... Auburn University: [s.n.], 1999.

MORA-GUTIÉRREZ, R. A. et al. An Optimization Algorithm Inspired by Social Creativity Systems. **Computing**, v. 94, n. 11, p. 887–914, 1 nov. 2012.

MORA-GUTIÉRREZ, R. A.; RAMÍREZ-RODRÍGUEZ, J.; RINCÓN-GARCÍA, E. A. An Optimization Algorithm Inspired by Musical Composition. **Artificial Intelligence Review**, v. 41, n. 3, p. 301–315, 1 mar. 2014.

MUHAMMAD, A.; MOUSTAFA, M. Improving Region Based CNN Object Detector Using Bayesian Optimization. In: 2018 IEEE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING, APPLICATIONS AND SYSTEMS (IPAS), dez. 2018, [S.l: s.n.], dez. 2018. p. 32–36.

NABIL, E. A Modified Flower Pollination Algorithm for Global Optimization. **Expert Systems with Applications**, v. 57, p. 192–203, 15 set. 2016.

NADDAF-SH, M.-M. et al. Real-Time Road Crack Mapping Using an Optimized Convolutional Neural Network. **Complexity**, v. 2019, 1 jan. 2019. Disponível em: <https://doi.org/10.1155/2019/2470735>. Acesso em: 4 fev. 2021.

NARASIMHAN, H. Parallel artificial bee colony (PABC) algorithm. In: 2009 WORLD CONGRESS ON NATURE BIOLOGICALLY INSPIRED COMPUTING (NABIC), dez. 2009, [S.l: s.n.], dez. 2009. p. 306–311.

NELDER, J. A.; MEAD, R. A Simplex Method for Function Minimization. **The Computer Journal**, v. 7, n. 4, p. 308–313, 1 jan. 1965.

NEMATI, M.; MOMENI, H.; BAZRKAR, N. Binary Black Holes Algorithm. **International Journal of Computer Applications**, v. 79, n. 6, p. 36–42, 18 out. 2013.

NEZAMABADI-POUR, H.; BARANI, F. Gravitational Search Algorithm: Concepts, Variants, and Operators. **Handb. Res. Mod. Optim. Algorithms Appl. Eng. Econ.** [S.l.]: IGI Global, 2016. . Disponível em: <https://www.igi-global.com/gateway/chapter/147535>. Acesso em: 30 mar. 2020.

NIETO-VESPERINAS, M.; NAVARRO, R.; FUENTES, F. J. Performance of a Simulated-Annealing Algorithm for Phase Retrieval. **JOSA A**, v. 5, n. 1, p. 30–38, 1 jan. 1988.

NOMURA, Y. et al. Pilot Study of Eruption Forecasting with Muography Using Convolutional Neural Network. **Scientific Reports**, v. 10, n. 1, p. 5272, 24 mar. 2020.

NUMPY. **NumPy**. Disponível em: <https://numpy.org/>. Acesso em: 4 jan. 2021.

OBAYYA, M. I.; EL-GHANDOUR, M.; ALROWAIS, F. Contactless Palm Vein Authentication Using Deep Learning With Bayesian Optimization. **IEEE Access**, v. 9, p. 1940–1957, 2021.

OLIVAS-PADILLA, B. E.; CHACON-MURGUIA, M. I. Classification of Multiple Motor Imagery Using Deep Convolutional Neural Networks and Spatial Filters. **Applied Soft Computing**, v. 75, p. 461–472, 1 fev. 2019.

OLSON, R. S. et al. Data-driven Advice for Applying Machine Learning to Bioinformatics Problems. **ArXiv**, arXiv: 1708.05070, 2018. Disponível em: <http://arxiv.org/abs/1708.05070>.

OMRAN, M. G. H.; MAHDAVI, M. Global-Best Harmony Search. **Applied Mathematics and Computation**, v. 198, n. 2, p. 643–656, 1 maio 2008.

OSMANI, A.; HAMIDI, M. Hybrid and Convolutional Neural Networks for Locomotion Recognition. UbiComp '18, 8 out. 2018, New York, NY, USA. *Anais*... New York, NY, USA: Association for Computing Machinery, 8 out. 2018. p. 1531–1540. Disponível em: <https://doi.org/10.1145/3267305.3267520>. Acesso em: 4 fev. 2021.

OUAARAB, A.; AHIOD, B.; YANG, X.-S. Random-Key Cuckoo Search for the Travelling Salesman Problem. **Soft Computing**, v. 19, n. 4, p. 1099–1106, 1 abr. 2015.

OZKIS, A.; BABALIK, A. Accelerated ABC (A-ABC) Algorithm for Continuous Optimization Problems. **Lecture Notes on Software Engineering**, p. 262–266, 2013.

PAIXÃO, A. R. **Sistema de classificação inteligente de cargas elétricas similares e não similares**. 2016. Universidade Federal do Espírito Santo, São Mateus, 2016. Disponível em: <http://repositorio.ufes.br>. Acesso em: 3 fev. 2021.

PAN, Q.-K. et al. A Self-Adaptive Global Best Harmony Search Algorithm for Continuous Optimization Problems. **Applied Mathematics and Computation**, v. 216, n. 3, p. 830–848, 1 abr. 2010.

PANDAS. **Pandas - Python Data Analysis Library**. Disponível em: <https://pandas.pydata.org/>. Acesso em: 4 jan. 2021.

PANT, M.; THANGARAJ, R.; ABRAHAM, A. Particle Swarm Optimization: Performance Tuning and Empirical Analysis. In: ABRAHAM, A. et al. (Org.). . **Found. Comput. Intell. Vol. 3 Glob. Optim.** Studies in Computational Intelligence. Berlin, Heidelberg: Springer, 2009. p. 101–128. Disponível em: <https://doi.org/10.1007/978-3-642-01085-9_5>. Acesso em: 22 mar. 2020.

PIRIM, H.; BAYRAKTAR, E.; EKSIOGLU, B. Tabu Search: A Comparative Study. **Tabu Search**, 1 set. 2008. Disponível em: <https://www.intechopen.com/books/tabu_search/tabu_search__a_comparative_study>. Acesso em: 14 abr. 2020.

PROVOST, F.; JENSEN, D.; OATES, T. Efficient progressive sampling. KDD '99, 1 ago. 1999, San Diego, California, USA. *Anais*... San Diego, California, USA: Association for Computing Machinery, 1 ago. 1999. p. 23–32. Disponível em: <https://doi.org/10.1145/312129.312188>. Acesso em: 20 mar. 2020.

RAO, R. V.; SAVSANI, V. J.; VAKHARIA, D. P. Teaching–Learning-Based Optimization: A Novel Method for Constrained Mechanical Design Optimization Problems. **Computer-Aided Design**, v. 43, n. 3, p. 303–315, 1 mar. 2011.

RAQUEL, C. R.; NAVAL JR, P. C. An effective use of crowding distance in multiobjective particle swarm optimization. GECCO '05, 25 jun. 2005, Washington DC, USA. *Anais*... Washington DC, USA: Association for Computing Machinery, 25 jun. 2005. p. 257–264. Disponível em: <https://doi.org/10.1145/1068009.1068047>. Acesso em: 15 abr. 2020.

RASHEDI, E.; NEZAMABADI-POUR, H.; SARYAZDI, S. GSA: A Gravitational Search Algorithm. **Information Sciences**, Special Section on High Order Fuzzy Sets. v. 179, n. 13, p. 2232–2248, 13 jun. 2009.

RECHENBERG, I. **Evolution Strategy: Optimization of Technical Systems by Means of Biological Evolution**. [S.l.]: John Wiley & Sons, Inc., 1975. v. 86. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1002/fedr.19750860506>. Acesso em: 21 mar. 2020.

REEVES, C. R. (Org.). **Modern heuristic techniques for combinatorial problems**. USA: John Wiley & Sons, Inc., 1993.

REYES-SIERRA, M.; COELLO, C. A. C. Multi-Objective particle swarm optimizers: A survey of the state-of-the-art. **International Journal of Computational Inteligence Research**, v. 2, n. 3, p. 287–308, 2006.

RHODES, A. D. et al. Bayesian optimization for refining object proposals. In: 2017 SEVENTH INTERNATIONAL CONFERENCE ON IMAGE PROCESSING THEORY, TOOLS AND APPLICATIONS (IPTA), nov. 2017, [S.l: s.n.], nov. 2017. p. 1–7.

RIBEIRO, C. C.; ROSSETI, I. Efficient Parallel Cooperative Implementations of GRASP Heuristics. **Parallel Computing**, v. 33, n. 1, p. 21–35, 1 fev. 2007.

RIPLEY, B. D. Statistical Aspects of Neural Networks. In: NETWORKS AND CHAOS: STATISTICAL AND PROBABILISTIC ASPECTS, 1993, [S.l.]: Chapman & Hall, 1993. p. 40–123. Disponível em: <https://www.bibsonomy.org/bibtex/1009e7b09b4ed084d68eef4e931bec4bc/idsia>. Acesso em: 19 mar. 2020.

RODRIGUES, D. et al. Binary Flower Pollination Algorithm and Its Application to Feature Selection. In: YANG, X.-S. (Org.). . **Recent Adv. Swarm Intell. Evol. Comput.** Studies in Computational Intelligence. Cham: Springer International Publishing, 2015. . Disponível em: <https://doi.org/10.1007/978-3-319-13826-8_5>. Acesso em: 15 abr. 2020.

ROSENBLATT, F. The Perceptron: A Probabilistic Model for Information Storage and Organization in The Brain. **Psychological Review**, p. 65–386, 1958.

RUDER, S. An overview of gradient descent optimization algorithms. arXiv: 1609.04747, 2016. Disponível em: <http://arxiv.org/abs/1609.04747>. Acesso em: 10 mar. 2020.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning Representations by Back-Propagating Errors. **Nature**, v. 323, n. 6088, p. 533–536, out. 1986.

RUSSAKOVSKY, O. et al. ImageNet Large Scale Visual Recognition Challenge. **International Journal of Computer Vision**, v. 115, n. 3, p. 211–252, 1 dez. 2015.

RUSSELL, S.; NORVIG, P. **Artificial Intelligence: A Modern Approach**. 4. ed. [S.l.]: Pearson, 2020. Disponível em: <http://aima.cs.berkeley.edu/>. Acesso em: 22 dez. 2020.

SAHUMBAIEV, I. et al. 3D-CNN HadNet classification of MRI for Alzheimer's Disease diagnosis. In: 2018 IEEE NUCLEAR SCIENCE SYMPOSIUM AND MEDICAL IMAGING CONFERENCE PROCEEDINGS (NSS/MIC), nov. 2018, [S.l: s.n.], nov. 2018. p. 1–4.

SALAMA, A.; HASSANIEN, A. E.; FAHMY, A. Sheep Identification Using a Hybrid Deep Learning and Bayesian Optimization Approach. **IEEE Access**, v. 7, p. 31681–31687, 2019.

SALEM, S. A. BOA: A novel optimization algorithm. In: 2012 INTERNATIONAL CONFERENCE ON ENGINEERING AND TECHNOLOGY (ICET), out. 2012, [S.l: s.n.], out. 2012. p. 1–5.

SALLEH, M. N. B. M.; HUSSAIN, K. Accelerated mine blast algorithm for ANFIS training for solving classification problems. In: INTERNATIONAL CONFERENCE ON RECENT TRENDS IN COMPUTER SCIENCE AND ELECTRONICS ENGINEERING (RTCSE'16), 2016, Kuala Lumpur, Malaysia. *Anais*... Kuala Lumpur, Malaysia: [s.n.], 2016.

SAMEEN, M. I.; PRADHAN, B.; LEE, S. Application of Convolutional Neural Networks Featuring Bayesian Optimization for Landslide Susceptibility Assessment. **CATENA**, v. 186, p. 104249, 1 mar. 2020.

SANDERS, S.; GIRAUD-CARRIER, C. Informing the Use of Hyperparameter Optimization Through Metalearning. In: 2017 IEEE INTERNATIONAL CONFERENCE ON DATA MINING (ICDM), nov. 2017, [S.l: s.n.], nov. 2017. p. 1051–1056.

SAYADI, M.; RAMEZANIAN, R.; GHAFFARI-NASAB, N. A Discrete Firefly Meta-Heuristic with Local Search for Makespan Minimization in Permutation Flow Shop Scheduling Problems. **International Journal of Industrial Engineering Computations**, v. 1, n. 1, p. 1–10, 2010.

SAYED, S. A.-F.; NABIL, E.; BADR, A. A Binary Clonal Flower Pollination Algorithm for Feature Selection. **Pattern Recognition Letters**, v. 77, p. 21–27, 1 jul. 2016.

SAYOTI, F.; RIFFI, M. E. Random-Keys Golden Ball Algorithm for Solving Traveling Salesman Problem. **International Review on Modelling and Simulations (IREMOS)**, v. 8, n. 1, p. 84-89–89, 28 fev. 2015.

SCHNEIDER, R. K. **Identificação inteligente de cargas elétricas similares em Smart Grid**. 2018. Universidade Federal do Espírito Santo, São Mateus, 2018. Disponível em: <http://repositorio.ufes.br>. Acesso em: 3 fev. 2021.

SCIKIT-LEARN. **Partial Dependence and Individual Conditional Expectation plots**. Disponível em: <https://scikit-learn.org/stable/modules/partial_dependence.html>. Acesso em: 1 fev. 2021a.

_____. **Scikit-learn**. Disponível em: <https://scikit-learn.org/stable/>. Acesso em: 3 fev. 2021b.

_____. **Scikit-learn**. Disponível em: <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html>. Acesso em: 3 fev. 2021c.

SHAHRIARI, B. et al. Taking the Human Out of the Loop: A Review of Bayesian Optimization. **Proceedings of the IEEE**, v. 104, n. 1, p. 148–175, jan. 2016.

SHAHRIARI, Bobak; BOUCHARD-COTE, A.; FREITAS, N. Unbounded Bayesian Optimization via Regularization. In: ARTIFICIAL INTELLIGENCE AND STATISTICS, 2 maio 2016, [S.l.]: PMLR, 2 maio 2016. p. 1168–1176. Disponível em: <http://proceedings.mlr.press/v51/shahriari16.html>. Acesso em: 4 fev. 2021.

SHAO, W.; GEISSLER, C.; SIVRIKAYA, F. Graduated Optimization of Black-Box Functions. **ArXiv**, arXiv: 1906.01279, 4 jun. 2019. Disponível em: <http://arxiv.org/abs/1906.01279>. Acesso em: 21 mar. 2020.

SHARAFI, Y.; KHANESAR, M. A.; TESHNEHLAB, M. Discrete binary cat swarm optimization algorithm. In: 2013 3RD IEEE INTERNATIONAL CONFERENCE ON COMPUTER, CONTROL AND COMMUNICATION (IC4), set. 2013, [S.l: s.n.], set. 2013. p. 1–6.

SHAW, P. Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems. Lecture Notes in Computer Science, 1998, Berlin, Heidelberg. *Anais*... Berlin, Heidelberg: Springer, 1998. p. 417–431.

SHI, D. et al. Designing a lightweight 1D convolutional neural network with Bayesian optimization for wheel flat detection using carbody accelerations. **International Journal of Rail Transportation**, v. 0, n. 0, p. 1–31, 24 jul. 2020.

SHIN, C. et al. Data Requirements for Applying Machine Learning to Energy Disaggregation. **Energies**, v. 12, n. 9, p. 1696, 5 maio 2019.

SILVA, I. N. da; SPATTI, D. H.; FLAUZINO, R. A. **Redes Neurais Artificiais para Engenharia e Ciências Aplicadas**. 2. ed. [S.l.]: ArtLiber Editora LTDA, 2016. Disponível em: <https://artliber.com.br/index.php?route=product/product&product_id=77>. Acesso em: 3 fev. 2021.

SILVA, R. D. **Aprendizagem de máquina aplicada a métodos de classificação de supernovas**. 2018. Universidade Federal do Espírito Santo, Vitória, 2018. Disponível em: <http://repositorio.ufes.br>. Acesso em: 3 fev. 2021.

SINGH, R. D.; MITTAL, A.; BHATIA, R. K. 3D Convolutional Neural Network for Object Recognition: A Review. **Multimedia Tools and Applications**, v. 78, n. 12, p. 15951–15995, 2018.

SISINNI, E. et al. Industrial Internet of Things: Challenges, Opportunities, and Directions. **IEEE Transactions on Industrial Informatics**, v. 14, n. 11, p. 4724–4734, nov. 2018.

SLIVKINS, A. Introduction to Multi-Armed Bandits. **ArXiv**, arXiv: 1904.07272, 29 set. 2019. Disponível em: <http://arxiv.org/abs/1904.07272>. Acesso em: 20 mar. 2020.

SNASELOVA, P.; ZBORIL, F. Genetic Algorithm Using Theory of Chaos. **Procedia Computer Science**, International Conference On Computational Science, ICCS 2015. v. 51, p. 316–325, 1 jan. 2015.

SOEIRO, F. J. da C. P.; BECCENERI, J. C.; NETO, A. J. S. Recozimento Simulado (Simulated Annealing). **Téc. Intel. Comput. Inspiradas Na Nat. Apl. Em Probl. Inversos Em Transferência Radiativa**. São Carlos: Sociedade Brasileira de Matemática Aplicada e Computacional, 2009. p. 35–42. Disponível em: <http://mtc-m16d.sid.inpe.br/col/sid.inpe.br/mtc-m19@80/2009/08.21.17.02.53/doc/mirrorget.cgi?metadatarepository=sid.inpe.br/mtc-m19@80/2010/01.20.19.25.07&choice=full&languagebutton=pt-BR>. Acesso em: 30 mar. 2020.

SPECHT, D. F. Probabilistic Neural Networks. **Neural Networks**, v. 3, n. 1, p. 109–118, 1 jan. 1990.

SRINIVAS, N.; DEB, K. Muiltiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. **Evolutionary Computation**, v. 2, n. 3, p. 221–248, set. 1994.

SRIVASTAVA, N. et al. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. **Journal of Machine Learning Research**, v. 15, p. 1929–1958, 2014.

STAMOULIS, D. et al. Designing adaptive neural networks for energy-constrained image classification. ICCAD '18, 5 nov. 2018, New York, NY, USA. *Anais*... New York, NY, USA: Association for Computing Machinery, 5 nov. 2018. p. 1–8. Disponível em: <https://doi.org/10.1145/3240765.3240796>. Acesso em: 4 fev. 2021.

STEIN, B. van; WANG, H.; BÄCK, T. Automatic Configuration of Deep Neural Networks with Parallel Efficient Global Optimization. In: 2019 INTERNATIONAL JOINT CONFERENCE ON NEURAL NETWORKS (IJCNN), jul. 2019, [S.l: s.n.], jul. 2019. p. 1–7.

STÜTZLE, T.; HOOS, H. H. MAX–MIN Ant System. **Future Generation Computer Systems**, v. 16, n. 8, p. 889–914, 1 jun. 2000.

SUGIHARA, K. Measures for Performance Evaluation of Genetic Algorithms. 1997, [S.l: s.n.], 1997. p. 172–175.

TABATABAEI, S. M.; DICK, S.; XU, W. Toward Non-Intrusive Load Monitoring via Multi-Label Classification. **IEEE Transactions on Smart Grid**, v. 8, n. 1, p. 26–40, jan. 2017.

TAILLARD, E. D. **La programmation a memoire adaptative et les algorithmes pseudo-gloutons**. Technical Report. [S.l.]: Istituto Dalle Molle Di Studi Sull Intelligenza Artificiale, 1998.

TAILLARD, É. D.; VOSS, S. Popmusic — Partial Optimization Metaheuristic under Special Intensification Conditions. In: RIBEIRO, C. C.; HANSEN, P. (Org.). . **Essays Surv. Metaheuristics**. Operations Research/Computer Science Interfaces Series. Boston, MA: Springer US, 2002. p. 613–629. Disponível em: <https://doi.org/10.1007/978-1-4615-1507-4_27>. Acesso em: 14 abr. 2020.

TALBI, E.-G. **Metaheuristics**. [S.l.]: Wiley, 2009. Disponível em: <https://www.wiley.com/en-us/Metaheuristics%3A+From+Design+to+Implementation+-p-9780470278581>. Acesso em: 10 mar. 2020.

THORNTON, C. et al. Auto-WEKA: combined selection and hyperparameter optimization of classification algorithms. KDD '13, 11 ago. 2013, Chicago, Illinois, USA. *Anais*... Chicago, Illinois, USA: Association for Computing Machinery, 11 ago. 2013. p. 847–855. Disponível em: <https://doi.org/10.1145/2487575.2487629>. Acesso em: 19 mar. 2020.

TRIVEDI, I. N. et al. Novel Adaptive Whale Optimization Algorithm for Global Optimization. **Indian Journal of Science and Technology**, v. 9, n. 38, 21 out. 2016. Disponível em: <http://www.indjst.org/index.php/indjst/article/view/101939>. Acesso em: 14 abr. 2020.

TSENG, L.-Y.; LIANG, S.-C. A Hybrid Metaheuristic for the Quadratic Assignment Problem. **Computational Optimization and Applications**, v. 34, n. 1, p. 85–113, 1 maio 2006.

U.S. DEPARTMENT OF ENERGY. **Smart Grid**. Disponível em: <https://www.smartgrid.gov/the_smart_grid/smart_grid.html>. Acesso em: 3 fev. 2021.

VAN LAARHOVEN, P. J. M.; AARTS, E. H. L. Performance of the Simulated Annealing Algorithm. In: VAN LAARHOVEN, P. J. M.; AARTS, E. H. L. (Org.). . **Simulated Annealing Theory Appl.** Mathematics and Its Applications. Dordrecht: Springer Netherlands, 1987. p. 77–98. Disponível em: <https://doi.org/10.1007/978-94-015-7744-1_6>. Acesso em: 30 mar. 2020.

VIANNA, V. W.; CELESTE, W. C.; FREITAS, R. R. de. Energy Efficiency in the Context of Industry 4.0. **International Journal of Advanced Engineering Research and Science**, v. 6, n. 12, 7 dez. 2019. Disponível em: <https://ijaers.com/detail/energy-efficiency-in-the-context-of-industry-4-0/>. Acesso em: 8 fev. 2021.

WALDMANN, P.; PFEIFFER, C.; MÉSZÁROS, G. Sparse Convolutional Neural Networks for Genome-Wide Prediction. **Frontiers in Genetics**, v. 11, 2020. Disponível em: <https://www.frontiersin.org/articles/10.3389/fgene.2020.00025/full>. Acesso em: 4 fev. 2021.

WALTON, S. et al. Modified Cuckoo Search: A New Gradient Free Optimisation Algorithm. **Chaos, Solitons & Fractals**, v. 44, n. 9, p. 710–718, 1 set. 2011.

WANG, C. et al. Research on Bearing Fault Diagnosis Method Based on an Adaptive Anti-Noise Network under Long Time Series. **Sensors (Basel, Switzerland)**, v. 20, n. 24, 8 dez. 2020.

WANG, H. et al. Gaussian Bare-Bones Differential Evolution. **IEEE Transactions on Cybernetics**, v. 43, n. 2, p. 634–647, abr. 2013.

WANG, Y. et al. Review of Smart Meter Data Analytics: Applications, Methodologies, and Challenges. **IEEE Transactions on Smart Grid**, v. 10, n. 3, p. 3125–3148, maio 2019.

WEI, S. et al. Automatic Modulation Recognition Using Neural Architecture Search. In: 2019 INTERNATIONAL CONFERENCE ON HIGH PERFORMANCE BIG DATA AND

INTELLIGENT SYSTEMS (HPBD&IS), 1 maio 2019, [S.l.]: IEEE Computer Society, 1 maio 2019. p. 151–156. Disponível em: <https://www.computer.org/csdl/proceedings-article/hpbd&is/2019/08735458/1aPuQLxZn1u>. Acesso em: 28 jan. 2021.

WRIGHT, A. H. Genetic Algorithms for Real Parameter Optimization. In: RAWLINS, G. J. E. (Org.). . **Found. Genet. Algorithms**. [S.l.]: Elsevier, 1991. v. 1. p. 205–218. Disponível em: <http://www.sciencedirect.com/science/article/pii/B9780080506845500161>. Acesso em: 15 abr. 2020.

WU, X. et al. A Nonintrusive Fast Residential Load Identification Algorithm Based on Frequency-Domain Template Filtering: NONINTRUSIVE FAST RESIDENTIAL LOAD IDENTIFICATION ALGORITHM. **IEEJ Transactions on Electrical and Electronic Engineering**, v. 12, p. S125–S133, jun. 2017.

XIA, M. et al. Dilated Residual Attention Network for Load Disaggregation. **Neural Computing and Applications**, v. 31, n. 12, p. 8931–8953, dez. 2019.

YANG, H.; JIAO, S.; SUN, P. Bayesian-Convolutional Neural Network Model Transfer Learning for Image Detection of Concrete Water-Binder Ratio. **IEEE Access**, v. 8, p. 35350–35367, 2020.

YANG, X.-S. A New Metaheuristic Bat-Inspired Algorithm. In: GONZÁLEZ, J. R. et al. (Org.). . **Nat. Inspired Coop. Strateg. Optim. NICSO 2010**. Studies in Computational Intelligence. Berlin, Heidelberg: Springer, 2010a. p. 65–74. Disponível em: <https://doi.org/10.1007/978-3-642-12538-6_6>. Acesso em: 30 mar. 2020.

_____. Firefly algorithm. **Nat.-Inspired Metaheuristic Algorithms**. 2. ed. [S.l.]: Luniver Press, 2010b. p. 128.

_____. Flower Pollination Algorithm for Global Optimization. Lecture Notes in Computer Science, 2012, Berlin, Heidelberg. *Anais*... Berlin, Heidelberg: Springer, 2012. p. 240–249.

YANG, X.-S.; DEB, S. Cuckoo Search via Lévy flights. In: 2009 WORLD CONGRESS ON NATURE BIOLOGICALLY INSPIRED COMPUTING (NABIC), dez. 2009, [S.l: s.n.], dez. 2009. p. 210–214.

YANG, X.-S.; DEB, S.; FONG, S. Accelerated Particle Swarm Optimization and Support Vector Machine for Business Optimization and Applications. Communications in Computer and Information Science, 2011, Berlin, Heidelberg. *Anais*... Berlin, Heidelberg: Springer, 2011. p. 53–66.

YOO, D. G.; KIM, J. H.; GEEM, Z. W. Overview of Harmony Search Algorithm and Its Applications in Civil Engineering. **Evolutionary Intelligence**, v. 7, n. 1, p. 3–16, 1 abr. 2014.

YOON, H.-J. et al. Model-based Hyperparameter Optimization of Convolutional Neural Networks for Information Extraction from Cancer Pathology Reports on HPC. In: 2019 IEEE EMBS INTERNATIONAL CONFERENCE ON BIOMEDICAL HEALTH INFORMATICS (BHI), maio 2019, [S.l: s.n.], maio 2019. p. 1–4.

YOON, S.-J. et al. Automatic Multi-Class Intertrochanteric Femur Fracture Detection from CT Images Based on AO/OTA Classification Using Faster R-CNN-BO Method. **Journal of Applied Biomedicine**, v. 18, n. 4, p. 97–105, 14 dez. 2020.

ZAMANI, A.; BARAKATI, S. M.; YOUSOFI-DARMIAN, S. Design of a Fractional Order PID Controller Using GBMO Algorithm for Load–Frequency Control with Governor Saturation Consideration. **ISA Transactions**, v. 64, p. 56–66, 1 set. 2016.

ZAVALA, A. E. M.; AGUIRRE, A. H.; DIHARCE, E. R. V. Particle evolutionary swarm optimization algorithm (PESO). In: SIXTH MEXICAN INTERNATIONAL CONFERENCE ON COMPUTER SCIENCE (ENC'05), set. 2005, [S.l: s.n.], set. 2005. p. 282–289.

ZAVALA, G. R. et al. A Survey of Multi-Objective Metaheuristics Applied to Structural Optimization. **Structural and Multidisciplinary Optimization**, v. 49, n. 4, p. 537–558, 1 abr. 2014.

ZHAN, X.; ZHAO, W. Classification of Carbon Fiber Reinforced Polymer Defects Based on One-Dimensional CNN. **Laser & Optoelectronics Progress**, v. 57, n. 10, p. 101013, 2020.

ZHANG, L. et al. A Task Scheduling Algorithm Based on PSO for Grid Computing. **International Journal of Computational Intelligence Research**, v. 4, n. 1, p. 37–44, 1 jan. 2008.

ZHANG, Y. et al. Improving object detection with deep convolutional networks via Bayesian optimization and structured prediction. In: 2015 IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION (CVPR), jun. 2015, [S.l: s.n.], jun. 2015. p. 249–258.

ZHANG, Z. et al. Convolutional Neural Network Using Bayesian Optimization for Laser Welding Tailor Rolled Blanks Penetration Detection. In: ASME 2019 14TH INTERNATIONAL MANUFACTURING SCIENCE AND ENGINEERING CONFERENCE, 27 nov. 2019, [S.l.]: American Society of Mechanical Engineers Digital Collection, 27 nov. 2019. Disponível em: <https://asmedigitalcollection.asme.org/MSEC/proceedings/MSEC2019/58752/V002T03A079/1070854>. Acesso em: 4 fev. 2021.

ZHOU, Y. **Study on Genetic Algorithm Improvement and Application**. 2006. Worcester Polytechnic Institute, Worcester, Massachusetts, 2006.

ZHU, W. et al. An Optimized Convolutional Neural Network for Chatter Detection in the Milling of Thin-Walled Parts. **The International Journal of Advanced Manufacturing Technology**, v. 106, n. 9, p. 3881–3895, 1 fev. 2020.

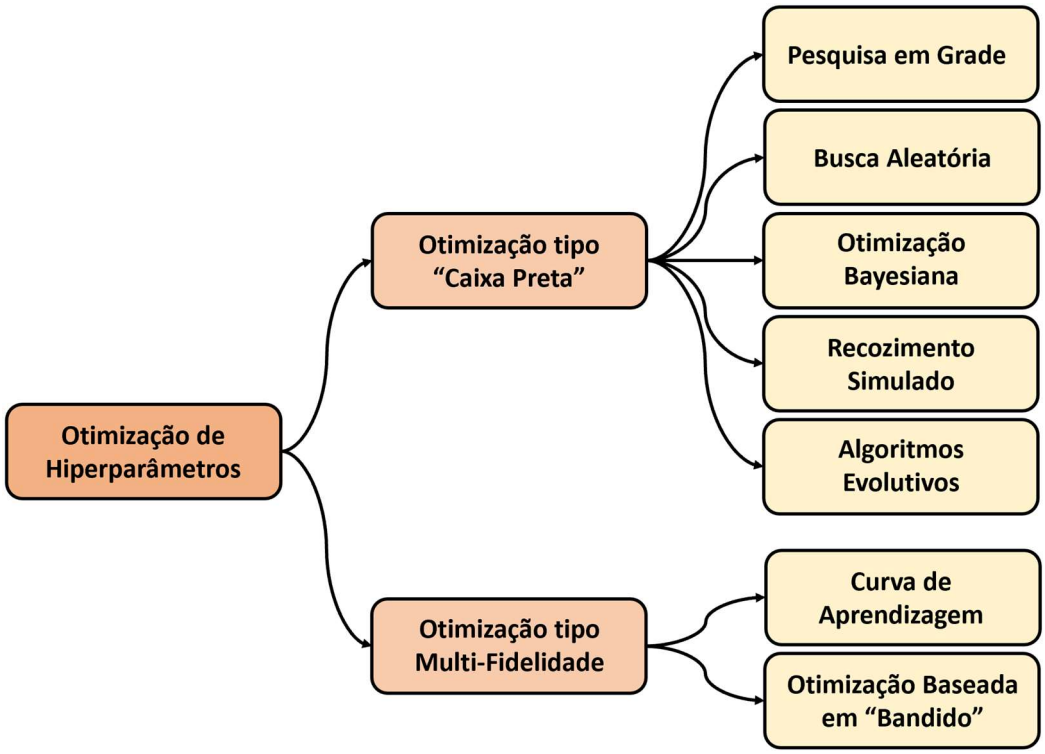
ZOU, F. et al. Bare-Bones Teaching-Learning-Based Optimization. **The Scientific World Journal**, DOI: https://doi.org/10.1155/2014/136920DOI: https://doi.org/10.1155/2014/136920, v. 2014, p. e136920, 2014. Disponível em: <https://www.hindawi.com/journals/tswj/2014/136920/>. Acesso em: 14 abr. 2020.

# APÊNDICE A – Taxonomia de Otimização Metaheurística

Tal como apresentado na Revisão de Literatura, trabalhos como (ELSHAWI; MAHER; SAKR, 2019) buscam desenvolver o ajuste fino de hiperparâmetros (hiperajuste) de forma automatizada, a partir de técnicas baseadas em conceitos matemáticos e até mesmo biológicos. O esquema da Figura 1 apresenta uma classificação resumida, das principais técnicas de otimização de hiperparâmetros em modelos de aprendizado de máquina.

De maneira geral, os autores classificam a otimização de hiperparâmetros através de duas abordagens distintas, as do tipo "caixa preta", isto é, processos de otimização que possuem pelo menos uma função cuja forma analítica não é conhecida (SHAO; GEISSLER; SIVRIKAYA, 2019), ou as do tipo multi-fidelidade, que representam as ferramentas que avaliam amostras em blocos com maior ou menor fidelidade de análise dos dados.

Figura 1 – Principais técnicas de otimização de hiperparâmetros.




Fonte: Adaptado de (ELSHAWI; MAHER; SAKR, 2019)

O caso da otimização a partir da modelagem de curva de aprendizado, como o próprio nome se refere, consiste na categoria de algoritmos que utilizam o conceito de

tais curvas como métrica de desempenho. Por exemplo, ao avaliar o desempenho de um mesmo modelo ao processar conjuntos de dados menores em um mesmo conjunto de dados.

Essa técnica foi utilizada em (PROVOST; JENSEN; OATES, 1999) ao adotar o aumento progressivo de amostragem para investigar como a acurácia do modelo se comportava, e em (KOHAVI; JOHN, 1995), ao utilizar um método que extrapolava o espaço de busca dos hiperparâmetros e, a após sucessivas execuções do algoritmo de base, utilizar a estimativa do erro de cada configuração para ajustar o modelo.

Nesta linha de raciocínio, a técnica de extrapolação da curva de aprendizado consiste em uma abordagem preditiva, como investigado em (DOMHAN; SPRINGENBERG; HUTTER, 2015), e é útil quando se deseja extrapolar a curva observada em uma determinada configuração, de modo que o treinamento seja interrompido caso seja previsto que o desempenho do modelo não atingirá o melhor desempenho de otimização obtido até o momento (HUTTER; KOTTHOFF; VANSCHOREN, 2019).

Paralelamente, o método de otimização do tipo “bandido”, cujo termo advém da contextualização popular das antigas máquinas caça-níquel (*one-armed bandit*), é uma estrutura para algoritmos que tomam decisões, sob condição de incerteza, ao longo do tempo.

Na versão básica de um algoritmo do tipo bandido de múltiplos braços (*multi-armed bandit*) (KATEHAKIS; VEINOTT, 1987), o modelo possui $K$ possibilidades de ações para escolher (neste caso, os braços), e $T$ rodadas. Durante cada rodada, o algoritmo escolhe um braço e recebe um retorno, que é observado e computado, exigindo que o modelo explore o espaço de busca com diferentes braços e diferentes retornos, a fim de otimizar o conjunto de hiperparâmetro escolhido (SLIVKINS, 2019).

Entretanto, em um contexto mais atual, onde a maioria dos problemas reais são matematicamente complexos de se reproduzir e analisar, torna-se bem-vinda a aplicação de ferramentas que entreguem resultados satisfatórios a um custo e tempo computacional também aceitáveis.

Visando abordar problemas complexos de forma eficiente, a aplicação de métodos do tipo caixa preta (*black-box*), ou metaheurísticas, constitui um conjunto de algoritmos

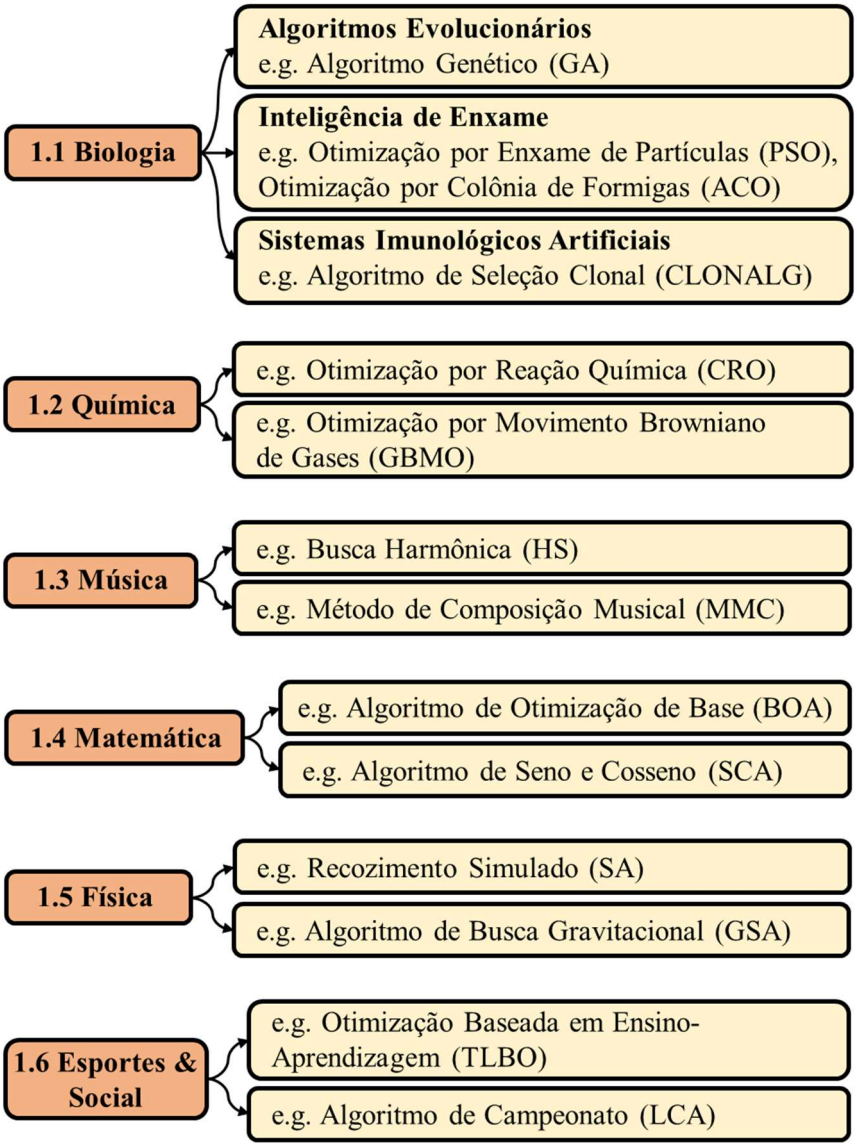
utilizados para encontrar um ou mais conjuntos de soluções satisfatórias sem ter que, necessariamente: percorrer todo o espaço de busca, que pode ser composto por uma quantidade elevada de dimensões combinatórias; ou se adaptar totalmente às restrições matemáticas de um determinado problema.

O prefixo “meta” é aplicado no sentido de indicar que um método heurístico está sobreposto a outro método do mesmo tipo, estabelecendo um novo nível heurístico. Além disso, tais métodos exploratórios costumam se referenciar em comportamentos biológicos ou físicos, presentes no mundo real (HASHIMOTO, 2004), isto é, já é naturalmente validado.

É importante ressaltar que o tópico de otimização de hiperparâmetros é vasto e permanece em constante atualização, de maneira que o foco desta revisão de literatura se dará nos principais algoritmos metaheurísticos, isto é, aqueles aplicados com maior frequência ou com aplicação multidisciplinar.

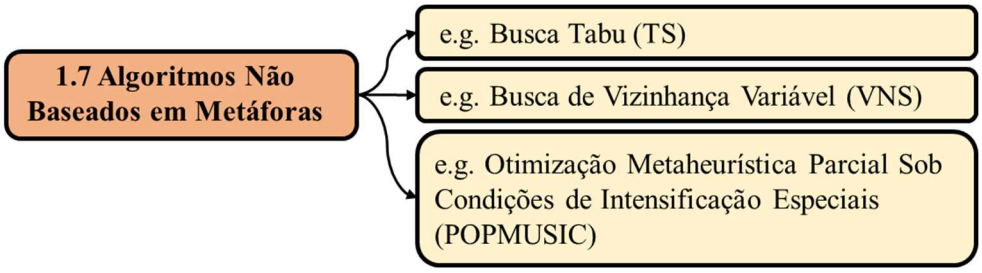
Portanto, visando explorar as definições supracitadas e fornecer um apanhado consistente e ordenado a respeito da classificação dos principais algoritmos, as Figuras 2, 3 e 4, baseadas no estudo disponível em (ABDEL-BASSET; ABDEL-FATAH; SANGAIAH, 2018), apresentam e conduzem a organização teórica deste capítulo, que será baseada em uma taxonomia das metaheurísticas usuais no problema de otimização de hiperparâmetros. Uma lista mais abrangente de metaheurísticas e seus respectivos proponentes pode ser consultada no trabalho dos referidos autores.

Figura 2 – Taxonomia de metaheurísticas baseadas em metáfora.



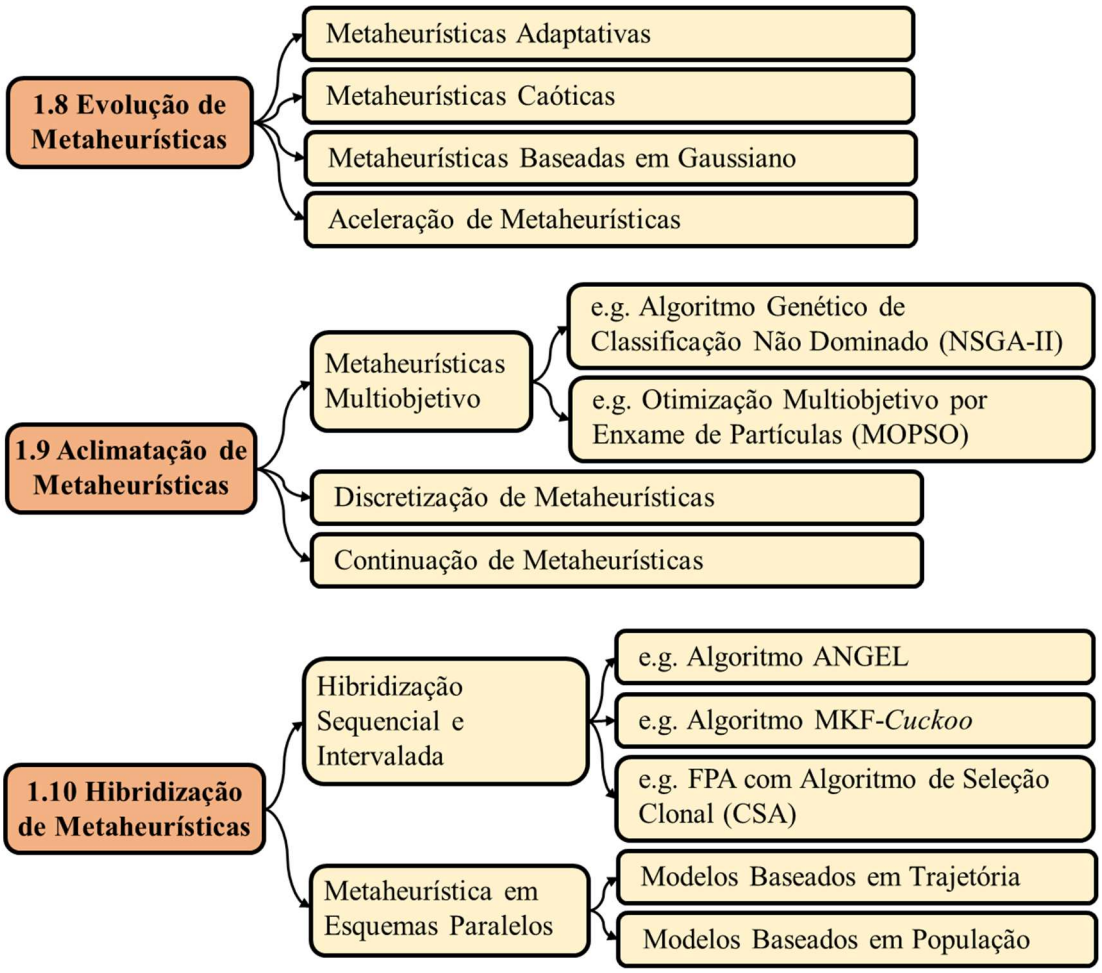

Fonte: Adaptado de (ABDEL-BASSET; ABDEL-FATAH; SANGAIAH, 2018)

Figura 3 – Taxonomia de metaheurísticas não baseadas em metáfora.




Fonte: Adaptado de (ABDEL-BASSET; ABDEL-FATAH; SANGAIAH, 2018)

Figura 4 – Taxonomia de variantes de metaheurísticas.




Fonte: Adaptado de (ABDEL-BASSET; ABDEL-FATAH; SANGAIAH, 2018)

## 1.1 Metaheurísticas Baseadas em Biologia

Em um apanhado geral, verifica-se que a maior parte das ferramentas metaheurísticas são baseadas em princípios de evolução biológica e, mais especificamente, em simular tais fenômenos (estruturas, comportamentos, padrões e outros). Nesta linha de raciocínio é possível citar três principais paradigmas: o evolucionário, o de enxames e o de sistemas imunológicos (ABDEL-BASSET; ABDEL-FATAH; SANGAIAH, 2018).

### 1.1.1 Algoritmos Evolucionários

Os algoritmos evolucionários simulam o fenômeno da progressão biológica a nível celular, a partir da seleção, cruzamento, mutação e reprodução de elementos para gerar melhores candidatos à solução.

No caso da computação evolucionária, citam-se quatro paradigmas históricos, a programação evolucionária (FOGEL; OWENS; WALSH, 1966); as estratégias evolucionárias (RECHENBERG, 1975); os algoritmos genéticos e a programação genética (HOLLAND, J. H., 1975).

De acordo com (ABDEL-BASSET; ABDEL-FATAH; SANGAIAH, 2018) o fundamento biológico ao qual os algoritmos genéticos se sustentam está no princípio de Darwin (sobrevivência do mais apto), em que é simulado o processo de evolução biológica de cromossomos através da seleção, cruzamento e mutação.

Os cromossomos representam as possíveis soluções e são examinados segundo sua aptidão. A seleção é a fase que garante o processo de geração de novas soluções e consiste na triagem de pais para a procriação. Durante o cruzamento, as partes de dois cromossomos triados são comutados. Já na mutação, as partes dos cromossomos são modificadas aleatoriamente para evitar ótimos locais.

Diversas investigações foram conduzidas ao longo dos anos, com intuito de avaliar este tipo de metaheurística em problemas diversos, como em (EIBEN; AARTS; VAN HEE, 1991), (LOUIS; RAWLINS, 1992), (SUGIHARA, 1997), (KOSTERS; KOK; FLORÉEN, 1999), (ZHOU, 2006) e (CORUS et al., 2014).

### 1.1.2 Inteligência de Enxame

No caso da inteligência de enxame, ocorre a simulação do comportamento coletivo de indivíduos em uma sociedade, como insetos, pássaros, cardumes e outros.

Esse tipo de fenômeno depende do princípio da descentralização, isto é, as soluções candidatas são atualizadas a partir da interação local entre os indivíduos e o ambiente. Logo, os elementos devem se separar e percorrer o espaço de busca no intento de identificar uma possível solução.

Segundo (ABDEL-BASSET; ABDEL-FATAH; SANGAIAH, 2018), os algoritmos baseados em inteligência de enxame mais usuais são o de otimização por

enxame de partículas (PSO) (EBERHART; KENNEDY, 1995) e otimização por colônia de formigas (ACO) (DORIGO, 1992).

O algoritmo PSO, por exemplo, é considerado um modelo de inteligência de enxame semi evolucionário, o qual é inspirado no comportamento social de animais, como pássaros em bando, ou peixes em um cardume. Nele ocorre a amostragem aleatória em uma população de soluções, com o intuito de identificar a melhor solução, repetindo este processo por uma quantidade limitada de vezes. Diversos pesquisadores investigaram o desempenho desta metaheurística, como em (PANT; THANGARAJ; ABRAHAM, 2009), (AGARWAL; SINGH; ANAND, 2013) e (KAUR; KAUR, 2020).

De forma análoga, a metaheurística denominada otimização por colônia de formigas (ACO), que foi também investigada em (DORIGO; STUTZLE, 2004), é inspirada em tais colônias e, mais especificamente, no comportamento de forrageamento destes insetos.

A fundamentação biológica deste algoritmo está baseada na comunicação entre as formigas, através de trilhas de feromônios químicos, os quais guiam outras formigas aos trajetos mais curtos em busca de alimentos. Esta característica tem sido explorada para lidar com problemas de otimização discreta (BLUM, 2005).

### 1.1.3 Sistemas Imunológicos Artificiais

Os sistemas imunológicos artificiais são baseados na teoria da imunologia, isto é, nas suas funções, princípios e modelos (DE CASTRO; TIMMIS, 2002).

Em processos de otimização, os anticorpos representam as possíveis soluções candidatas que se desenvolveram a partir da recorrência de operadores de clonagem, mutação e seleção, enquanto que o antígeno representa uma função objetivo. O conjunto de soluções adequadas é mantido em uma célula de memória.

A maioria das metaheurísticas baseadas em sistemas imunológicos dependem dos princípios de seleção clonal, como os algoritmos de seleção clonal (CLONALG) (CASTRO; ZUBEN, 2002), de rede imune artificial (opt-AINET) (DE CASTRO; TIMMIS, 2002), o algoritmo *B-Cell* (KELSEY; TIMMIS, 2003) e de seleção negativa (JI; DASGUPTA, 2007).

Em (BURNET, 1959) foi apresentada a teoria da seleção clonal, com o objetivo de investigar a resposta do sistema imunológico adaptativo (linfócitos) ao estímulo antigênico, onde foi confirmado que apenas as células capazes de identificar um antígeno se reproduzirão, ao passo que as que não identificam um antígeno não se reproduzirão.

Nesse contexto, a metaheurística CLONALG utiliza dos paradigmas da seleção clonal, pois o sistema imunológico natural pode ter objetivos múltiplos ou contraditórios e não tem razão para desenvolver uma resposta ideal. A população de anticorpos, representando as soluções candidatas, é gerada aleatoriamente e avaliada, de modo que os anticorpos de maior afinidade são clonados para gerar mais anticorpos contra o antígeno. Quando o ciclo alcança o objetivo (imunidade) as melhores soluções são transferidas para uma célula de memória (ABDEL-BASSET; ABDEL-FATAH; SANGAIAH, 2018).

## 1.2 Metaheurísticas Baseadas em Química

### 1.2.1 Otimização Baseada em Reação Química

No trabalho de (LAM; LI, 2010) é desenvolvido um algoritmo de otimização baseado em reação química (CRO), que simula tal dinâmica de interação entre as moléculas com o objetivo de atingir um estado constante de baixa energia. Neste algoritmo, as moléculas representam uma possível solução e possuem dois tipos de energia, potencial (PE) e cinética (KE), em que a energia potencial representa a função objetivo e a potencial representa a tolerância em mudar para uma estrutura menos eficiente.

Sua estruturação lógica baseia-se no movimento aleatório de colisão molécula-parede ou da colisão molécula-molécula, em um recipiente fechado. Este padrão de comportamento estabeleceu quatro reações químicas fundamentais, em termos de otimização: colisão ineficaz na parede (ON-WALL), decomposição (DEC), colisão ineficaz intermolecular (INTER) e síntese (SYN) (ABDEL-BASSET; ABDEL-FATAH; SANGAIAH, 2018).

Na reação do tipo ON-WALL, a molécula colide com a parede e se recupera, porém, perdendo relativa energia cinética. Através da aplicação da lei de conservação de energia, a porção perdida é direcionada para um *buffer* central de energia.

A reação do tipo DEC ocorrerá se a molécula colidir com a parede e se desagregar. Assim, se as novas partículas possuírem energia suficiente, novas moléculas serão geradas. Caso contrário, a energia remanescente é enviada ao *buffer* central.

A reação do tipo INTER é semelhante à do tipo ON-WALL, mas devido à colisão de duas moléculas, neste caso. Entretanto, desta vez não há perda de energia cinética, isto é, não há energia cinética sendo transferida ao *buffer* central.

Por fim, a reação do tipo SYN resulta do choque entre duas moléculas, ocorrendo sua fusão.

### 1.2.2 Otimização por Movimento Browniano de Gases

A metaheurística GBMO, proposta em (ABDECHIRI; MEYBODI; BAHRAMI, 2013), é baseada no movimento browniano, isto é, no deslocamento aleatório de partículas suspensas em uma substância fluídica. Cada possível solução é indicada por uma molécula caracterizada por sua posição, massa, velocidade e raio turbulento, as quais se movem em direção ao objetivo a partir do movimento browniano, de acordo com suas posições. O movimento molecular do fluido não possui direção especificada e se distribui irregularmente em todas as direções, como consequência da colisão contínua de tais moléculas.

Em (ABDECHIRI; MEYBODI; BAHRAMI, 2013) foi evidenciado que a metaheurística GBMO é eficiente em resolver alguns problemas de otimização. A alta velocidade das moléculas no espaço de busca permite uma melhor execução se comparada a algumas metaheurísticas conhecidas, como a otimização de enxame de partículas (PSO), algoritmo genético (GA), algoritmo competitivo imperialista (ICA) e o algoritmo de busca gravitacional (GSA). O tempo de execução do GBMO é menor que o GA, GSA e ICA, porém, um pouco mais longo que o PSO (ZAMANI; BARAKATI; YOUSOFI-DARMIAN, 2016).

## 1.3 Metaheurísticas Baseadas em Música

### 1.3.1 Busca Harmônica

Em (GEEM; KIM; LOGANATHAN, 2001) foi desenvolvido uma metaheurística baseada em um processo de improvisação musical, simulando instrumentistas

reproduzindo tons aleatórios dentro de um alcance delimitado, culminando em um vetor de harmonia. Assim, caso o conjunto de tons promova uma boa harmonia, o resultado é armazenado. A cada época de execução, a probabilidade de obter harmonias melhores, isto é, boas soluções candidatas, torna-se maior.

O algoritmo de busca harmônica inicia com a delimitação do número de vetores armazenados na memória, ou seja, o tamanho da memória de harmonia (HMS). Além disso, são definidas a taxa de consideração da memória de harmonia (HMCR) e a taxa de ajuste de afinação (PAR). Cada novo vetor é gerado de acordo com HMCR, PAR e seleção aleatória (1-HMCR). Posteriormente, este novo vetor é comparado com o vetor de pior resultado: se o novo vetor é melhor que o vetor com o pior resultado, então, o vetor de pior resultado é substituído pelo vetor recém-gerado (GEEM, 2009).

Em (KOUGIAS; THEODOSSIOU, 2010), (AHANGARAN; RAMEZANI, 2013), (ABDEL-RAOUF; METWALLY, 2013), (MANJARRES et al., 2013) e (YOO; KIM; GEEM, 2014) são desenvolvidas simulações para esta metaheurística em diferentes cenários.

### 1.3.2 Método de Composição Musical

A metaheurística designada Método de Composição Musical (MMC), investigada em (MORA-GUTIÉRREZ et al., 2012) e (MORA-GUTIÉRREZ; RAMÍREZ-RODRÍGUEZ; RINCÓN-GARCÍA, 2014), simula uma sociedade de compositores que utilizam da troca de informações entre si, com o objetivo de produzir uma composição musical.

As músicas representam as soluções candidatas e os compositores representam seus respectivos vetores n-dimensionais de variáveis de decisão. No início do processo, os compositores produzem, individualmente, um conjunto de músicas aleatórias, as quais são armazenadas na matriz de partituras. Após, é executada a troca de informações respeitando as regras de interação: se há um link entre os compositores e a pior música de um deles for melhor que a do seu associado, então a matriz de conhecimento do compositor com pior desempenho é atualizada. Por fim, novas músicas são produzidas baseadas nas matrizes de conhecimento de cada compositor, atualizando a matriz de partituras (ABDEL-BASSET; ABDEL-FATAH; SANGAIAH, 2018).

Testes evidenciaram a superioridade do MMC ao compará-lo com: o algoritmo de busca harmônica otimizado (MAHDAVI; FESANGHARY; DAMANGIR, 2007); o algoritmo de busca harmônica tipo *global-best* (OMRAN; MAHDAVI, 2008); e ao algoritmo de busca harmônica auto adaptável (PAN et al., 2010).

## 1.4 Metaheurísticas Baseadas em Matemática

### 1.4.1 Algoritmo de Otimização de Base

O algoritmo de otimização de base (BOA), proposto em (SALEM, 2012), funciona a partir da combinação dos operadores $(+, -, \times, \div)$. Em sua inicialização, é definido um parâmetro de deslocamento e cria-se uma população aleatória de soluções candidatas, em que quatro delas são escolhidas e avaliadas segundo as Equações 1.1 a 1.4:

$$x_i^{+}(j) = x_i(j) + \Delta \quad (1.1)$$

$$x_i^{-}(j) = x_i(j) - \Delta \quad (1.2)$$

$$x_i^{\times}(j) = x_i(j) \times \Delta \quad (1.3)$$

$$x_i^{\div}(j) = x_i(j) \div \Delta \quad (1.4)$$

Nas equações acima, $x_i(j)$ representa a última solução e $\Delta$ representa o parâmetro de deslocamento (todas as soluções em um intervalo predefinido).

Após a avaliação, é eleita como solução candidata aquela que apresentar o melhor desempenho (ABDEL-BASSET; ABDEL-FATAH; SANGAIAH, 2018).

Em (SALEM, 2012) foi proposto o teste comparativo de desempenho do BOA junto ao algoritmo genético (HOLLAND, J. H., 1975) e ao sistema de formigas (AS) (DORIGO; MANIEZZO; COLORNI, 1996), que obteve resultados superiores, no que concerne à taxa de sucesso e à qualidade das soluções.

### 1.4.2 Algoritmo de Seno e Cosseno

O uso de equações baseadas em seno e cosseno serviu de estímulo ao algoritmo proposto em (MIRJALILI, 2016), que inicia com a criação aleatória de uma população para, então, atualizar os locais das soluções candidatas em relação à melhor solução atual. Este processo de iteração ocorre a partir de:

$$X_i^{(t+1)} = \begin{cases} X_i^t + \left(r_1 \times sen(r_2) \times \left|r_3 P_i^t - X_i^t\right|\right), & se\ r_4 < 0{,}5 \\ X_i^t + \left(r_1 \times cos(r_2) \times \left|r_3 P_i^t - X_i^t\right|\right), & se\ r_4 \geq 0{,}5 \end{cases} \quad (1.5)$$

em que $X_i^t$ e $P_i^t$ representam, respectivamente, a última posição de uma solução e a melhor posição da solução atual na i-*ésima* dimensão e t-*ésima* iteração. O parâmetro $r_1$ determina a direção do movimento, $r_2$ define o quanto o movimento deve estar na direção (ou fora dela), $r_3$ fornece pesos aleatórios para o destino e $r_4$ alterna entre os componentes seno e cosseno da Equação 1.5.

Finalmente, visando o balanceamento entre a exploração generalizada e a exploração de regiões promissoras (determinação do mínimo global), o alcance de seno e cosseno é atualizado de forma adaptativa, fazendo:

$$r_1 = a - t \times \frac{a}{T} \quad (1.6)$$

com *a* representando uma constante e *T* representando o limite de iterações.

Em (MIRJALILI, 2016) foram realizados diversos testes comparativos de desempenho com outras metaheurísticas como PSO, GA, algoritmo de morcego (BA) (YANG, X.-S., 2010a), algoritmo de vagalume (FA) (YANG, X.-S., 2010b) e algoritmo de polinização de flores (FPA) (YANG, X.-S., 2012). Foi possível evidenciar superioridade de desempenho do SCA no experimento proposto, com número de parâmetros reduzido e boa capacidade de lidar com a otimização de ordem complexa.

## 1.5 Metaheurísticas Baseadas em Física

### 1.5.1 Recozimento Simulado

Proposto pela primeira vez em (METROPOLIS et al., 1953), o recozimento simulado (SA) foi estabelecido através de um algoritmo capaz de simular um conjunto de átomos em equilíbrio sob determinada temperatura. Em (KIRKPATRICK; GELATT; VECCHI, 1983), a mesma proposta foi aplicada, desta vez, em problemas de otimização, dando início às investigações a partir desta metaheurística.

Durante a fase de inicialização é gerada uma solução aleatória $x^c$ à temperatura *T*. Em seguida, outra solução aleatória $x^n$ é concebida, com o objetivo de calcular a diferença de energia, isto é:

$$\Delta E = f(x^n) - f(\, x^c) \qquad (1.7)$$

Dessa forma, se $\Delta E$ reduzir, $x^n$ se torna a solução atual pela qual a busca continuará; caso contrário, a solução atual pode ser aceita de acordo com uma probabilidade dada por:

$$P(\Delta, T) = e^{\left(\frac{-\Delta}{T}\right)} \qquad (1.8)$$

O deslocamento dos átomos de um material à temperatura *T* é simulado através de tais iterações. Portanto, substituindo a função objetivo pela energia e definindo as configurações atômicas como conjuntos de variáveis de projeto, é gerado um conjunto de configurações de um problema de otimização sob certa temperatura.

Inicialmente, o SA "funde" o sistema a ser otimizado a uma temperatura elevada e, em seguida, reduz a temperatura até que haja um "congelamento", tal que a função objetivo não perceba mais melhoras. Assim, para cada temperatura, o algoritmo deve ser executado um número de vezes tal que o estado de equilíbrio seja atingido. Portanto, a sequência de temperaturas e o número de rearranjos executados em cada temperatura para obter o equilíbrio representam o esquema de recozimento desta metaheurística (SOEIRO; BECCENERI; NETO, 2009).

Diversas empreitadas foram executadas no sentido de avaliar o desempenho deste algoritmo (VAN LAARHOVEN; AARTS, 1987), (NIETO-VESPERINAS; NAVARRO; FUENTES, 1988), (REEVES, 1993) e (GU, X., 2008). Em linhas gerais, foi percebido a flexibilidade para processar funções complexas e dados pouco estruturados, revelando, porém, um alto tempo de convergência, o que significa que é necessária uma análise do problema e do custo computacional frente a uma possível implementação de tal ferramenta.

### 1.5.2 Algoritmo de Busca Gravitacional

Baseado nas leis de Newton sobre gravidade e movimento, foi proposto em (RASHEDI; NEZAMABADI-POUR; SARYAZDI, 2009) o algoritmo de busca

gravitacional (GSA). Nesta metaheurística, cada solução candidata representa um corpo com posição, massa inercial e massa gravitacional ativa e passiva, em que as suas posições consistem nas soluções e suas massas consistem nas aptidões.

A atualização da solução depende da velocidade, semelhante ao que ocorre na otimização por enxame de partículas (PSO). Os corpos com maior desempenho e maior massa gravitacional possuem maior raio de atração efetiva, provocando grande atração entre si, de forma que os outros corpos com menor aptidão têm sua trajetória deslocada em direção às melhores soluções (ABDEL-BASSET; ABDEL-FATAH; SANGAIAH, 2018).

A metaheurística GSA tem sido objeto de estudos em aplicações de otimização nas mais diversas áreas. Contudo, foi investigada com maior amplitude em (NEZAMABADI-POUR; BARANI, 2016).

## 1.6 Metaheurísticas Baseadas em Esportes e Comportamento Social

### 1.6.1 Otimização Baseada em Ensino-Aprendizagem

O algoritmo de otimização baseada em ensino-aprendizagem (TLBO) (RAO; SAVSANI; VAKHARIA, 2011) busca replicar a influência de professores sobre os alunos no processo de aprendizado padrão. O professor representa a melhor solução, que compartilha seu conhecimento com os alunos, os quais representam a população de soluções, enquanto que a qualidade do ensino representa a aptidão.

O processo é classificado em duas fases. Na primeira, a melhor solução é selecionada para ser professor e a média das posições dos alunos é calculada e deslocada para a posição do professor. Na fase seguinte, o aluno *i* incrementa seu conhecimento através da interação com outro aluno *j*, selecionado aleatoriamente. Os dois alunos, *i* e *j,* são comparados através de três regras:

1. Caso as duas soluções forem viáveis, a solução com melhor aptidão é selecionada.

2. Caso uma solução seja viável e a outra inviável, a solução viável é selecionada.

3. Caso ambas as soluções sejam inviáveis, é selecionada a solução com menor transgressão de restrições.

Esse processo de iteração, portanto, implica que o conhecimento do aluno *i* será modificado se o conhecimento do aluno selecionado *j* for superior (ABDEL-BASSET; ABDEL-FATAH; SANGAIAH, 2018).

Ainda em (RAO; SAVSANI; VAKHARIA, 2011), a eficiência da metaheurística TLBO foi investigada aplicando-o a problemas específicos de mecânica e comparando-o com a Estratégia Evolucionária com Múltiplos Membros (M-ES) (MEZURA-MONTES; COELLO, 2005), a Otimização por Enxame Evolucionário de Partículas (PESO) (ZAVALA, A. E. M.; AGUIRRE; DIHARCE, 2005), a Evolução Diferencial Cultural (CDE) (LANDA BECERRA; COELLO, 2006), a Evolução Diferencial Coevolucionária (CoDE) (HUANG, F.; WANG; HE, 2007) e a Colônia de Abelhas Artificial (ABC) (KARABOGA; BASTURK, 2007).

### 1.6.2 Algoritmo de Campeonato

A metaheurística que fundamenta o algoritmo de campeonato (LCA) foi proposta em (HUSSEINZADEH KASHAN, 2014) e está baseada na emulação de uma liga esportiva organizada por temporadas. As soluções são representadas pelas equipes que competem em uma liga. A liga, neste caso, representa a população de soluções. Uma análise de cada partida é realizada através de aplicação da metodologia de verificação de pontos fortes, fracos, oportunidades e ameaças (SWOT), isto é, para avançar rumo à final, a equipe deve possuir uma boa combinação de atributos (aptidão), o que significa que as equipes vão se adaptando a partir da reformulação de suas soluções até que o algoritmo convirja ou atinja seu critério de parada.

O autor evidenciou em (HUSSEINZADEH KASHAN, 2014) um desempenho considerado satisfatório do LCA se comparado a metaheurísticas bem conhecidas como Algoritmo Genético (GA), Evolução Diferencial (DE), Otimização por Enxame de Partículas (PSO), Colônia de Abelhas Artificiais (ABC) e, mais especificamente, com o Otimizador de Enxame Dinâmico de Partículas com Múltiplos Enxames (DMS-PSO) (LIANG, J. J.; SUGANTHAN, 2005). Em (ABDULHAMID et al., 2015) é desenvolvida

uma investigação abrangente sobre tal metaheurística, apresentando possíveis prospecções e desafios.

## 1.7 Metaheurísticas não Baseadas em Metáforas

As metaheurísticas não baseadas em metáforas, como o nome sugere, representam a categoria de algoritmos cuja designação não corresponde à qualidade ou comportamento de terceiros. Nesta revisão de literatura são apresentados os algoritmos de busca tabu (TS), busca de vizinhança variável (VNS) e otimização metaheurística parcial sob condições de intensificação especiais (POPMUSIC).

### 1.7.1 Busca Tabu

A primeira evidência de utilização do termo metaheurística na bibliografia é em (GLOVER; MCMILLAN, 1986), os quais propuseram um algoritmo denominado busca tabu. A ideia principal é evitar que o algoritmo revisite áreas já pesquisadas no espaço de busca, promovendo a diversificação de soluções e evitando a incidência de ótimos locais (ABDEL-BASSET; ABDEL-FATAH; SANGAIAH, 2018).

Esse algoritmo está amparado pela memória adaptativa e pela exploração responsiva. A memória adaptativa, ou lista tabu, guarda o histórico das etapas realizadas durante o processo de pesquisa, de forma a evitar possível condição cíclica. A exploração responsiva utiliza os parâmetros da lista tabu para concentrar o processo de busca em regiões mais promissoras, tornando mais eficiente o processo de exploração e, consequentemente, sua convergência para um ótimo global.

Uma abordagem mais aprofundada e abrangente desta metaheurística foi investigada e desenvolvida nos trabalhos disponíveis em (GLOVER; TAILLARD; TAILLARD, 1993), (GLOVER, 1995), (ČANGALOVIĆ et al., 1996), (HANAFI, 2001), (MASTROLILLI; GAMBARDELLA, 2005), (GLOVER; LAGUNA; MARTÍ, 2007) e em (PIRIM; BAYRAKTAR; EKSIOGLU, 2008).

### 1.7.2 Busca de Vizinhança Variável

A metaheurística de busca de vizinhança variável (VNS), proposta em (MLADENOVIĆ; HANSEN, 1997), é um algoritmo computacional estocástico derivado do algoritmo de escalada de montanha (BROWNLEE, 2012). Contudo, sua proposta consiste na busca iterativa em bairros com proporções cada vez maiores e com o objetivo de encontrar um local ideal. Sua estratégia está definida nas seguintes premissas:

1) Uma solução ótima local para uma estrutura de bairro pode não ser a mesma para uma estrutura de bairro diferente;

2) Uma solução ótima global é uma solução ótima local para todas as estruturas de bairro possíveis;

3) Soluções ótimas locais são relativamente próximas às soluções ótimas globais para muitas classes de problemas.

A ideia principal é explorar, de forma sistemática e/ou aleatória, vários bairros durante a busca de uma solução ótima (ou próxima da ótima), aplicando, de maneira alternada, as duas abordagens computacionais. Esta metaheurística executa pesquisa local simples para obter as soluções ótimas locais no interior da estrutura do bairro. Em seguida, executa uma busca nos bairros mais afastados da solução ótima local e se muda daquele ponto caso haja alguma melhoria na solução, o que permite manter as variáveis mais adequadas e encontrar os vizinhos mais promissores (ABDEL-BASSET; ABDEL-FATAH; SANGAIAH, 2018).

Nos trabalhos desenvolvidos em (HANSEN; MLADENOVIĆ, 2001), (HANSEN; MLADENOVIĆ, 2002) e (HANSEN; MLADENOVIĆ, 2003), foram abordados os fundamentos teóricos e aplicações práticas concernentes a tal metaheurística.

### 1.7.3 Otimização Metaheurística Parcial sob Condições de Intensificação Especiais

Desenvolvida em (TAILLARD, É. D.; VOSS, 2002), a metaheurística de otimização parcial sob condições de intensificação especiais (POPMUSIC) é baseada na busca em grandes espaços exploratórios, através da divisão de um único problema $S$ em problemas menores ($s_1$, $s_{2,...,}$ $s_n$), que, então, são designados como sementes ou

subproblemas *R*. Estes subproblemas são solucionados através de outra metaheurística, ou um algoritmo exato, definido pelo usuário. O processo de iteração ocorre até que todos os subproblemas sejam otimizados.

Pelo fato supracitado, o POPMUSIC foi concebido como uma estrutura metaheurística que comporta outros procedimentos de exploração e aplicações, assim como foi feito em: (SHAW, 1998), ao utilizar a busca por grandes vizinhanças; em (TAILLARD, E. D., 1998), ao utilizar as otimizações locais (LOPT); e (BENT; VAN HENTENRYCK, 2010), ao utilizar a decomposição aleatória adaptativa.

O desempenho do POPMUSIC foi posto à prova por seus autores para resolver os problemas do agrupamento de centroides e do balanceamento de peças mecânicas, em que evidenciaram desempenho satisfatório para o primeiro problema (se comparado ao VNS) e para o segundo problema, configurado com apoio da metaheurística TS (se comparado à TS pura).

## 1.8 Evolução de Metaheurísticas

Como já mencionado, uma metaheurística é, basicamente, um processo matemático iterativo que obtém uma nova solução a partir de uma solução anterior. Além disso, é estruturado como um vetor de *k* parâmetros e *m* números aleatórios que promovem uma busca estocástica. Contudo, é possível melhorar o desempenho destas ferramentas de maneira que se transformem ao longo do processamento de suas operações, através de um processo de adaptação natural, no sentido de diminuir o tempo de convergência e melhorar a qualidade das soluções. Nos itens a seguir estão relacionados alguns dos principais exemplos, seguindo as referências indicadas em (ABDEL-BASSET; ABDEL-FATAH; SANGAIAH, 2018).

### 1.8.1 Metaheurísticas Adaptativas

Metaheurísticas adaptativas são aquelas que dispõem de atributos capazes de, automaticamente, ajustar o espaço de busca ou o processo de iteração para prevenir a convergência prematura. Pode-se citar, como exemplos, o Sistema de Formiga *MAX-MIN*, como descrito em (STÜTZLE; HOOS, 2000), o algoritmo de Otimização por Enxame de Partículas Adaptativo (CAI; PAN; CHEN, 2004), a Busca Harmônica

Adaptativa (MAHDAVI; FESANGHARY; DAMANGIR, 2007), a Busca por Cuckoo Modificada (WALTON et al., 2011), o Recozimento Simulado Adaptativo (JUNIOR et al., 2012), o algoritmo de Vagalume Adaptativo (CHEUNG; DING; SHEN, 2014), o algoritmo de Otimização por Baleia (TRIVEDI et al., 2016), o algoritmo de Lobo Branco Melhorado (KUMAR; KUMAR; CHHABRA, 2016), entre outros.

### 1.8.2 Metaheurísticas Caóticas

Metaheurísticas baseadas no caos são ferramentas computacionais de interesse em aplicações onde, geralmente, necessita-se de alta aleatoriedade e taxa de convergência. Este tipo de condição é alcançado ao aplicar os denominados mapas de caos (mapa logístico, mapa de Chebyshev, mapa de tendas, entre outros), que representam funções evolutivas que fornecem uma sequência deterministicamente limitada de números aleatórios com base na condição inicial com diferente domínio do tempo, isto é, contínuo ou discreto, podendo, inclusive, serem aplicados como substitutos aos geradores de sequências aleatórias nas metaheurísticas de interesse (ABDEL-BASSET; ABDEL-FATAH; SANGAIAH, 2018).

Pode-se citar como exemplos de metaheurísticas baseadas no caos aquelas desenvolvidas em: (HEFNY; AZAB, 2010), ao utilizar a Otimização de Enxame de Partículas Caóticas; (ALATAS, 2010), ao utilizar a Busca Harmônica Caótica; (SNASELOVA; ZBORIL, 2015), ao investigar o Algoritmo Genético Caótico; (BINGOL; ALATAS, 2016), referente ao Algoritmo de Campeonato de Liga Caótica; dentre outras adaptações.

### 1.8.3 Metaheurísticas Baseadas em Gaussiano

Como a designação sugere, as metaheurísticas assim classificadas são aquelas que utilizam a distribuição gaussiana para realizar uma espécie de ajuste fino durante seu processamento, podendo, inclusive, substituir a atualização regular de uma solução pela amostragem de tal tipo de distribuição. Exemplos de aplicação desta metodologia foram desenvolvidos utilizando algoritmo de Enxame de Partículas tipo Esqueleto (KENNEDY, 2003), algoritmo de Vagalume Gaussiano (FARAHANI et al., 2011), Evolução Diferencial de Esqueleto Gaussiano (WANG, H. et al., 2013), Otimização Baseada em Ensino-Aprendizagem de Esqueleto (ZOU et al., 2014), entre outros.

### 1.8.4 Aceleração de Metaheurísticas

As metaheurísticas com esta classificação possuem alterações com o objetivo de acelerar a convergência do processamento através da omissão de parâmetros menos relevantes ou adicionando-os, caso seja adequado. Como exemplos, cita-se: o PSO Acelerado (YANG, X.-S.; DEB; FONG, 2011), que consiste, basicamente, na eliminação da solução local ótima (*lbest*), de forma que a nova solução dependerá, exclusivamente, da solução global ótima (*gbest*); a Colônia de Abelhas Artificial Acelerada (OZKIS; BABALIK, 2013); o algoritmo de Explosão de Minas Acelerado (SALLEH; HUSSAIN, 2016); e a Otimização Baseada em Biogeografia Acelerada (LOHOKARE et al., 2010).

## 1.9 Aclimatação de Metaheurísticas

Aclimatação é uma designação geral utilizada com o intuito de descrever o processo de um organismo ajustar-se às mudanças que ocorrem em seu habitat. Neste sentido, o termo é utilizado para se referir às metaheurísticas que são adaptadas para processar problemas mais complexos, com diferentes concepções ou domínios matemáticos. Os itens a seguir apresentam as metaheurísticas multiobjetivo, discretizadas e híbridas.

### 1.9.1 Metaheurísticas Multiobjetivo

Para uma variedade abrangente de problemas de otimização, as metaheurísticas mono objetivo não são capazes de fornecer uma análise unificada, pois não consideram a possibilidade de haverem múltiplos objetivos conflitantes. Através das metaheurísticas multiobjetivo (ou “otimização vetorial” ou “otimização multicritério” ou “otimização de Pareto”), utiliza-se o conceito de “fronteira de Pareto”, que estabelece um conjunto de boas soluções, visando os objetivos de interesse do usuário (JONES, D. F.; MIRRAZAVI; TAMIZ, 2002), (GANDIBLEUX et al., 2004), (ZAVALA, G. R. et al., 2014), (ABDEL-BASSET; ABDEL-FATAH; SANGAIAH, 2018).

Exemplos de metaheurísticas multiobjetivo de relevância na literatura são o Algoritmo Genético de Sorteio Não Dominado (NSGA) (SRINIVAS; DEB, 1994) e o NSGA-II (DEB et al., 2002), os quais utilizam o conceito de fronteira de Pareto para lidar com otimizações de múltiplos critérios. O NSGA-II é uma evolução de seu antecessor,

com maior velocidade de convergência e melhor atributo de aleatoriedade (ABDEL-BASSET; ABDEL-FATAH; SANGAIAH, 2018).

Outra metaheurística de mesma aplicabilidade é o algoritmo de Otimização por Enxame de Partículas Multiobjetivo (MOPSO), desenvolvido em (MOORE; CHAPMAN, 1999). De acordo com (REYES-SIERRA; COELLO, 2006), há duas variações principais desta metaheurística. Na primeira, cada função objetivo é considerada individualmente, o que dificulta processar as informações de cada função, no sentido de guiar as partículas (soluções) em direção à fronteira de Pareto. Na segunda, ocorre a análise de cada partícula para todas as funções objetivo (baseado na fronteira de Pareto), o que possibilita guiar as partículas para as melhores posições não dominadas por um objetivo. Já em (RAQUEL; NAVAL JR, 2005), foi introduzido um mecanismo de "distanciamento de aglomeração" com o intuito de manter a diversidade de soluções não dominadas no repositório externo.

### 1.9.2 Discretização de Metaheurísticas

Muitos problemas de otimização não podem ser processados através de metaheurísticas que trabalham no domínio contínuo, o que levou ao desenvolvimento de métodos de discretização para lidar com problemas que combinam valores binários e inteiros, como o Problema do Caixeiro Viajante, o Problema de Localização da Instalação, o Problema da Mochila, entre outros (ABDEL-BASSET; ABDEL-FATAH; SANGAIAH, 2018).

A Função Sigmoide (SF) é uma das formas utilizadas para transformar um espaço de busca contínuo em um espaço binário. Alguns exemplos de metaheurísticas que utilizaram tal método são a Otimização por Enxame de Partículas Binário (BPSO) (KHANESAR; TESHNEHLAB; SHOOREHDELI, 2007), o algoritmo de Vagalume Discreto (SAYADI; RAMEZANIAN; GHAFFARI-NASAB, 2010), a Otimização por Enxame de Gatos Binária (BCSO) (SHARAFI; KHANESAR; TESHNEHLAB, 2013), o algoritmo de Buracos Negros Binários (NEMATI; MOMENI; BAZRKAR, 2013), o algoritmo de Polinização de Flores Binário (BFPA) (RODRIGUES et al., 2015), entre outros.

Outro método de relevância é a Chave-Aleatória (RK), que utiliza um vetor de números aleatórios e atribui um peso a cada elemento com o objetivo de executar uma

permutação como solução. Pode-se citar como exemplos de metaheurísticas nesta linha o Algoritmo Genético com Chave-Aleatória (BEAN, 1994), o Algoritmo de Salto de Sapo Embaralhado Aprimorado (CHEN, M.-R. et al., 2011), o Algoritmo de Bola Dourada de Chaves-Aleatórias (SAYOTI; RIFFI, 2015), a Pesquisa de Cuckoo de Chaves-Aleatórias (OUAARAB; AHIOD; YANG, 2015), entre outros.

Um último método de discretização relevante na literatura é o Menor Valor de Posição (SPV), utilizado com frequência em problemas envolvendo a programação de tarefas. As metaheurísticas na literatura que utilizam este método são o PSO (ZHANG, L. et al., 2008), o Algoritmo de Otimização de Agitação (SOA) (ABDELHAFIEZ; ALTURKI, 2011), o Algoritmo de Vagalume Discreto (MARICHELVAM; PRABAHARAN; YANG, 2014), o Algoritmo de Pesquisa de Gravitação Memética (MGSA) (HUANG, K. W.; CHEN; YANG, 2015), entre outros. Em (KRAUSE et al., 2013) é disponibilizada uma discussão mais rica em detalhes sobre o método de discretização.

### 1.9.3 Continuação de Metaheurísticas

A continuação de metaheurísticas representa a categoria de metaheurísticas do domínio discreto adaptadas para aplicação no domínio contínuo. Uma abordagem já mencionada é através da discretização. Já a outra é estimar os parâmetros de distribuição e, neste sentido, uma metaheurística relevante na literatura é o Algoritmo Genético com Código Real (RCGA) (WRIGHT, 1991). Nesta metaheurística, uma solução candidata (cromossomo) é designada como um vetor de números de ponto flutuante, ao invés de bits binários (perda de precisão). Ainda em (LAM; LI; YU, 2012), foi proposto um algoritmo CRO com Código Real (RCCRO) que utiliza a distribuição gaussiana para produzir perturbações para explorar no domínio contínuo.

## 1.10 Hibridização de Metaheurísticas

A hibridização de metaheurísticas consiste em aliar os pontos fortes de metaheurísticas diferentes em apenas uma. Tal objetivo pode ser alcançado utilizando técnicas de nível alto, isto é, ocorre baixa interferência entre o trabalho interno dos algoritmos hibridizados, ou nível baixo, que designa uma função metaheurística

exportada para outra metaheurística. O processamento destas metaheurísticas híbridas pode ser realizado de forma colaborativa ou integrada (ABDEL-BASSET; ABDEL-FATAH; SANGAIAH, 2018).

Além disso, os autores especificam que a hibridização pode ser categorizada, baseada em diferentes atributos conforme a seguir:

- Modo de hibridização (e.g. metaheurística, aprendizado de máquina, métodos exatos, entre outros);
- Ordem de execução das metaheurísticas (sequencial, intervalada ou paralela);
- Estratégia de controle (colaborativa ou integrativa);
- Nível de hibridização (alto ou baixo).

### 1.10.1 Hibridização Sequencial e Intervalada

A metaheurística híbrida ANGEL (*ANt colony*, *GEnetic algorithm*, *Local search method*) é uma combinação das três metaheurísticas supracitadas (TSENG; LIANG, 2006). A população inicial é gerada através do ACO e, em seguida, processada através do GA. O método de busca local (LS) é utilizado como mecanismo de suporte no processo exploratório.

Outro exemplo de metaheurística híbrida é investigada em (BINU; SELVI; GEORGE, 2013), a partir da combinação do algoritmo de Pesquisa Cuckoo (CS) (YANG, X.-S.; DEB, 2009), com o algoritmo MKF-Cuckoo e (CHEN, L.; CHEN; LU, 2011), concebendo o *Multiple Kernel-based Fuzzy C Means* (MKFCM). Nessa metaheurística, as possíveis soluções são codificadas através da escolha de um centroide aleatório, retirado do conjunto de dados de entrada. A função *fitness* aplica a abordagem *Fuzzy* para encontrar a distância mínima entre os pontos de dados com o centroide vizinho mais próximo. Após isso, os procedimentos da metaheurística CS são executados.

Um último caso relevante na literatura é a metaheurística híbrida desenvolvida em (NABIL, 2016), em que os atributos de exploração do algoritmo de Polinização de Flores (baseado na polinização local) foram mesclados com o algoritmo de Seleção Clonal (CSA). Em (SAYED; NABIL; BADR, 2016) foi proposta uma versão binária para a mesma metaheurística.

### 1.10.2 Metaheurísticas em Esquemas Paralelos

Com o aumento da complexidade de alguns problemas de otimização mais atuais, a abordagem metaheurística sequencial já não atende de forma plena, o que tem norteado pesquisas recentes rumo ao processamento paralelo, abordado em (CRAINIC; TOULOUSE, 1998). Aplicações nesse sentido foram, também, desenvolvidas em (LUQUE; ALBA, 2011), com o emprego de Algoritmos Genéticos Paralelos, em (NARASIMHAN, 2009), através da Colônia de Abelhas Artificial Paralela, em (RIBEIRO; ROSSETI, 2007), com o Procedimento de Pesquisa Adaptativa Aleatória Gulosa (GRASP) paralela, entre outros. Já em (ALBA, 2005) e (ALBA; LUQUE; NESMACHNOW, 2013), os autores apresentam relevante fundamentação teórica e perspectivas sobre o paralelismo em problemas de otimização com o emprego de metaheurísticas.

Finalmente, pode-se implementar tal abordagem a partir de modelos baseados em trajetória, isto é, atualizando uma solução por outra a partir de seus vizinhos, ou por modelos baseados em população, que consiste na atualização de toda a população a cada iteração, o que reflete na utilização de mais processadores a fim de reduzir o tempo de convergência (ABDEL-BASSET; ABDEL-FATAH; SANGAIAH, 2018).

# APÊNDICE B – Referencial Bibliográfico Sobre Otimização de Hiperparâmetros de CNN

Tabela 13 – Investigação bibliométrica - ajuste de CNNs e otimização bayesiana (continua).

| Autor(es) | Artigo |
|---|---|
| (BASHA; VINAKOTA; DUBEY; et al., 2021). | AutoFCL: Automatically Tuning Fully Connected Layers for Handling Small Dataset |
| (OBAYYA; EL-GHANDOUR; ALROWAIS, 2021) | Contactless Palm Vein Authentication Using Deep Learning with Bayesian Optimization |
| (BASHA; VINAKOTA; PULABAIGARI; et al., 2021) | AutoTune: Automatically Tuning Convolutional Neural Networks for Improved Transfer Learning |
| (WANG, C. et al., 2020) | Research on Bearing Fault Diagnosis Method Based on an Adaptive Anti-Noise Network under Long Time Series |
| (HIZUKURI et al., 2020) | Computer-Aided Diagnosis Scheme for Distinguishing Between Benign and Malignant Masses on Breast DCE-MRI Images Using Deep Convolutional Neural Network with Bayesian Optimization |
| (HU; HUANG; YU, 2020) | Efficient Mapping of Crash Risk at Intersections with Connected Vehicle Data and Deep Learning Models |
| (SHI et al., 2020) | Designing a Lightweight 1D Convolutional Neural Network with Bayesian Optimization for Wheel Flat Detection Using Carbody Accelerations |
| (EKICI et al., 2020) | Power Quality Event Classification Using Optimized Bayesian Convolutional Neural Networks |
| (FERREIRA et al., 2020) | Galaxy Merger Rates up to z similar to 3 Using a Bayesian Deep Learning Model: A Major-merger Classifier Using IllustrisTNG Simulation Data |
| (LI, Dan et al., 2020) | Acoustic Emission Wave Classification for Rail Crack Monitoring Based on Synchrosqueezed Wavelet Transform and Multi-branch Convolutional Neural Network |
| (DOKE et al., 2020) | Using CNN with Bayesian Optimization to Identify Cerebral Micro-Bleeds |
| (KOJIMA et al., 2020) | kGCN: A Graph-Based Deep Learning Framework for Chemical Structures |
| (LU et al., 2020) | Bayesian Optimized Deep Convolutional Network for Bearing Diagnosis |
| (ZHAN; ZHAO, 2020) | Classification of Carbon Fiber Reinforced Polymer Defects Based on One-Dimensional CNN |

Fonte: (Web of Science, 2021)

Tabela 13 – Investigação bibliométrica - ajuste de CNNs e otimização bayesiana (continuação).

| Autor(es) | Artigo |
|---|---|
| (NOMURA et al., 2020) | Pilot Study of Eruption Forecasting with Muography Using Convolutional Neural Network |
| (SAMEEN; PRADHAN; LEE, 2020) | Application of Convolutional Neural Networks Featuring Bayesian Optimization for Landslide Susceptibility Assessment |
| (WALDMANN; PFEIFFER; MÉSZÁROS, 2020) | Sparse Convolutional Neural Networks for Genome-Wide Prediction |
| (YOON, S.-J. et al., 2020) | Automatic Multi-Class Intertrochanteric Femur Fracture Detection from CT Images Based on AO/OTA Classification Using Faster R-CNN-BO Method |
| (YANG, H.; JIAO; SUN, 2020) | Bayesian-Convolutional Neural Network Model Transfer Learning for Image Detection of Concrete Water-Binder Ratio |
| (DUAN; ZHANG, 2020) | Two-Stream Convolutional Neural Network Based on Gradient Image for Aluminum Profile Surface Defects Classification and Recognition |
| (DU et al., 2020) | Identification of COPD From Multi-View Snapshots of 3D Lung Airway Tree via Deep CNN |
| (LI, J.; HE, 2020) | A Bayesian Optimization AdaBN-DCNN Method with Self-Optimized Structure and Hyperparameters for Domain Adaptation Remaining Useful Life Prediction |
| (KIM; LEE; CHOE, 2020) | Bayesian Optimization-Based Global Optimal Rank Selection for Compression of Convolutional Neural Networks |
| (MONTEIRO; SKOURUP; ZHANG, 2020) | Optimizing CNN Hyperparameters for Mental Fatigue Assessment in Demanding Maritime Operations |
| (LI, Y. et al., 2019) | Self-Learning Perfect Optical Chirality via a Deep Neural Network |
| (NADDAF-SH et al., 2019) | Real-Time Road Crack Mapping Using an Optimized Convolutional Neural Network |
| (LU et al., 2019) | Bayesian Optimized Deep Convolutional Network for Electrochemical Drilling Process |
| (CUI; BAI, 2019) | A New Hyperparameters Optimization Method for Convolutional Neural Networks |
| (LIANG, X., 2019) | Image-based Post-disaster Inspection of Reinforced Concrete Bridge Systems Using Deep Learning with Bayesian Optimization |
| (ELOLA et al., 2019) | Deep Neural Networks for ECG-Based Pulse Detection during Out-of-Hospital Cardiac Arrest |

Fonte: (Web of Science, 2021)

Tabela 13 – Investigação bibliométrica - ajuste de CNNs e otimização bayesiana (continuação).

| Autor(es) | Artigo |
|---|---|
| (ITAKURA et al., 2019) | Estimation of Citrus Maturity with Fluorescence Spectroscopy Using Deep Learning |
| (OLIVAS-PADILLA; CHACON-MURGUIA, 2019) | Classification of Multiple Motor Imagery Using Deep Convolutional Neural Networks and Spatial Filters |
| (KIRCHGÄSSNER; WALLSCHEID; BÖCKER, 2019) | Deep Residual Convolutional and Recurrent Neural Networks for Temperature Estimation in Permanent Magnet Synchronous Motors |
| (AHMED; VIRIRI, 2019) | Deep Learning Using Bayesian Optimization for Facial Age Estimation |
| (STEIN; WANG; BÄCK, 2019) | Automatic Configuration of Deep Neural Networks with Parallel Efficient Global Optimization |
| (KALOURIS; ZACHARAKI; MEGALOOIKONOMOU, 2019) | Improving CNN-based activity recognition by data augmentation and transfer learning |
| (MA; CUI; YANG, 2019) | Deep Neural Architecture Search with Deep Graph Bayesian Optimization |
| (YOON, H.-J. et al., 2019) | Model-based Hyperparameter Optimization of Convolutional Neural Networks for Information Extraction from Cancer Pathology Reports on HPC |
| (ZHANG, Z. et al., 2019) | Convolutional Neural Network Using Bayesian Optimization for Laser Welding Tailor Rolled Blanks Penetration Detection |
| (JIN; LI; JUNG, 2019) | Prediction of Flow Properties on Turbine Vane Airfoil Surface from 3D Geometry with Convolutional Neural Network |
| (WEI et al., 2019) | Automatic Modulation Recognition Using Neural Architecture Search |
| (BORGLI et al., 2019) | Automatic Hyperparameter Optimization for Transfer Learning on Medical Image Datasets Using Bayesian Optimization |
| (SALAMA; HASSANIEN; FAHMY, 2019) | Sheep Identification Using a Hybrid Deep Learning and Bayesian Optimization Approach |
| (HAN; LIU; FAN, 2018) | A new image classification method using CNN transfer learning and web data augmentation |
| (STAMOULIS et al., 2018) | Designing Adaptive Neural Networks for Energy-Constrained Image Classification |
| (MUHAMMAD; MOUSTAFA, 2018) | Improving Region Based CNN Object Detector Using Bayesian Optimization |

Fonte: (Web of Science, 2021)

Tabela 13 – Investigação bibliométrica - ajuste de CNNs e otimização bayesiana (conclusão).

| Autor(es) | Artigo |
|---|---|
| (OSMANI; HAMIDI, 2018) | Hybrid and Convolutional Neural Networks for Locomotion Recognition |
| (LE et al., 2018) | Human Motion Classification with Micro-Doppler Radar and Bayesian-Optimized Convolutional Neural Networks |
| (SAHUMBAIEV et al., 2018) | 3D-CNN HadNet classification of MRI for Alzheimer's Disease diagnosis |
| (RHODES et al., 2017) | Bayesian Optimization for Refining Object Proposals |
| (SHAHRIARI, Bobak; BOUCHARD-COTE; FREITAS, 2016) | Unbounded Bayesian Optimization via Regularization |
| (ZHANG, Y. et al., 2015) | Improving Object Detection with Deep Convolutional Networks via Bayesian Optimization and Structured Prediction |

Fonte: (Web of Science, 2021)

# APÊNDICE C – Histórico de Exploração do Espaço de Busca: *Dataset* $A_0$

Tabela 14 – Histórico de otimização – 2º estágio $A_0$ (continua).

| ***Fitness*** | **num_conv_layers** | **num_conv_nodes** | **num_dense_layers** | **num_dense_nodes** |
|---|---|---|---|---|
| 0,00625169 | 3 | 72 | 3 | 256 |
| 0,00643662 | 4 | 87 | 3 | 256 |
| 0,00655110 | 3 | 39 | 2 | 179 |
| 0,00702742 | 3 | 86 | 2 | 231 |
| 0,00744587 | 3 | 91 | 3 | 119 |
| 0,00788706 | 3 | 54 | 3 | 64 |
| 0,00793519 | 4 | 63 | 2 | 183 |
| 0,00807077 | 4 | 64 | 1 | 158 |
| 0,00810674 | 3 | 51 | 3 | 84 |
| 0,00851307 | 3 | 54 | 2 | 150 |
| 0,00870725 | 3 | 114 | 3 | 117 |
| 0,00878933 | 3 | 121 | 2 | 126 |
| 0,00956096 | 3 | 124 | 3 | 131 |
| 0,01010462 | 3 | 43 | 3 | 150 |
| 0,01016025 | 4 | 256 | 1 | 125 |
| 0,01064342 | 3 | 43 | 3 | 203 |
| 0,01067513 | 4 | 88 | 1 | 230 |
| 0,01072388 | 3 | 47 | 2 | 159 |
| 0,01087322 | 4 | 51 | 2 | 102 |
| 0,01107905 | 4 | 256 | 2 | 107 |
| 0,01156561 | 3 | 37 | 1 | 60 |
| 0,01176282 | 3 | 99 | 3 | 135 |
| 0,01203213 | 3 | 78 | 2 | 75 |
| 0,01206908 | 4 | 81 | 2 | 256 |
| 0,01224088 | 3 | 35 | 3 | 71 |
| 0,01310342 | 3 | 62 | 2 | 256 |
| 0,01368345 | 4 | 33 | 2 | 152 |
| 0,01373097 | 4 | 112 | 3 | 151 |
| 0,01384244 | 4 | 134 | 2 | 123 |
| 0,01395089 | 4 | 103 | 3 | 175 |
| 0,01420088 | 3 | 44 | 3 | 148 |
| 0,01533459 | 3 | 41 | 2 | 231 |
| 0,01596271 | 4 | 92 | 2 | 256 |

Fonte: (Autor, 2021)