UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

ANA PAULA CAPELETTI RAMOS ALMEIDA

AVALIAÇÃO DE MÉTODOS DE OTIMIZAÇÃO EM REDES NEURAIS CONVOLUCIONAIS PARA IDENTIFICAÇÃO DE DÍGITOS

SANTA HELENA, PARANÁ

2023

ANA PAULA CAPELETTI RAMOS ALMEIDA

AVALIAÇÃO DE MÉTODOS DE OTIMIZAÇÃO EM REDES NEURAIS CONVOLUCIONAIS PARA IDENTIFICAÇÃO DE DÍGITOS

EVALUATION OF OPTIMIZATION METHODS IN CONVOLUTIONAL NEURAL NETWORKS FOR DIGITAL IDENTIFICATION

Trabalho de Conclusão de Curso de graduação apresentado como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação, Universidade Tecnológica Federal do Paraná (UTFPR).
Orientador: Prof. Me. Evandro Alves Nakajima
Coorientador: Prof. Dr. Giuvane Conti

SANTA HELENA, PARANÁ

2023

4.0 Internacional

Esta licença permite remixe, adaptação e criação a partir do trabalho, para fins não comerciais, desde que sejam atribuídos créditos ao(s) autor(es). Conteúdos elaborados por terceiros, citados e referenciados nesta obra não são cobertos pela licença.

ANA PAULA CAPELETTI RAMOS ALMEIDA

AVALIAÇÃO DE MÉTODOS DE OTIMIZAÇÃO EM REDES NEURAIS CONVOLUCIONAIS PARA IDENTIFICAÇÃO DE DÍGITOS

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Ciência da Computação, Universidade Tecnológica Federal do Paraná (UTFPR).

Data de aprovação: 01 de dezembro de 2023

Davi Marcondes Rocha
Doutor em Engenharia Agrícola
Link para o currículo Lattes: http://lattes.cnpq.br/2423987011078680
Universidade Tecnológica Federal do Paraná

Evandro Alves Nakajima
Doutor em Engenharia Química
Link para o currículo Lattes: http://lattes.cnpq.br/1048416043189843
Universidade Tecnológica Federal do Paraná

Suzan Kelly Borges Piovesan
Doutorado em andamento em Engenharia Agrícola
Link para o currículo Lattes: http://lattes.cnpq.br/4373225816434776
Universidade Tecnológica Federal do Paraná

SANTA HELENA, PARANÁ

2023

RESUMO

Este trabalho propõe a criação de um *software* voltado para o reconhecimento de palavras em imagens, através de uma rede neural, realizando a análise das discrepâncias entre uma rede treinada por métodos clássicos, como o gradiente descendente, e abordagens probabilísticas. O objetivo central é fornecer suporte à comunidade acadêmica e aos interessados em redes neurais, identificação de caracteres, gradiente descendente e métodos probabilísticos. O foco concentra-se no desenvolvimento de um *software* de identificação de caracteres utilizando Redes Neurais Artificiais (RNA) com distintos métodos de *backpropagation*. A metodologia empregada envolveu o uso do Eclipse 2023-06 e do *Scene Builder* para a criação da interface gráfica, com Java na versão 8 e JavaFX como base de programação. O *software* foi concebido com duas vertentes principais: uma voltada aos usuários (“tela usuário” e “ajuda”) e outra destinada a testes internos (“tela dev” e “treina CNN”). Durante os testes realizados, a execução do algoritmo empregando o método do gradiente descendente apresentou resultados semelhantes aos obtidos com o *Particle Swarm Optimization* (PSO), com este último demonstrando um desempenho ligeiramente superior, alcançando uma acurácia de 72.5%. No entanto, a limitação da máquina de testes, sem a adição de uma GPU para processamento, inviabilizou a classificação de letras, resultando na classificação de dígitos manuscritos. É relevante destacar que esses testes foram conduzidos em conjuntos de imagens de pequena escala, revelando limitações do PSO quando aplicado a conjuntos de imagens mais extensos. Uma possível solução para lidar com essa questão seria a melhoria do hardware da máquina de testes, por meio da adição de uma GPU para processamento.

Palavras-chave: Gradiente Descendente; Scene Builder; JavaFX; Redes Neurais Artificiais; RNA; *Backpropagation*; *Particle Swarm Optimization*; PSO.

ABSTRACT

This work proposes the creation of software aimed at recognizing words in images, through a neural network, analyzing discrepancies between a network trained by classical methods, such as gradient descent, and probabilistic approaches. The central objective is to provide support to the academic community and those interested in neural networks, character identification, gradient descent and probabilistic methods. The focus is on the development of character identification software using Artificial Neural Networks (ANN) with different backpropagation methods. The methodology used involved the use of Eclipse 2023-06 and Scene Builder to create the graphical interface, with Java version 8 and JavaFX as the programming base. The software was designed with two main aspects: one aimed at users ("user screen" and "help") and the other intended for internal testing ("dev screen" and "train CNN"). During the tests carried out, the execution of the algorithm using the gradient descent method presented results similar to those obtained with Particle Swarm Optimization (PSO), with the latter demonstrating slightly better performance, reaching an accuracy of 72.5%. However, the limitation of the testing machine, without the addition of a GPU for processing, made the classification of letters unfeasible, resulting in the classification of handwritten digits. It is important to highlight that these tests were conducted on small-scale image sets, revealing limitations of PSO when applied to larger image sets. A possible solution to deal with this issue would be to improve the hardware of the testing machine, through the addition of a GPU for processing.

Keywords: Gradient Descent; Scene Builder; JavaFX; Artificial Neural Network; ANN;

Backpropagation; Particle Swarm Optimization; PSO.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

CNN: *Convolutional Neural Network* (Redes Neurais Convolucionais)

EDM: *Euclidean Distance Map* (Mapa de Distâncias Euclidianas)

PSO: *Particle Swarm Optimization* (Otimização por Enxame de Partículas)

RNA: Redes Neurais Artificiais

RELU: *Rectified Linear Unit* (Unidade Linear Retificada)

LISTA DE SÍMBOLOS

w: Peso relacionado ao neurônio

X_i^k: Posição da partícula i na iteração k

v_i^k: Velocidade da partícula i na iteração k

p_i: Melhor posição da partícula i

g: Melhor posição entre todas as partículas

c_i: Constante de efeito local/global

K: Dimensão do pooling

SUMÁRIO

1 INTRODUÇÃO

Uma das áreas de pesquisa com uma ampla visibilidade na atualidade é a simulação de capacidades cognitivas de um ser humano, a qual tem por objetivo projetar máquinas capazes de exibir um comportamento inteligente. No cérebro o neurônio é o responsável pela resposta a estímulos do meio externo ao corpo, como a luz e o calor. Um interesse que surge desses fatos é a tentativa de reproduzir o funcionamento da inteligência do cérebro em um meio técnico. Utilizar uma estrutura artificial para mapear as sinapses realizadas pelos neurônios, assim transformando as redes neurais biológicas em artificiais.

O objetivo deste trabalho foi desenvolver um sistema para reconhecer dígitos manuscritos em imagens, através de uma rede neural. Avaliando também o método PSO e o gradiente descendente no desenvolvimento da rede neural utilizada. Dentre os métodos utilizados na otimização dos parâmetros presentes na rede neural pode-se destacar o gradiente descendente, que é um método eficaz, que possui suas restrições, como necessitar que a função a ser otimizada seja diferenciável.

Nesse sentido, a otimização baseada em algoritmos probabilísticos possui algumas vantagens, dentre as quais destacam-se: não requerem que a função objetivo seja contínua ou diferenciável; trabalham adequadamente tanto com parâmetros contínuos quanto com discretos; não necessitam de formulações complexas ou reformulações para o problema; realizam buscas simultâneas no espaço de possíveis soluções através de uma população de indivíduos; otimizam um grande número de variáveis, desde que a avaliação da função objetivo não tenha um custo computacional demasiadamente alto.

1.1 OBJETIVOS

Expõem-se a seguir os objetivos geral e específicos que se pretende atingir com o trabalho.

1.1.1 Geral

Desenvolver um *software* para identificação de caracteres utilizando Redes Neurais Artificiais (RNA) com diferentes métodos de *backpropagation*.

1.1.2 Específicos

1) Criar uma base de dados contendo diversas imagens das letras do alfabeto, utilizando diferentes fontes e tratamentos de imagens;
2) Construir uma rede neural e integrá-la com o sistema automatizado que foi desenvolvido;
3) Treinar a rede neural criada utilizando a base de imagens produzida para este projeto;
4) Construir um classificador capaz de identificar as letras presentes no texto da imagem;
5) Construir um *software* para os testes internos, no qual eram mostradas as características principais encontradas pela RNA para a classificação dos caracteres presentes nas imagens;
6) Avaliar por meio de testes estatísticos o classificador e os resultados obtidos;
7) Construir um *software* para os usuários, no qual os utilizadores podiam selecionar uma imagem para transformá-la em texto.

1.2 CONTRIBUIÇÕES DO TRABALHO

A contribuição efetiva deste trabalho está no desenvolvimento de um *software* para identificação de caracteres, utilizando métodos não tradicionais de Inteligência Artificial,

como o *Particle Swarm Optimization* (PSO), para verificar se esses métodos trazem benefícios no treinamento das Redes Neurais Convolucionais (CNN[1]).

1.3 JUSTIFICATIVA

A inteligência artificial está cada vez mais presente no dia a dia das pessoas, desde assistentes virtuais, serviços digitais, mídias sociais, bem como recursos disponíveis em dispositivos inteligentes que se integram de maneira natural ao cotidiano das pessoas e são cada vez mais utilizados não apenas para resolver problemas complexos, mas também para ajudar nas trivialidades diárias (SGARBOSA e VECHIO, 2020). Uma aplicação prática possível é a utilização da rede neural desenvolvida neste trabalho para tradução simultânea entre idiomas, dentre eles a língua de sinais.

Assim, há vários tipos de redes neurais na atualidade, com formas diferentes de implementação, sejam elas tradicionais ou não. Então, surge a ideia de verificar se os métodos não tradicionais como o *Particle Swarm Optimization* (PSO) podem trazer benefício no treinamento das redes neurais convolucionais. Com isso, a finalidade deste trabalho é criar um *software* para estudar as diferenças entre uma rede treinada com métodos clássicos como o gradiente descendente e os métodos probabilísticos. O projeto visa auxiliar a área acadêmica e os demais interessados nas áreas de: redes neurais, identificação de caracteres, gradiente descendente e métodos probabilísticos.

1.4 DELIMITAÇÕES DO TRABALHO

Existem métodos diferentes para identificação de caracteres como o *toolbox* do *Matlab Neural Network* (TEIXEIRA *et al.*, 2009), Letreiramentos da Liberdade (ÁVILA e FARIAS, 2021) e *OpenTLD* (CARVALHO, 2013). O foco deste trabalho é identificação de caracteres por meio de redes neurais treinadas com métodos clássicos como o gradiente descendente e o

1 *Convolutional Neural Network*

método probabilístico PSO, que pode trazer uma melhora nos resultados ao serem comparados com os métodos tradicionais.

2 REFERENCIAL TEÓRICO

Este capítulo apresenta literaturas que sustentam a base teórica necessária para cumprir os objetivos deste projeto, apresentando as técnicas e tecnologias que se pretende utilizar para o desenvolvimento da solução computacional para o problema proposto.

2.1 VISÃO COMPUTACIONAL

A visão computacional é a ciência responsável pela extração de informações significativas a partir de imagens capturadas por câmeras de vídeo, *scanners*, sensores, entre outros dispositivos (MILANO e HONORATO, 2010), ou seja, é a ciência das máquinas que "enxergam" (WIZBICKI e BATTISTI, 2014). Utilizando um conjunto de técnicas da visão computacional é possível permitir que o computador tenha uma visão similar a visão humana, possibilitando analisar imagens de forma mais eficaz (tanto pela agilidade quando pela precisão) do que o ser humano seria capaz (GODOI, 2019).

As máquinas são capazes de realizar apenas o que foram projetadas para fazer e, dessa forma, erros podem ocorrer com pequenas alterações, como o exemplo apresentado no estudo realizado por Elsayed *et al.* (2018, p.1): "*small changes to images can cause computer vision models to make mistakes such as identifying a school bus as an ostrich*".

Os primeiros registros de trabalhos utilizando visão computacional, datados no início da década de 1970, foram vistos como componentes de percepção visuais ambiciosos e tinham como objetivo imitar a inteligência humana e dotar os robôs com comportamento inteligente (SZELISKI, 2010).

O primeiro conteúdo a respeito de visão computacional presente na base de dados do *Web of Science* é datado de 1975 e intitulado "*Analyzing Natural Images – Computational*

Theory of Texture Vision", do autor Marr David, contendo referências sobre visão computacional, processamento visual e visão de textura (MARR, 1975).

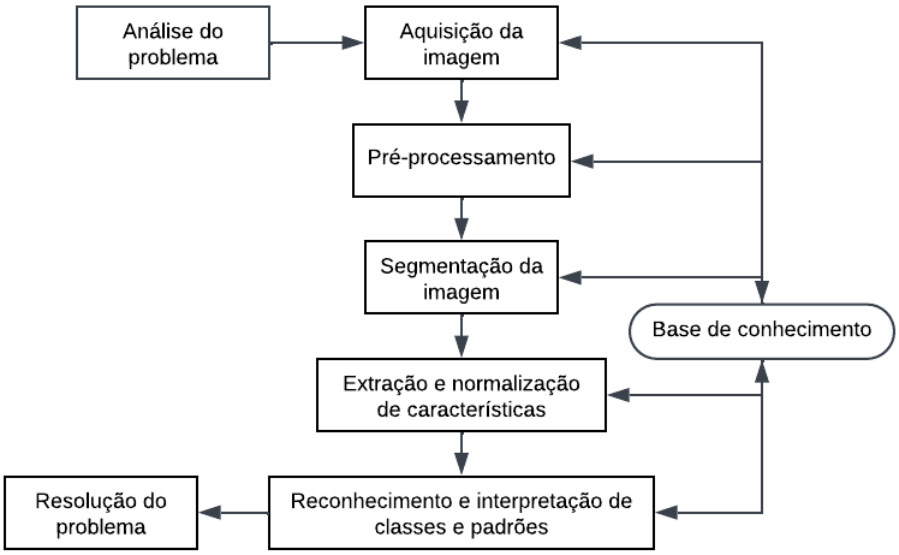
Ainda pela base de dados do *Web of Science* pode-se verificar o crescente interesse da pesquisa sobre visão computacional, uma vez que o número de artigos publicados em 2021 (2054 artigos) é aproximadamente 2 vezes maior do que há 8 anos (855 artigos em 2013) e 8 vezes maior de que há 20 anos (271 artigos em 2001) (WOS, 2022). Na década de 1980, as atenções se voltaram à matemática mais sofisticada, utilizando-se técnicas para realizar análises quantitativas de imagens e cenas.

Proveniente do avanço tecnológico, o uso da visão computacional concomitantemente com o aprendizado profundo possibilitou contribuições como o reconhecimento de objetos, reconhecimento facial e reconhecimento de ações/atividades (VOULODIMOS *et al.*, 2018).

A visão computacional não se integra apenas com o aprendizado de máquina, podendo se integrar com diversos segmentos tecnológicos que envolvem múltiplas áreas de conhecimento, tais como: agronomia, biologia, biometria, medicina e outras. Constituindo uma área com muitas aplicações práticas (NEVES, NETO e GONZAGA, 2012), como: construção de modelos 3D, imagens médicas, segurança automotiva, captura de movimento, vigilância, reconhecimento de impressões digitais (biometria) e classificação de imagens (SZELISKI, 2010).

A área de visão computacional pode incluir métodos de aquisição de imagens, pré-processamento, segmentação, extração de atributos ou características e reconhecimento de padrões (BORTH *et al.*, 2014). Estes métodos são comumente representados como uma estrutura sequencial conforme apresentado na Figura 1.

Figura 1: Representação da sequência do processamento de imagens.



Fonte: Adaptado de Lulio (2011).

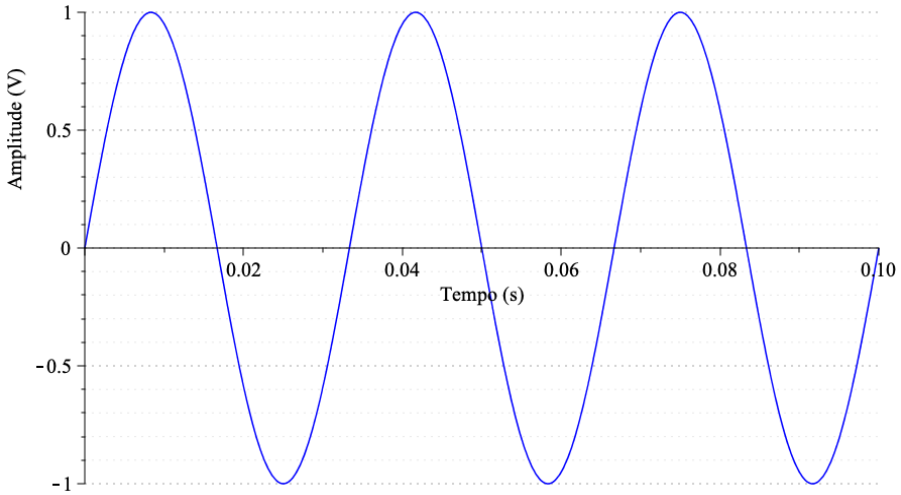
2.1.1 Aquisição da Imagem e Técnicas de Pré-processamento

Após a análise do problema, a etapa de aquisição de imagens é o primeiro passo em um sistema de visão computacional, sendo caracterizado pela captura, armazenamento e transmissão de uma imagem. Para realizar a captura de imagens, diferentes dispositivos podem ser utilizados, como, por exemplo, câmeras digitais, celulares, *smartphones*, *tablets*, infravermelhos, *webcams*, satélites etc. (BORTH *et al.*, 2014).

As técnicas de pré-processamento têm como principal objetivo melhorar a qualidade da imagem e se dividem em duas categorias principais: métodos que operam no domínio espacial e os métodos que operam no domínio da frequência (ALBUQUERQUE *et al.*, 2012). A técnica de processamento no domínio espacial opera diretamente sobre a matriz de *pixels* da imagem digitalizada, enquanto as técnicas de processamento no domínio da frequência baseiam-se na modificação da transformada de Fourier (FILHO e NETO, 1999). De forma resumida, a função dessa etapa é aprimorar a imagem para as etapas subsequentes (BORTH *et al.*, 2014).

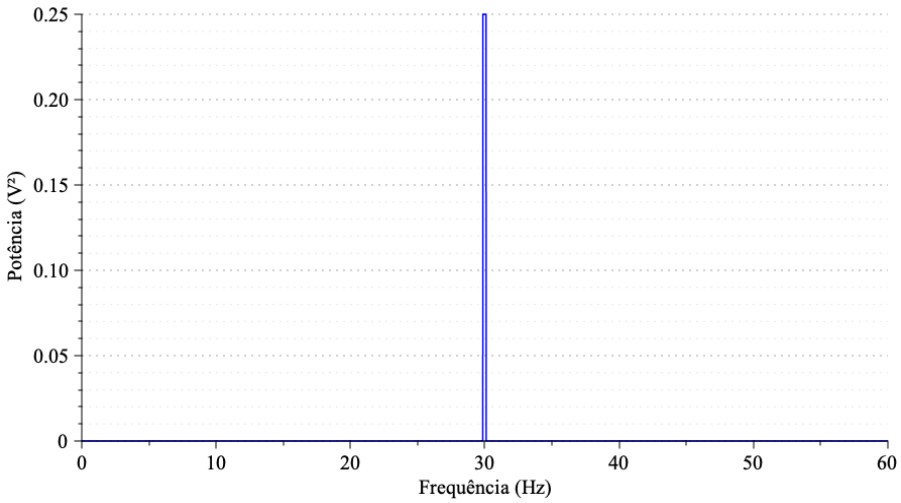
Um exemplo da técnica de processamento no domínio da frequência é o trabalho de Ocazionez (2009), desenvolvendo uma ferramenta computacional que possibilita o processamento de sinais de eletroencefalografia através do cálculo da transformada de Fourier, transformada de Fourier de curta duração e a transformada de *wavelets* na sua versão contínua e discreta. Com isso, através da análise de Fourier ele fez a transformação de um sinal no domínio do tempo passar para o domínio da frequência. Para exemplificar apresenta-se na Figura 2 e Figura 3 um sinal senoidal com frequência de 30 Hz e seu respectivo espectro de potência onde pode ser identificado um pico no valor da frequência do sinal.

Figura 2: Representação de sinal x(t)=sen(60πt).



Fonte: Imagem adaptada de Ocazionez (2009).

Figura 3: Representação de espectro de potência de x(t).

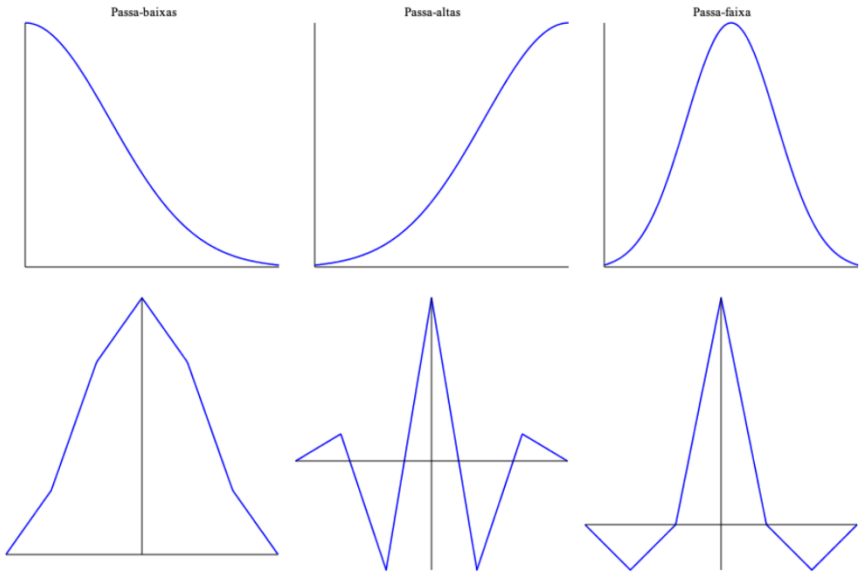


Fonte: Imagem adaptada de Ocazionez (2009).

O trabalho de Silva, Patrocínio e Schiabel (2019) é um exemplo de processamento no domínio espacial. Segundo eles grande parte das imagens que vemos é representada no domínio espacial, por uma matriz de intensidade de cor ou de escala de cinza, em um espaço bidimensional. Esses tipos de imagens no domínio espacial são amostrados discretamente a partir da intensidade de um sinal no espaço, desta forma, havendo uma correspondência direta entre as coordenadas da imagem e o espaço no "mundo real".

No livro de Filho e Neto (1999) é realizada a comparação entre o processamento no domínio de frequência e no domínio espacial, para isso eles utilizaram os filtros denominados "passa-baixa" (que atenua ou elimina os componentes de alta frequência no domínio das transformadas de Fourier, possuindo um efeito de suavização/borramento na imagem), "passa-altas" (que atenua ou elimina os componentes de baixa frequência e, em função disto, realçam as bordas e regiões de alto contraste da imagem) e "passa-faixa" (que remove ou atenua componentes acima de sua frequência de corte superior e abaixo de sua frequência de corte inferior). A Figura 4 mostra os três principais tipos de filtros existentes e os respectivos filtros espaciais correspondentes. A Figura 5, Figura 6 e Figura 7 mostram exemplos práticos da utilização dos filtros, onde a imagem "a" é a imagem original e a imagem "b" é a imagem após passar pelo respectivo filtro.

Figura 4: Respostas em frequência dos principais tipos de filtros (Acima) e filtros correspondentes no domínio espacial (Abaixo).



Fonte: Imagem adaptada de Filho e Neto (1999).

Figura 5: (a) Imagem original e (b) Imagem com o Filtro passa-baixa.

Fonte: Imagem adaptada de Bant (2014).

Figura 6: (a) Imagem original e (b) Imagem com o Filtro passa-faixa.

Fonte: Imagem adaptada de Bant (2014).

Figura 7: (a) Imagem original e (b) Imagem com o Filtro passa-alta.

Fonte: Imagem adaptada de Bant (2014).

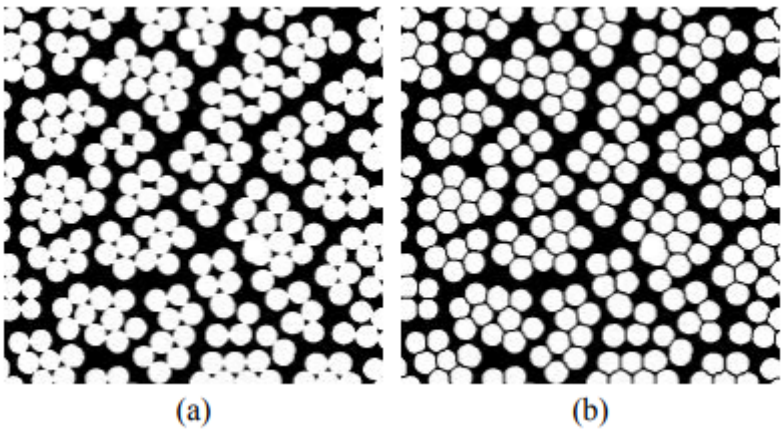
2.1.2 Segmentação e Pós-processamento

A segmentação visa reproduzir digitalmente a tarefa de reconhecer regiões de uma imagem como objetos, que é um processo cognitivo extremamente sofisticado realizado pela visão humana. A segmentação particiona a imagem em regiões e distingue estas regiões como objetos independentes uns dos outros e do fundo. Um exemplo desta etapa é uma imagem binária onde os objetos são regiões contíguas de *pixels* brancos em um fundo preto ou vice-versa (GOMES, 2007).

Muitas vezes o resultado da segmentação não é adequado, sendo necessária uma etapa de pós-processamento, como a separação de objetos que se tocam, que permite a formação de objetos mais complexos (GOMES, 2001).

A técnica clássica utilizada para separação de objetos é o método dos divisores de águas (*watersheds*), que se baseia no crescimento de sementes dos objetos. Calcula-se a imagem do Mapa de Distâncias Euclidianas (*Euclidean Distance Map* - EDM) e procede sua limiarização progressivamente, gerando sementes e agregando *pixels* a elas, sem permitir que estes agrupamentos se unam. A limiarização progride ampliando sua faixa de um em um a partir do maior valor de tom de cinza na imagem (*pixel* mais brilhante) até que a faixa vá deste valor até 1. O resultado deste processo é uma imagem binária onde linhas de 1 *pixel* de espessura (os chamados divisores de águas) dividem os objetos, como mostrado na Figura 8.

Figura 8: Método *watersheds*: a) Imagem binária inicial e b) Imagem binária com os objetos separados.



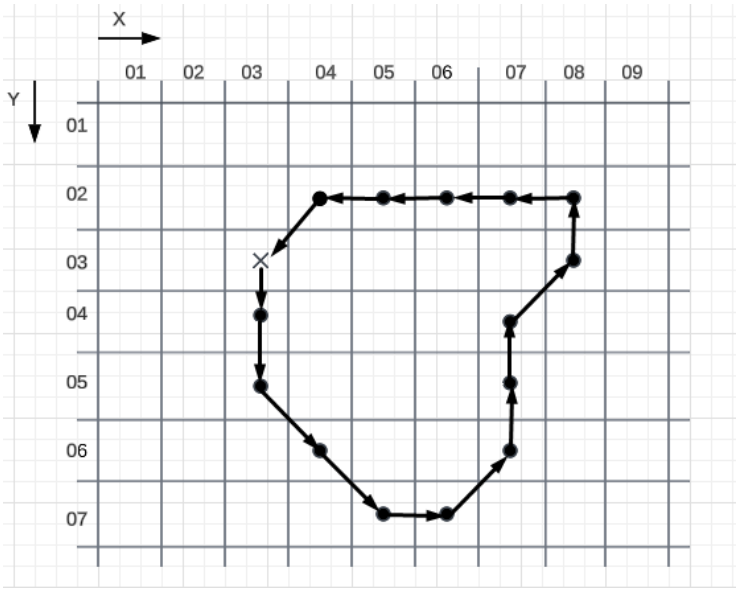
Fonte: Gomes (2001).

2.1.3 Extração de Características ou Atributos

A etapa de extração de características é realizada a partir dos dados de entrada, seguido por um algoritmo de seleção de características, que elimina os atributos mais irrelevantes segundo um determinado critério (BIANCHI, 2006). Dentro dos algoritmos de seleção há diversas técnicas, nas quais podem ser agrupadas em dois tipos: características externas (bordas) e características internas (*pixels* que compõem o objeto) (FRANCO, 2017). A escolha da representação é importante pois a partir dela diferentes algoritmos com diferentes graus de complexidade computacional podem ser utilizados (LINS, 2015).

Uma forma de representar as bordas é utilizando o código da cadeia, que, em vez de armazenar as coordenadas absolutas dos *pixels*, utiliza a posição relativa entre *pixels* consecutivos da borda. Desta forma, dado um ponto inicial pertencente à borda, o código é definido por uma sequência formada pelas direções entre cada *pixel* e seu vizinho, até que todos os *pixels* da borda sejam considerados (MIYAMOTO *et al.*, 2015). Com isso é possível recriar a borda e assim obter a medida do perímetro do objeto conforme Figura 9 (BAXES, 1994). Essa é uma das técnicas que podem ser utilizadas, porém existem outras, como aproximações poligonais, assinatura e esqueleto do objeto (MIYAMOTO *et al.*, 2015).

Figura 9: Lista do código de cadeia do contorno.



Fonte: Adaptado de Baxes (1994).

2.1.4 Reconhecimento de Padrões

O reconhecimento de objetos, tais como faces, cenas, defeitos, estruturas etc., é uma das principais tarefas de visão computacional e está diretamente relacionado ao reconhecimento de padrões (BORTH *et al.*, 2014). O objetivo principal do reconhecimento de padrões é a classificação (SOUZA, 1999).

As metodologias do reconhecimento de padrões se dividem em duas categorias: decisão teórica e a decisão estrutural (LULIO, 2011). Na decisão teórica os padrões são descritos de forma simbólica e a estrutura é a forma como estes padrões se relacionam. Na decisão estrutural são utilizadas técnicas da teoria de decisão, neste grupo os padrões são descritos por propriedades quantitativas e deve-se decidir se o objeto possui ou não estas propriedades (MARENGONI e STRINGHINI, 2009). Ambas as metodologias são usadas para extrair informações semânticas segundo algum critério.

A visão computacional utiliza as técnicas de reconhecimento de padrões para identificar imagens, elementos da imagem ou a interação entre os elementos em uma imagem. Desse modo, as técnicas de reconhecimento de padrões são utilizadas para extrair informações que tenham maior valor semântico. Essas técnicas podem ser empregadas em diversas áreas além da visão computacional, como, por exemplo, na mineração de dados e na taxonomia (CAVANI, 2007).

2.2 REDES NEURAIS ARTIFICIAIS

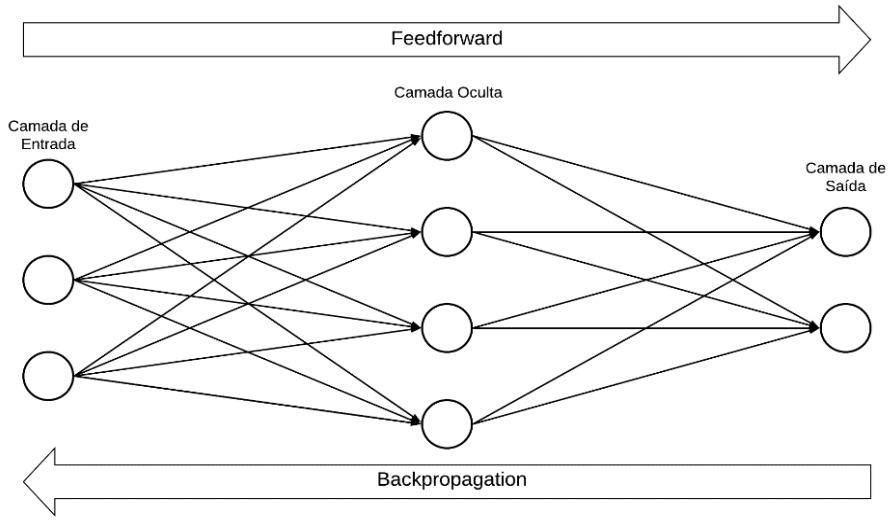
As redes neurais são compostas por, pelo menos, uma camada de entrada e uma camada de saída, podendo haver um grande número de camadas intermediárias, também denominadas de camadas ocultas. A saída dos neurônios de cada camada são as entradas dos neurônios da camada subsequente, desde a camada de entrada, até a camada de saída (AGUIAR, 2010).

Na literatura é possível verificar diversas áreas que aplicam as técnicas de redes neurais, dentre elas: reconhecimento automático de alvos, reconhecimento de caracteres,

reconhecimento de padrões, otimizações, robótica, diagnóstico médico, processamento de voz, biometria e análise de dados (OLUDARE *et al.*, 2018).

A rede neural se mostra eficiente na tomada de decisões e na busca de padrões, uma vez que aproxima entradas e saídas, convergindo para resposta correta (saída) baseando-se nos dados de entrada fornecidos (MARTINIANO *et al.*, 2016). A rede neural genérica (Figura 10), que é a mais comum, tem dois processos: *feedforward* e *backpropagation*.

Figura 10: Rede neural genérica.



Fonte: Imagem adaptada de Mohammad *et al.* (2013).

Em uma rede neural *feedforward* (isto é, que possui apenas esse processo), a informação se move em apenas uma direção: dos nós de entrada, através dos nós ocultos para os nós de saída. Não há ciclos ou loops na rede, ou seja, o *feedforward* tem como objetivo classificar o conjunto de estímulos aplicados externamente (coluna de entrada) e enviar essas informações para as próximas camadas até à camada de saída (HAYKIN, 2007).

Os pesos, normalmente representados pela letra *w*, são valores que representam o grau de importância que determinada entrada possui em relação a um determinado neurônio (HAYKIN, 2007). Assim, esse valor (*w*) para ser atribuído ao seu grau de relevância é multiplicado pelos valores de cada entrada, os quais passam para camada oculta. Posteriormente, os valores das camadas ocultas são multiplicados por novos pesos e somados, gerando resultados para camada de saída. Se o valor de saída do *feedforward* for um valor

específico, executa-se uma ação.

Um dos grandes avanços das Redes Neurais Artificias (AGUIAR, 2010) foi a implementação do *backpropagation*, formalizado por Rumelhart, Hinton e Williams (RUMELHART *et al.*, 1986) pois, ela proporciona o treinamento de redes *Perceptron* Multicamadas resultando em uma rede com grande poder de generalização e possibilitando a implementação de diversas aplicações, idealizadas atualmente (AGUIAR, 2010).

No *backpropagation* o processo de treinamento/aprendizagem ocorre e, para ajustar os pesos, são utilizados algoritmos de otimização (SLOWIK; BIALKO, 2008). Os algoritmos utilizados para a resolução de problemas de otimização podem ser de natureza determinística ou probabilística (POLAK, 1971).

2.3 REDES NEURAIS CONVOLUCIONAIS

Inicialmente proposto por Lecun *et al.* (1998) as CNN foram projetadas para aprender a extrair recursos relevantes diretamente dos *pixels* das imagens. A capacidade das CNN treinadas com o gradiente descendente para aprender completos, de alta dimensão e não lineares mapeamentos de grandes coleções de exemplos, a torna candidata óbvia para reconhecimento de imagens (LECUN *et al.*, 1998).

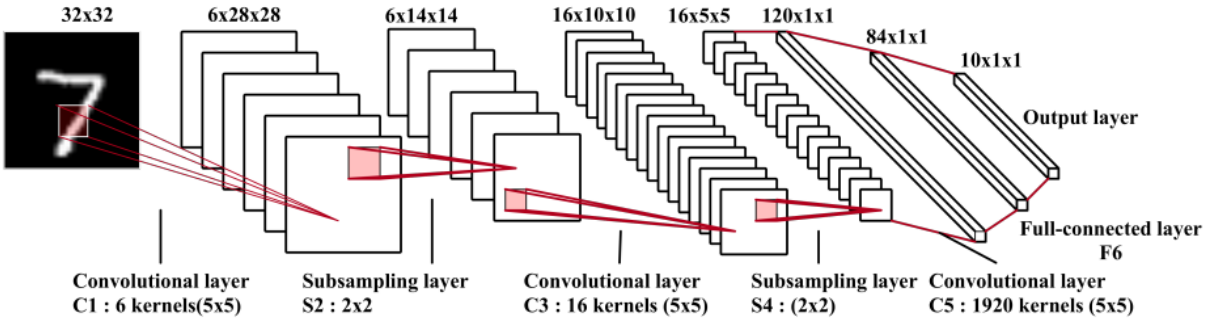
O que diferencia uma CNN de outras arquiteturas de redes neurais são as três camadas internas que ela possui, sendo respectivamente: a camada de convolução, camada de *Pooling* e por último uma camada onde todos os neurônios da rede se conectam (GU *et al.*, 2018).

Tomando como exemplo a LeNet-5, que possui as três camadas, a camada convolucional tem como objetivo aprender representações de recursos das entradas. Os *kernels* na primeira camada convolucional são projetados para detectar recursos de baixo nível, como arestas e curvas, enquanto os *kernels* em camadas superiores aprendem a codificar recursos mais abstratos.

Conforme mostrado na Figura 11, a camada de convolução é composta por vários *kernels* de convolução que são usados para calcular diferentes mapas de recursos (*feature maps*). Especificamente, cada neurônio de um mapa de recursos está conectado a uma região de neurônios vizinhos na camada anterior. Essa vizinhança é chamada de vizinhança do

neurônio (GU *et al.*, 2018).

Figura 11: A arquitetura da rede LeNet-5.



Fonte: Gu *et al.* (2018).

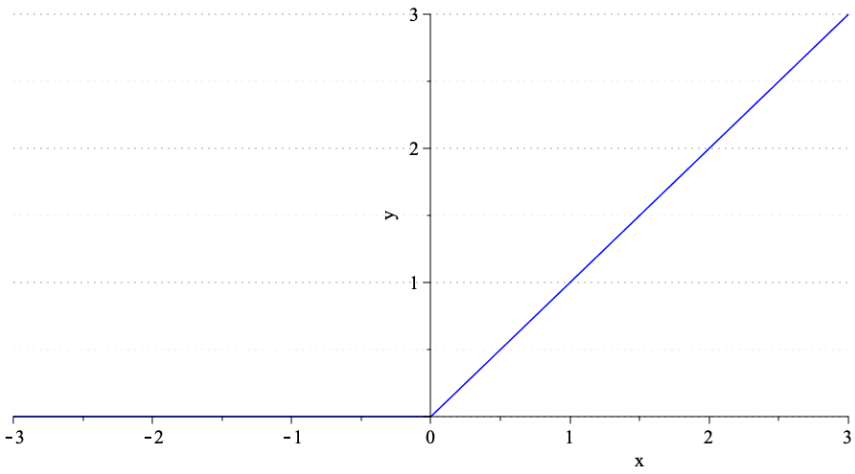
A camada de *pooling* visa alcançar a invariância de deslocamento que reduz a resolução dos mapas de recursos (*feature maps*). Cada mapa de recursos de uma camada de *pooling* é conectado ao seu mapa de recursos correspondente da camada anterior, camada convolucional (GU *et al.*, 2018).

Após várias camadas convolucionais e de *pooling*, pode haver uma ou mais camadas totalmente conectadas que visam executar raciocínio de alto nível. Eles pegam todos os neurônios da camada anterior e os conectam para cada neurônio da camada atual para gerar informações semânticas globais. Nem sempre é necessário que haja uma camada totalmente conectada, pois, ela pode ser substituída por uma camada de convolução 1 × 1 (GU *et al.*, 2018).

As camadas convolucionais geram mapas de recursos por filtros convolucionais lineares seguido por funções de ativação não lineares. Existem várias funções de ativação, dentre elas Unidade Linear Retificada (ReLu), Sigmoide e Linear.

A função ReLu é inspirada nos neurônios do cérebro, retornando valores positivos ou 0. Segundo Rizzo e Canato (2020), a função ReLu é a função de ativação mais usada no mundo e, em redes neurais convolucionais ou para aprendizado profundo é a mais eficiente. A função ReLu é definida como: $f(x) = \max_{x \in R}(0, x)$. Onde: x pertencente aos números reais e $max\,(0, x)$ retorna valores positivos ou valor zero, conforme exemplificado em sua representação gráfica na Figura 12.

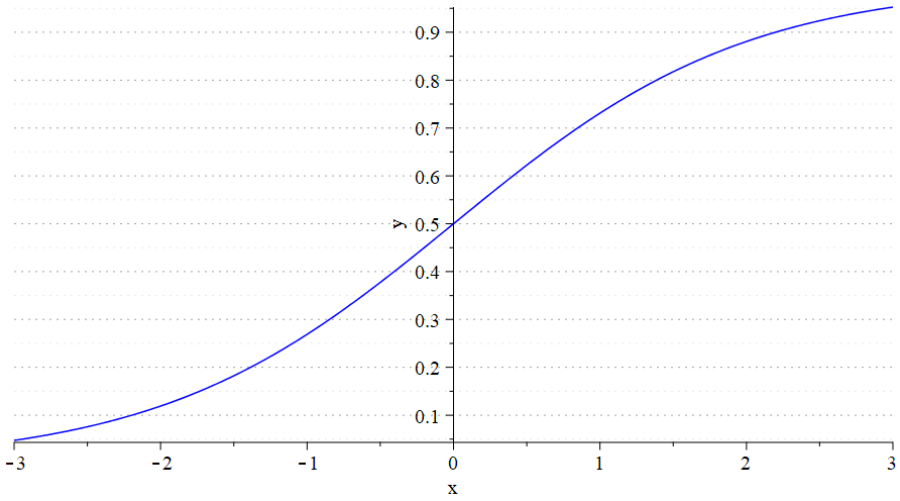
Figura 12: Representação gráfica da função de ativação ReLu.



Fonte: Imagem adaptada de Cunha (2022).

A função Sigmoide definida por $f(x) = \frac{1}{1+e^{-x}}$, segundo Gharat (2019), possuí formato de curva em 'S' e, ela mapeia qualquer valor real em outro valor entre 0 e 1. Sua representação gráfica se encontra na Figura 13.

Figura 13: Representação gráfica da função Sigmoide.

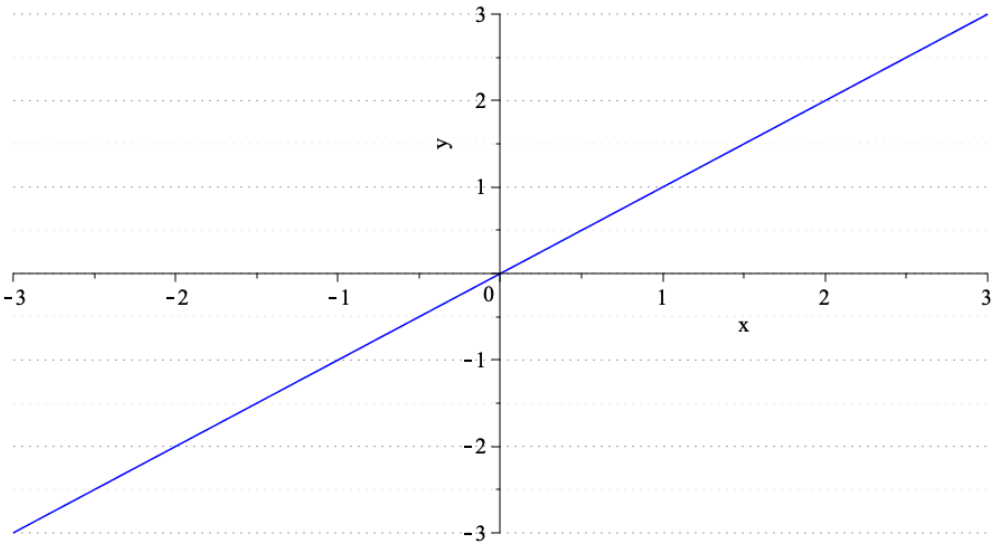


Fonte: Imagem adaptada de Rizzo e Canato (2020).

Por fim, a função linear é diretamente proporcional aos valores de entrada (SHARMA; SHARMA, 2020). Sua representação matemática é: $f(x) = ax$ (CUNHA, 2022). E sua

representação gráfica encontra-se na Figura 14.

Figura 14: Representação gráfica da função linear.



Fonte: Imagem adaptada de Cunha (2022).

No modelo tradicional de reconhecimento de padrões, um extrator de características reúne informações relevantes da entrada e elimina variabilidades irrelevantes. Em seguida, um classificador treinável categoriza os vetores de recursos resultantes em classes. No caso de reconhecimento de caracteres, uma rede pode ser alimentada com insumos quase brutos (por exemplo, imagens com tamanho normalizado). Embora isso possa ser feito com uma rede *feedforward* comum totalmente conectada, há problemas (LECUN *et al.*, 1998).

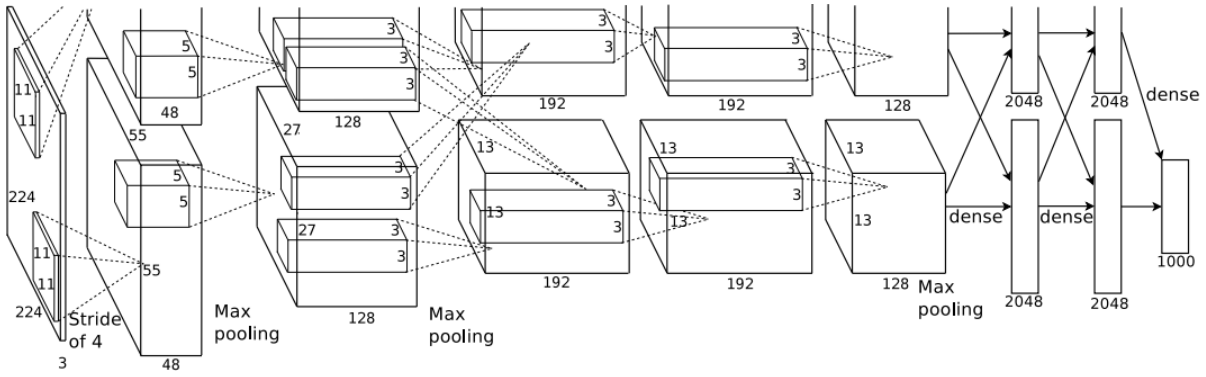
Primeiro, as imagens são grandes, muitas vezes elas possuem várias centenas de variáveis (*pixels*). Uma primeira camada totalmente conectada com, por exemplo, cem neurônios ocultos na primeira camada intermediária contêm dezenas de milhares de pesos. Esse grande número de parâmetros aumenta a capacidade do sistema e, portanto, requer um conjunto de treinamento maior. Além disso, o requisito de memória para armazenar tantos pesos pode descartar certas implementações de *hardware*. Mas a principal deficiência de redes não estruturadas para aplicações de imagem ou fala é que elas não têm invariância interna em relação às traduções ou distorções locais das entradas (LECUN *et al.*, 1998).

2.3.1 AlexNet

AlexNet é o nome de uma Rede Neural Convolucional (CNN) projetada por Alex Krizhevsky (KRIZHEVSKY; SUTSKEVER; HINTON, 2012). AlexNet é uma arquitetura que consiste em 5 camadas convolucionais, seguidas de 3 camadas de *pooling* que é seguida por três camadas completamente conectadas. Para reduzir o problema de sobre ajuste (*overfitting*), essas camadas totalmente conectadas são usadas com a camada *dropout*. Usos da camada de convolução: número de filtros para convoluir a imagem, e gerar mapas de características. A camada de unidade linear retificada (ReLU) é usada junto com a camada convolucional, pois a mesma executa uma operação não linear e converte todos os valores negativos para zero. A tarefa de juntar a camada é reduzir a dimensão espacial (*feature map*) que é derivada da camada anterior (ARYA e SINGH, 2019).

Krizhevsky (2017) introduziu a AlexNet, mostrando de forma explicita o delineamento de responsabilidades entre duas GPUs. Onde uma GPU executa as partes da camada na parte superior, enquanto a outra executa as partes da camada no fundo (Figura 15). As GPUs se comunicam apenas em determinadas camadas.

Figura 15: AlexNet introduzida por Krizhevsky.



Fonte: Krizhevsky (2017).

2.3.2 Lenet-5

As redes convolucionais combinam três ideias para garantir algum grau de invariância de deslocamento, escala e distorção, sendo eles: 1) campos receptivos locais; 2) pesos compartilhados (ou replicação em peso); e 3) subamostragem espacial ou temporal. Uma rede convolucional típica para reconhecer caracteres, apelidado de LeNet-5 (Lecun *et al.*, 1998), é mostrado na Figura 11.

O plano de entrada recebe imagens de caracteres com tamanho normalizado e centrado. Cada unidade em uma camada recebe entradas de um conjunto de unidades localizadas em uma pequena vizinhança na camada anterior (Lecun *et al.*, 1998). A ideia de conectar unidades de campos receptivos locais na camada de entrada e voltar para o perceptron teve início nos anos 1960, e foi quase simultâneo a descoberta de Hubel e Wiesel de neurônios localmente sensíveis e seletivos de orientação no sistema visual do gato (HUBEL e WIESEL, 1962). Com o campo receptivo local neurônios podem extrair características visuais elementares como arestas orientadas, extremidades, cantos ou recursos semelhantes em outros sinais, como espectrogramas de fala. Esses recursos são então combinados pelas camadas subsequentes para detectar recursos de ordem superior (Lecun *et al.*, 1998).

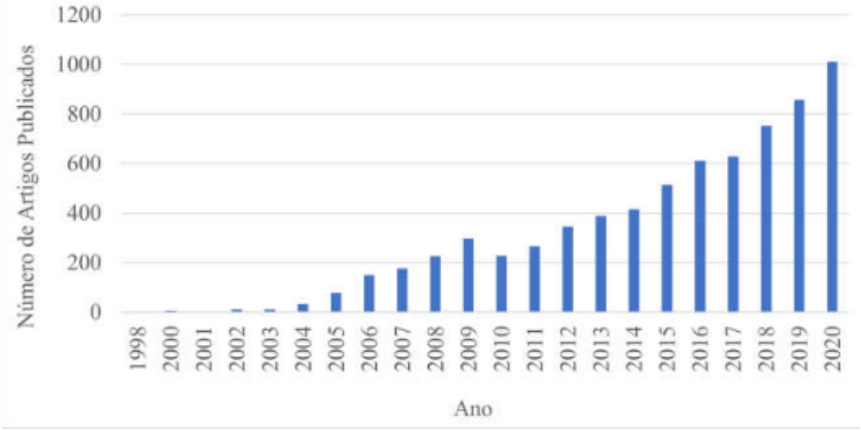
2.4 PARTICLE SWARM OPTIMIZATION

Existem algumas formas de se otimizar as redes neurais, dentre as formas pode-se citar o *Particle Swarm Op**ti**mization* (PSO) que é uma técnica estocástica criada em 1995 por James Kennedy e Russell Eberhart para simular comportamento de um bando de pássaros (KENNEDY; EBERHART, 1995). Um enxame pode ser caracterizado como um conjunto estruturado de organismos interativos. A ideia do algoritmo utiliza como base o bando de pássaros e, deste modo, costuma-se utilizar termos técnicos, tais como: espaço de busca, solução ótima, partículas, iteração, melhor local e melhor global, que são, respectivamente, a área sobrevoada pelos pássaros, a localização do objetivo (alimento, ninho), os pássaros, o número de vezes que o grupo se movimentaria, a melhor posição conhecida pelo pássaro

(experiência individual) e a melhor posição conhecida pelo bando de pássaros (experiência coletiva) (ALMEIDA e NAKAJIMA, 2021).

Pesquisas envolvendo o método PSO relacionado às redes neurais vem crescendo a cada ano, como indica a pesquisa realizada na base de dados *Web of Science* (WOS, 2021) com os termos "*particle swarm optimization*" e "*neural network*", a qual indica a primeira publicação (ZHENYA *et al.*, 1998) com o tema no ano de 1998 e mais de mil publicações em 2020, com um crescimento anual médio de 16,34% nos últimos dez anos, como indicado na Figura 16 (ALMEIDA e NAKAJIMA, 2021).

Figura 16: Artigos publicados por ano com os termos "*Particle swarm optimization*" e "*neural network*".



Fonte: Almeida e Nakajima (2021).

O esquema do PSO, descrito para *N* partículas, pode ser apresentado da seguinte forma (MARINI; WALCZAK, 2015):

1. Inicialização:

a. Inicialize a posição X_0^ipara todo i de **1** a $\boldsymbol{N}$;

b. Inicialize a melhor posição como a posição inicial;

c. Calcule a função *fitness* (objetivo) para cada partícula e faça a melhor partícula aquela que recebeu o melhor resultado da função.

2. Até atingir o critério de parada, faça:

a. Atualize a velocidade:

$$v_i^{k+1} = v_i^k + c_1\left(p_i - X_i^k\right) + c_2\left(g - X_i^k\right)$$

b. Atualize a posição da partícula;

c. Calcule a função *fitness* de cada partícula;
d. Atualize a melhor partícula local;
e. Atualize a melhor partícula global.

Em que X_i^k é a posição da partícula *i* na iteração *k*, p_i é a melhor posição da partícula *i*, *g* é a melhor posição entre todas as partículas, c_1 e c_2 são constantes aleatórias definidas previamente (ALMEIDA e NAKAJIMA, 2021). De acordo com Schwaab *et al.* (2008), as constantes c_1 e c_2 podem ser definidas com valor igual a 1,5.

2.5 CLASSIFICAÇÃO DE IMAGENS

De acordo com Rosa (1990), classificação significa a associação de pontos de uma imagem a uma classe/grupo ou ainda o processo de reconhecimento de classes ou grupos cujos membros exibem características comuns, ou seja, esse processo de classificação se caracteriza pela atribuição aos conjuntos de *pixels* que possuem características comuns entre si (TANGERINO e LOURENÇO, 2013).

A classificação de imagens pode ser dividida em supervisionada e não supervisionada. A supervisionada é utilizada quando se tem algum conhecimento sobre as classes da imagem. A classificação não supervisionada é útil quando não se tem informações sobre o número de classes presentes nas imagens (ROSA, 1990).

O método é dito não supervisionado quando o classificador não utiliza qualquer informação sobre as categorias existentes na imagem e define, sem a interferência do analista, a estratificação da cena, atribuindo a cada *pixel* uma determinada classe. Tal abordagem corresponde à técnica de segmentação de imagens, onde as mesmas são divididas em certas classes sem conhecimento prévio. O algoritmo define estas classes com base em regras estatísticas pré-selecionadas (RICHARDS, 1986).

O método de classificação é dito supervisionado quando há um conhecimento prévio de certas áreas em que se deseja trabalhar, o que permite a seleção de amostras de treinamento confiáveis. O algoritmo classificador opera com base na distribuição de probabilidade de cada classe selecionada (ADENIYI, 1985).

Apesar de não haver um consenso na delimitação das áreas no processamento computacional de imagens, geralmente, são considerados três níveis: o processamento de baixo nível, onde há o envolvimento de operações de pré-processamento, como melhoria na nitidez da imagem, redução de ruído e ajuste de contraste, onde tanto a entrada quanto a saída são imagens como um todo; o processamento de nível médio, que envolve tarefas como a segmentação, o ajuste de objetos para um processamento computacional, a classificação e o reconhecimento deles numa imagem, sendo essa tarefa caracterizada pela entrada que geralmente é uma imagem, e a saída que é um objeto de interesse; o último tipo é o de alto nível, envolve o reconhecimento de objetos com significado semântico, que geralmente requer uma informação contextual e inteligência artificial (OLIVEIRA, 2003).

2.6 MATRIZ DE CONFUSÃO

Segundo Castro e Braga (2011), tradicionalmente, a acurácia é utilizada na avaliação e seleção de modelos de classificação, medida em relação a um conjunto de teste específico. Tal abordagem é justificada pela busca da minimização da probabilidade de erro global. No entanto, os autores destacam a necessidade de considerar uma abordagem mais eficaz para avaliar um classificador, a qual envolve a análise dos acertos e erros para cada classe individualmente. Esta análise pode ser realizada por meio da interpretação do desempenho a partir de uma matriz de confusão, a qual compara os dados reais com os valores classificados pelo modelo. Um exemplo de matriz de confusão para classificadores é apresentado na Figura 17.

Figura 17: Matriz de confusão para classificadores.

Matriz de confusão		Situação real (referência)	
		Negativo	Positivo
Classe predita (modelo)	Negativo	Verdadeiro Negativo (TN)	Falso Positivo (FP)
	Positivo	Falso Negativo (FN)	Verdadeiro Positivo (TP)

Fonte: Adaptado de Fawcett (2006).

Conforme Castro e Braga (2011), os elementos situados ao longo da diagonal principal (em destaque na cor verde, na Figura 17) na matriz de confusão denotam as predições precisas realizadas pelo modelo. Esses elementos correspondem aos verdadeiros positivos (VP) e verdadeiros negativos (VN). Por outro lado, os elementos que não se encontram na diagonal representam os equívocos do modelo, identificados como falsos positivos (FP) e falsos negativos (FN).

Luque *et al.* (2019) ressaltam métricas que podem ser extraídas da matriz de confusão para compor os demonstrativos dos resultados, de acordo com as fórmulas abaixo:

- Sensibilidade ou taxa de verdadeiros positivos = $\frac{TP}{TP+FN}$
- Especificidade ou taxa de verdadeiros negativos = $\frac{TN}{TN+FP}$
- Acurácia ou taxa de casos corretamente classificados = $\frac{TN+TP}{TN+TP+FN+FP}$
- Precisão ou taxa de positivos previsto = $\frac{TP}{TP+FP}$

2.7 ESTADO DA ARTE

O reconhecimento de texto, seja ele em cena, vídeo e/ou nascido digitalmente de imagens, é considerado uma tarefa desafiadora (YE e DOERMANN, 2014) devido ao ambiente restrito, como variações de fundo, condição de iluminação irregular, movimento desfoque, baixa resolução, alto contraste, além das imagens incluem multi-orientação, multi-tipos, multi-estilo, diferentes fontes e tamanhos de fonte (RAGHUNANDAN *et al.*, 2018).

Segundo Farhani, Terbeh e Zrigui (2017) para se obter descrições relevantes das imagens normalmente utiliza-se dois tipos de abordagens: abordagem utilizando metadados e a abordagem para prever o conteúdo da imagem.

A abordagem utilizando metadados explora o texto existente para criar a descrição/frase da imagem. Há casos em que se encontra o texto na imagem, que é o caso da obra de (FENG e LAPATA, 2010) onde os autores fizeram um *software* que pré-processa as imagens e sugere palavras-chave para descrever seu conteúdo.

Já a abordagem para prever o conteúdo da imagem consiste em construir descrições/frases do zero em vez de recuperar o texto existente. Por exemplo, Yang *et al.*

(2011) que detecta objetos e cenas de uma imagem de entrada e, então usa estatísticas de texto para associar verbos para ter relações entre objetos.

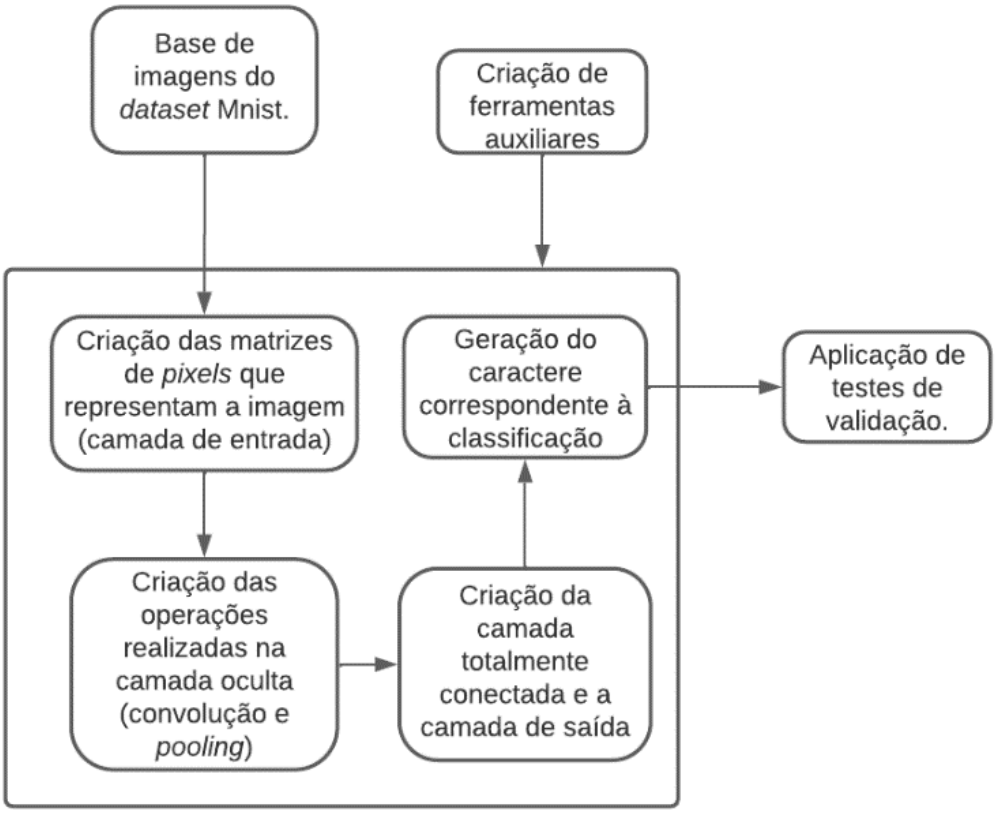
De forma similar ao trabalho proposto, há o Google Lens, que além de identificar os textos em imagens, traduz para mais de 100 idiomas em tempo real. Possuindo também outras funções como: identificação de plantas e animais, descobrir produtos e encontrar imagens visualmente parecidas. O Lens funciona comparando os objetos da foto enviada ao aplicativo com outras imagens e as classifica com base na relevância e semelhança com esses objetos (LENS, 2022).

Mesmo não sendo exclusivo para identificação de textos contidos em imagens, o Google Keep consegue realizar essa funcionalidade. Conseguindo ler textos complexos (com letras, números e caracteres especiais) (KEEP, 2022).

3 METODOLOGIA

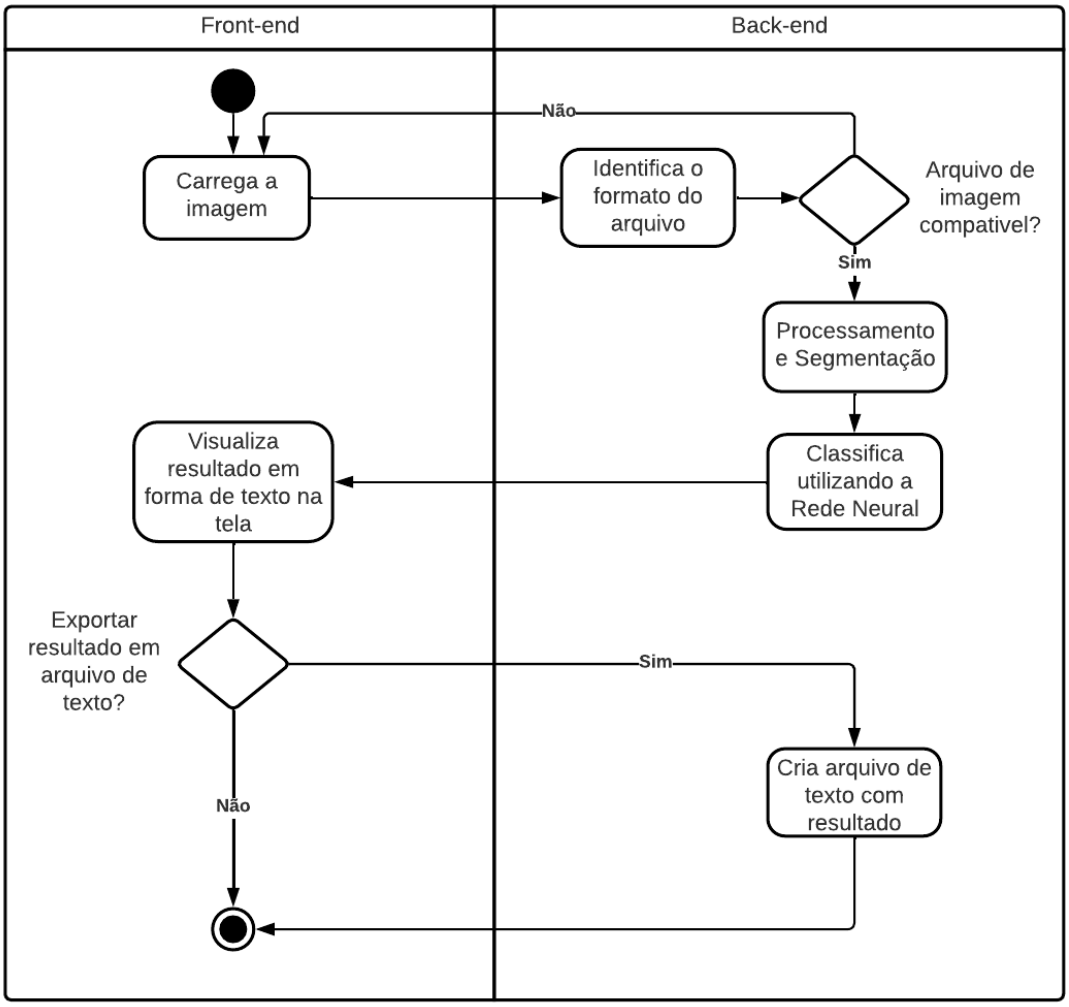
Os métodos e atividades necessárias para o desenvolvimento deste trabalho foram divididos em 5 etapas principais (Figura 18 e Figura 19): criação da base de imagens, desenvolvimento e treinamento da RNA, desenvolvimento do *software*, aplicação de testes estatísticos para os resultados encontrados e desenvolvimento da interface gráfica direcionada aos usuários.

Figura 18: Fluxograma de atividades do projeto.



Fonte: Elaborado pelo autor (2022).

Figura 19: Diagrama de atividades do *software* do usuário.



Fonte: Elaborado pelo autor (2022).

3.1 BASE DE IMAGENS

A base de imagens foi inicialmente gerada a partir de diferentes fontes compreendendo todos os caracteres alfabéticos da língua portuguesa. As 50 fontes selecionadas foram sorteadas dentre as fontes presentes no pacote *Office*, totalizando uma base de imagens inicial de 1300 imagens.

As imagens foram adicionadas em uma planilha do *Excel*, onde cada célula continha um caractere, por fim, com todas as 50 fontes escolhidas, de tamanhos 24, as letras foram

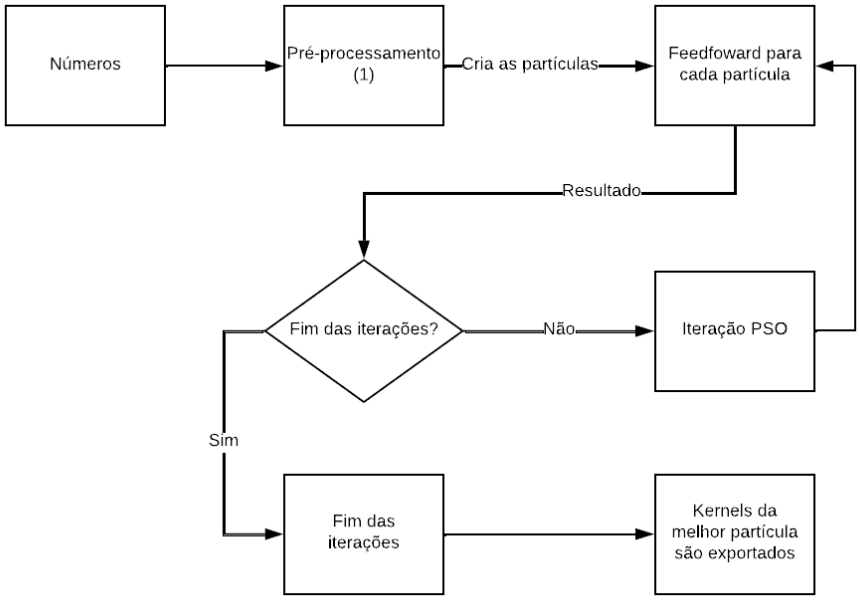
recortadas em tamanho padronizado (15 *pixels* x 15 *pixels*), com os caracteres centralizados na imagem.

O algoritmo implementado neste trabalho visava originalmente classificar as letras, mas, o mesmo apresentou uma baixa acurácia devido ao baixo número de imagens (1300) de treino criadas quando comparado a base do Mnist que possui 697.932 imagens. E, devido a capacidade de processamento e a forma como o algoritmo foi elaborado, tornou-se inviável manter a classificação de letras. Com isso, a base foi atualizada para a abordagem de classificação de números manuscritos, utilizando a base de imagens do próprio Mnist, assim, utilizando o tamanho das imagens da base do Mnist, 28 *pixels* de altura por 28 *pixels* de largura.

3.2 PRÉ-PROCESSAMENTO DIGITAL DAS IMAGENS

A base de imagens foi submetida a etapas de pré-processamento antes de serem submetidas ao treinamento da RNA deste trabalho e, leva como base o fluxograma apresentado na Figura 20.

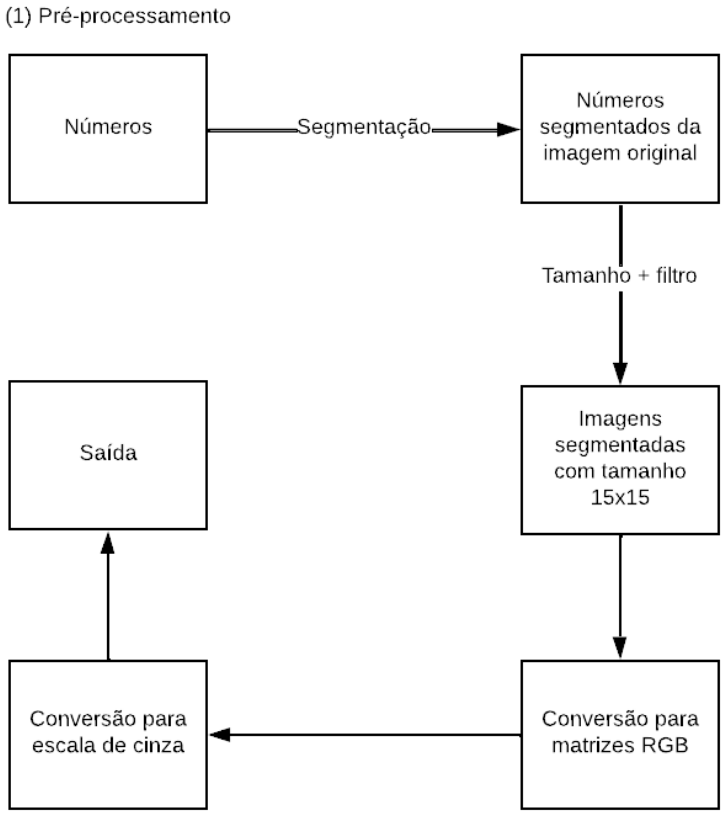
Figura 20: Fluxograma geral da etapa de treino.



Fonte: Elaborado pelo autor (2023).

A primeira etapa (“pré-processamento (1)” apresentado na Figura 20) consiste em realizar a segmentação dos números presentes na imagem de entrada do usuário, onde cada número presente na imagem fica com o tamanho final de 28 *pixels* de altura por 28 *pixels* de largura, mantendo a proporção presente nas imagens utilizadas para treinamento, presentes na base de imagens do Mnist. As imagens dos números segregados são passados juntamente com filtros para a realizar a conversão deles para matrizes RGB (*Red-Green-Blue*) e então passá-la para escala de cinza, gerando a saída dessa etapa. Essa etapa é apresentada também no fluxograma da Figura 21.

Figura 21: Fluxograma da etapa de pré-processamento.

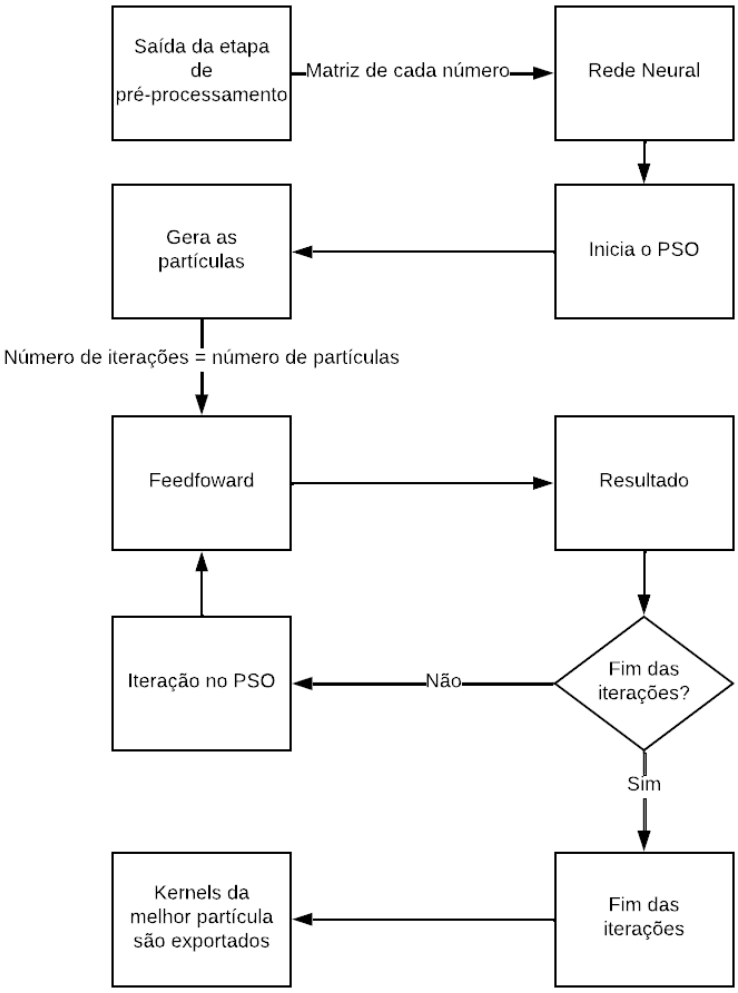


Fonte: Elaborado pelo autor (2023).

Para realizar a transformação das imagens para a escala de cinza os três canais de cores de RGB foram unificados para um canal único de cor que varia entre 0 a 255. Os valores para escala de cinza são unificados utilizando a seguinte fórmula: R * 0.3 + G * 0.59 + B * 0.11.

A matriz de cada número em encala de cinza, com tamanho 15 *pixels* x 15 *pixels*, resultado da saída da Figura 21, entra na rede neural. Iniciando o PSO, gerando as partículas e passando para o *Feedfoward* o número de iterações sendo igual ao número de partículas, onde cada *Feedfoward* gera um resultado. Essa etapa é apresentada na Figura 22.

Figura 22: Fluxograma da etapa final do pré-processamento das imagens.



Fonte: Elaborado pelo autor (2023).

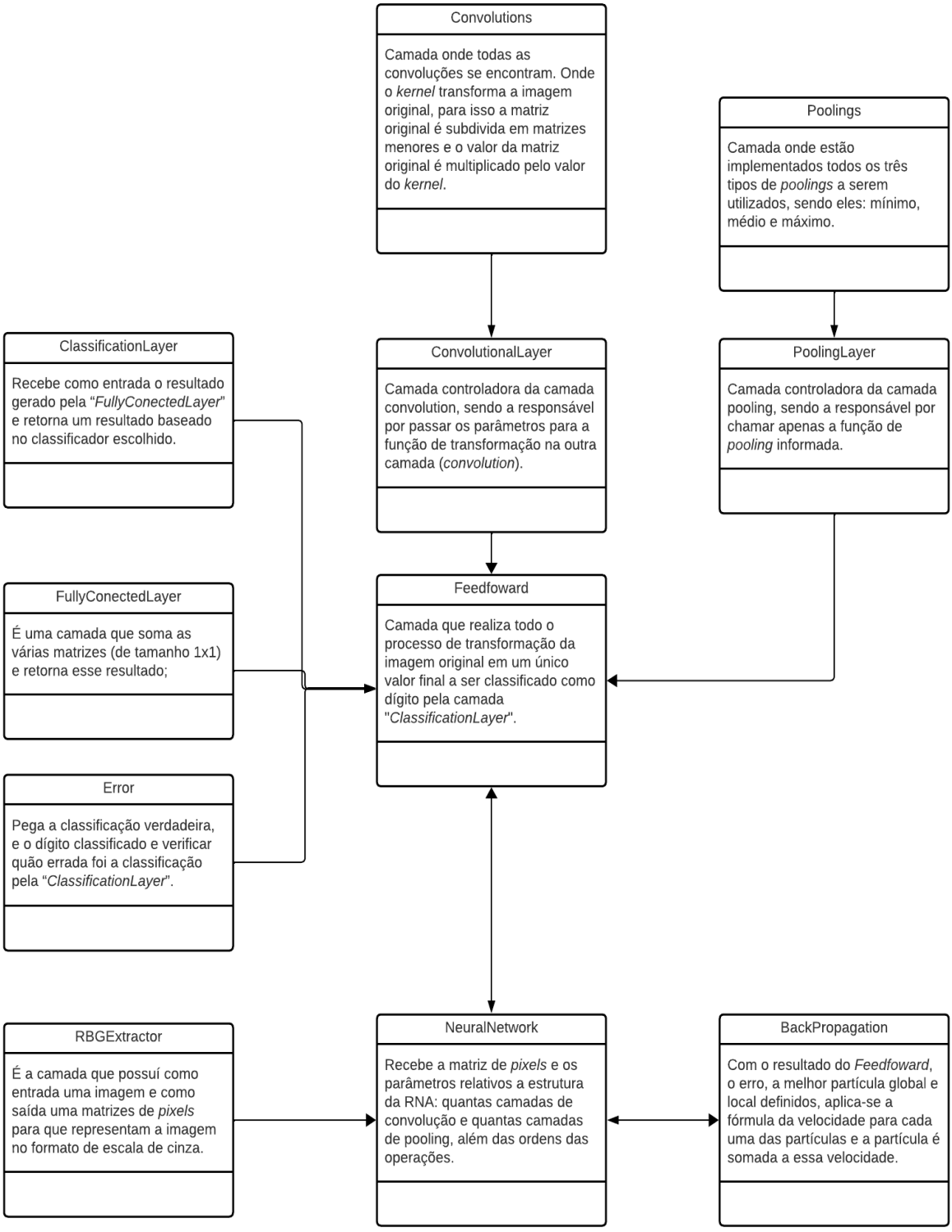
3.3 ARQUITETURA E TREINAMENTO DA RNA

A rede neural elaborada para treinamento do aplicativo dos usuários foi criada baseada na Figura 23, sendo construída levando em consideração o modelo Builder, para simplificar alterações e melhorias no código e, consequentemente melhorar a rede neural treinada. A RNA foi baseada na RNA mostrada na Figura 11 LeNet-5. O modelo utilizado possui 1 camada de *pooling* máximo (com tamanho 2x2), uma camada de convolução (com 10 *kernels*, com tamanho 13x13) e outra camada de *pooling* máximo (com tamanho 2x2) e, por fim, a camada totalmente conectada. Desta forma, a fórmula final ficaria: 28/2 = 14 – 13 + 1 = 2/2 = 1.

- *RGBExtractor*: camada onde possuí como entrada uma imagem e como saída três matrizes de pixels para que representam a imagem no formato RGB;
- *Convolution*: camada onde o *kernel* transforma a imagem original, para isso a matriz original é subdivida em matrizes menores (tamanho da matriz da imagem – tamanho do *kernel* + 1) e o valor da matriz original é multiplicado pelo valor do *kernel*;
- *Pooling*: camada onde estão implementados todos os três tipos de *poolings* a serem utilizados, sendo eles: mínimo, médio e máximo;
- *ConvolutionalLayer*: camada controladora da camada *convolution*, sendo a responsável por passar os parâmetros para a função de transformação na outra camada (*convolution*);
- *PoolingLayer*: camada controladora da camada *pooling*, sendo a responsável por chamar apenas a função de *pooling* informada, sendo: 1 = *pooling* mínimo, 2 = *pooling* médio e 3 = *pooling* máximo;
- *Feedfoward*: camada que realiza todo o processo de transformação da imagem original em um único valor final a ser classificado como letra pela camada *classificationLayer*. Para que isso ocorra é passado uma ordem de operações e realizado as camadas envoltas no *pooling* e na *convolution*. Visto que, dependendo do tamanho das matrizes de *pooling* e/ou convolução (*convolution*) não seria possível realizar as operações matemáticas, nesta camada também havia um verificador de tamanho, que informa se seria possível ou não realizar as operações com os tamanhos das matrizes de *kernel*, *pooling* e a matriz original;
- *FullyConectedLayer*: é uma camada que soma as várias matrizes (de tamanho 1x1) e retorna esse resultado;

- *ClassificationLayer*: recebe como resultado o valor gerado pela "*FullyConectedLayer*" e retorna um resultado baseado no classificador escolhido, sigmoide ou tangente hiperbólica, informando assim, qual é a letra;
- *Error*: pega a classificação verdadeira (letra correta), e a letra classificada e verificar quão errada foi a classificação pela "*ClassificationLayer*" realizando uma subtração entre a posição da classificação correta e a classificação dada pela RNA, com o resultado em módulo sendo seu retorno;
- *Backpropagation*: com o resultado do *Feedfoward*, o erro, a melhor partícula global e local definidos, aplica-se a fórmula da velocidade para cada uma das partículas $v_{ij}^{k+1} = v_{ij}^{k} + c_1\lambda_1\left(L_{ij} - X_{ij}^{k}\right) + c_2\lambda_2\left(G_{ij} - X_{ij}^{k}\right)$ onde: i é a linha da matriz correspondente, j é a coluna, L é o melhor *kernel* local, G é o melhor *kernel* global, c_1 e c_2 são constantes pré-definidas, λ_1 e λ_2 são valores randômicos entre 0 e 1, k é a iteração e X é o *kernel*. Desta forma, a velocidade da partícula é calculada e a partícula é somada a essa velocidade;
- *NeuralNetwork*: Recebe a matriz de pixels e os parâmetros relativos à estrutura da *convnet*: quantas camadas de convolução e quantas camadas de *pooling*, além das ordens das operações.

Figura 23: Diagrama de explicação das classes da rede neural para treinamento do aplicativo.



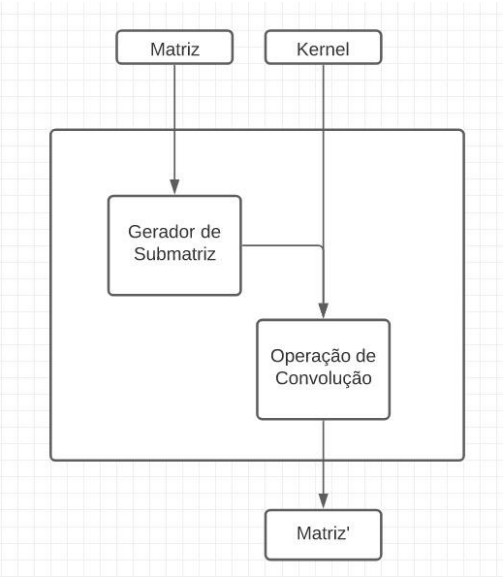
Fonte: Elaborado pelo autor (2023).

3.2 DESENVOLVIMENTO DO *SOFTWARE*

3.2.1 Ferramentas Auxiliares

Essa etapa contempla principalmente as funções envolvendo operações entre matrizes. A função de convolução (Figura 24), que tem como entrada uma matriz de tamanho $n \times n$, um *kernel* (que é uma matriz de tamanho $m \times m$) e, como saída uma matriz de tamanho $n\text{-}m+1 \times n\text{-}m+1$. Para que ocorra essa saída uma outra função extrai uma matriz do mesmo tamanho do *kernel* a partir da matriz de entrada e, com essa submatriz e o *kernel* é realizada a operação de convolução. Somando-se as entradas da matriz resultante da convolução, tem-se uma das entradas da matriz final. O processo é realizado em todas as submatrizes de tamanho $m \times m$ da matriz original.

Figura 24: Sequência da função de convolução.



Fonte: Elaborado pelo autor (2022).

Para a função *pooling* é necessário ter um tamanho e qual o tipo utilizado. Neste trabalho foram considerados os tipos: mínimo, máximo e médio. Yildirim (2021) explica que ambas as técnicas são utilizadas nas CNN e, cada um tem seus pontos fortes, como: o mínimo que ajuda a selecionar os *pixels* mais escuros da imagem, já o máximo ajuda a selecionar os

mais claros e o médio é usado para extrair características suaves da imagem.

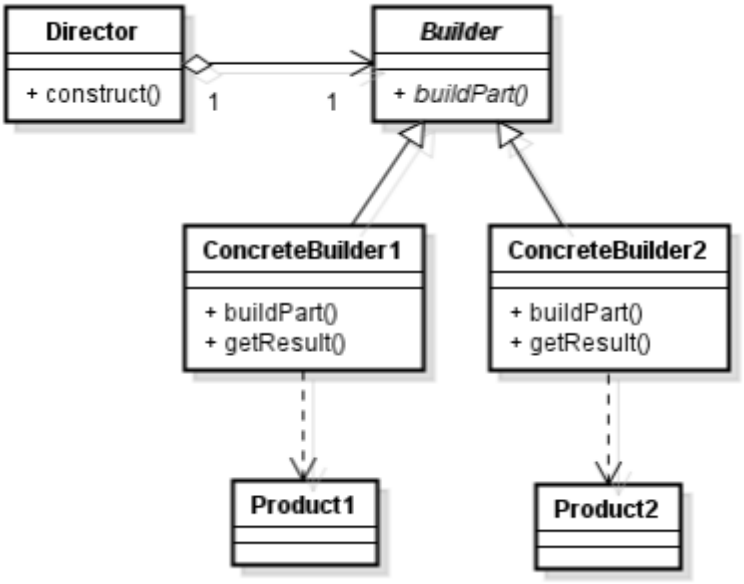
Ou seja, a partir da matriz original, é extraída uma submatriz de tamanho igual ao do *pooling* desejado (*K*) e a operação é realizada, com isso se obtém uma das entradas da matriz resultante. Esse processo percorre todas as submatrizes de tamanho (*K*).

Ainda dentro das ferramentas auxiliares entra também o PSO, onde cada partícula seria um *kernel*, dentro de um intervalo indo de 10 a 200 partículas, onde eram realizadas de 100 a 1000 iterações. Criando muitos indivíduos com pequenas diferenças em seus pesos, para que, os melhores sejam modificados e replicados, até que chegue cada vez mais perto de um ótimo global, neste caso, a identificação de caracteres.

3.2.2 Rede Neural

Para a elaboração da Rede Neural, foi utilizado a linguagem de programação Java com o padrão de projeto criacional *Builder*. Esse tipo de padrão subdivide o processo de construção da estrutura de um objeto, de modo que o mesmo processo de construção possa criar várias representações diferentes (LUQUE e SILVA, 2014). A proposta deste padrão envolve a definição de uma classe, *Director*, que fica responsável pela criação do objeto que se deseja construir, chamado de forma genérica, de *Product*. Com isso a classe *Director* conduz a criação do produto passo a passo através de chamadas aos métodos de uma classe de construção. O diagrama de classes desse padrão de projeto pode ser visto na Figura 25.

Figura 25: Diagrama de classes do padrão *Builder*.



Fonte: Luque e Silva (2014).

Segundo Luque e Silva (2014), o *Builder* tem como motivação ser utilizado quando é necessário construir um objeto/projeto através de várias etapas de forma que, dependendo dos parâmetros especificados, diferentes representações desse objeto sejam obtidas. Algumas das vantagens de se utilizar esse padrão de projeto são: possuir um controle maior sobre o processo de construção do objeto, dado que ele é executado passo a passo e esconder a forma como as diferentes partes de um objeto são criadas e unidas durante sua construção.

As classes participantes do padrão são:

Director: constrói objetos passo a passo enviando mensagens a um *Builder*;

ConcreteBuilder: define uma implementação da interface *Builder*;

Builder: define uma interface (classe abstrata ou interface) que deve ser implementada pelos diferentes construtores de objeto;

Product: o objeto complexo em construção (inclui classes que definem as partes constituintes).

O padrão de projeto para o desenvolvimento deste projeto foi o *Builder*. E o diagrama de explicação das classes do projeto encontra-se na Figura 23. A Figura 18 mostra fluxograma de atividades proposto para este trabalho, a camada de entrada da rede neural, é uma imagem

transformada em uma matriz de *pixels*, que é bidimensional (imagens padronizadas em duas faixas de cores, como: preto e branco).

Após a matriz de *pixels* passa pelas camadas de convolução, *pooling*, pela camada totalmente conectada e ser gerado o caractere correspondente. Onde os tamanhos dos *kernels* da camada de convolução eram de: 2x2 e 5x5, enquanto há três tipos de *pooling*, mínimo, máximo e médio, explicados anteriormente.

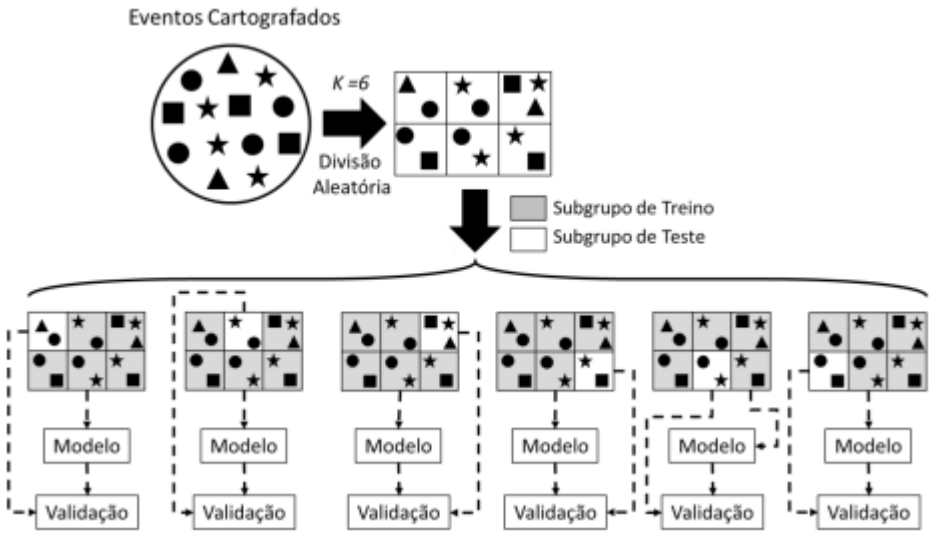
3.3 APLICAÇÃO DOS TESTES NO CLASSIFICADOR

Após o desenvolvimento do *software*, deu-se início ao processo de testes, para que seja realizada uma avaliação profunda dos resultados obtidos pelo *software* e pelo classificador construído. O objetivo desta etapa é validar se a aplicação atingiu um nível aceitável nas classificações que foram realizadas e, realizar uma comparação entre os métodos estatísticos, para assim ser possível analisar se algum método apresentava uma melhora considerável em relação aos demais métodos.

Para a analisar o desempenho do classificador construído neste projeto, foi utilizada a técnica de validação cruzada (BARELLA, 2016). Foi realizado o teste com os valores para treino e teste, respectivamente, 80% e 20%.

No método de validação cruzada, do tipo *k-fold* (*k-Fold Cross-Validation*), é um método capaz de utilizar os dados de forma mais eficiente, já que ele não reserva uma grande quantidade de informações para a validação. O processo envolve a partição aleatória de um espaço amostral em *k* subconjuntos de tamanhos equivalentes sendo que o modelo é estimado e validado *k* vezes. O grupo de treinamento é formado pela integração dos k-1 subconjuntos, desta forma o grupo de validação é, necessariamente, composto pelo subconjunto *k* não utilizado (Figura 26).

Figura 26: Figura exemplificativa da técnica de Validação Cruzada *k-Fold* com uma compartimentação aleatória do espaço amostral k = 6.



Fonte: Barella (2016).

Como explicado anteriormente, a ideia inicial era construir um algoritmo para classificar as letras, mas, o mesmo possuiu uma baixa acurácia, devido ao baixo número de imagens de teste, 1300, que comparado a base do eMnist que possui 697.932 imagens. E, devido a capacidade de processamento e a forma como o algoritmo foi elaborado, tornou-se inviável manter a classificação de letras. Com isso, a base foi atualizada para a abordagem de classificação de números manuscritos, utilizando a base de imagens do próprio Mnist, assim, utilizando o tamanho das imagens da base do Mnist, 28 *pixels* de altura por 28 *pixels* de largura.

3.4 DESENVOLVIMENTO DO *SOFTWARE* PARA OS USUÁRIOS

Foi desenvolvida uma interface gráfica utilizando o *JavaFX Scene Builder*, a qual o protótipo inicial é exibido na Figura 27. Nessa interface, o menu na parte superior possui dois submenus, um para carregar a imagem selecionando o diretório onde encontra-se a mesma e, o outro para exportar o texto (após ser classificado). Há também um botão de ajuda localizado no menu, que abre uma aba no navegador do usuário, essa aba contém um *site* simples

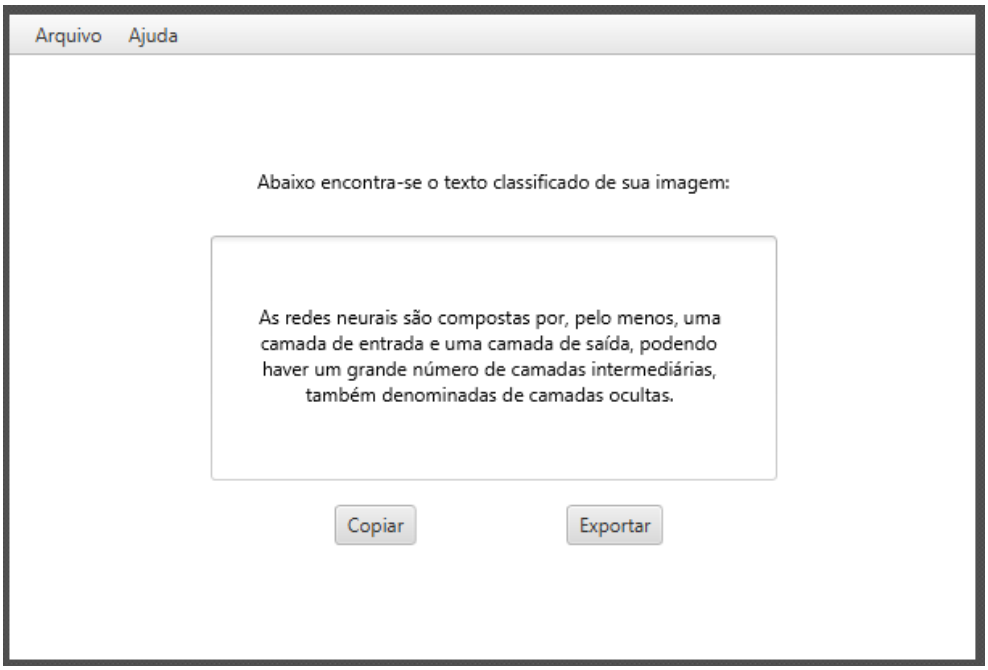
informando o passo a passo para utilizar o *software*. Além disso, no centro da tela principal, há uma mensagem instruindo o usuário a arrastar uma imagem até o local.

Figura 27: Protótipo de tela 1 do *software* do usuário.

Fonte: Elaborado pelo autor (2022).

A Figura 28 apresenta a concepção inicial da interface após o *software* realizar a classificação e identificação do texto contido na imagem. Nessa versão, o menu superior permanece similar ao da tela inicial *software* (Figura 27), porém, com algumas modificações, sendo elas: uma mensagem informando que a identificação foi realizada e que o texto se encontra abaixo, centralizado na tela do *software*. O usuário tem a opção de passar o *mouse* por cima do texto para selecioná-lo e copiá-lo ou, clicar no botão “Copiar”. Havendo também a possibilidade de exportar o texto gerado como um arquivo “*.txt*”, para isto, basta o usuário acionar o botão “Exportar”.

Figura 28: Protótipo de tela 2 do *software* do usuário.



Fonte: Elaborado pelo autor (2022).

3.5 MATERIAL UTILIZADO NO DESENVOLVIMENTO DO *SOFTWARE*

O ambiente de desenvolvimento utilizado foi o Eclipse 2023-06, a máquina utilizada para os testes do projeto é equipada com um processador AMD *Ryzen* 7 5700X, 32GB de memória RAM, DDR4 3200MHz e um SSD NVMe M.2 2280 PCIe com taxas de leitura de 7300MB/s e gravação de 6000MB/s, complementada por uma placa de vídeo NVIDIA RTX 3060 *Ti*. Além disso, o programa *Scene Builder* foi empregado para a prototipação da interface gráfica. O projeto foi desenvolvido utilizando a linguagem de programação Java na versão 8, juntamente com o *framework* JavaFX.

4 ANÁLISE DE RESULTADOS

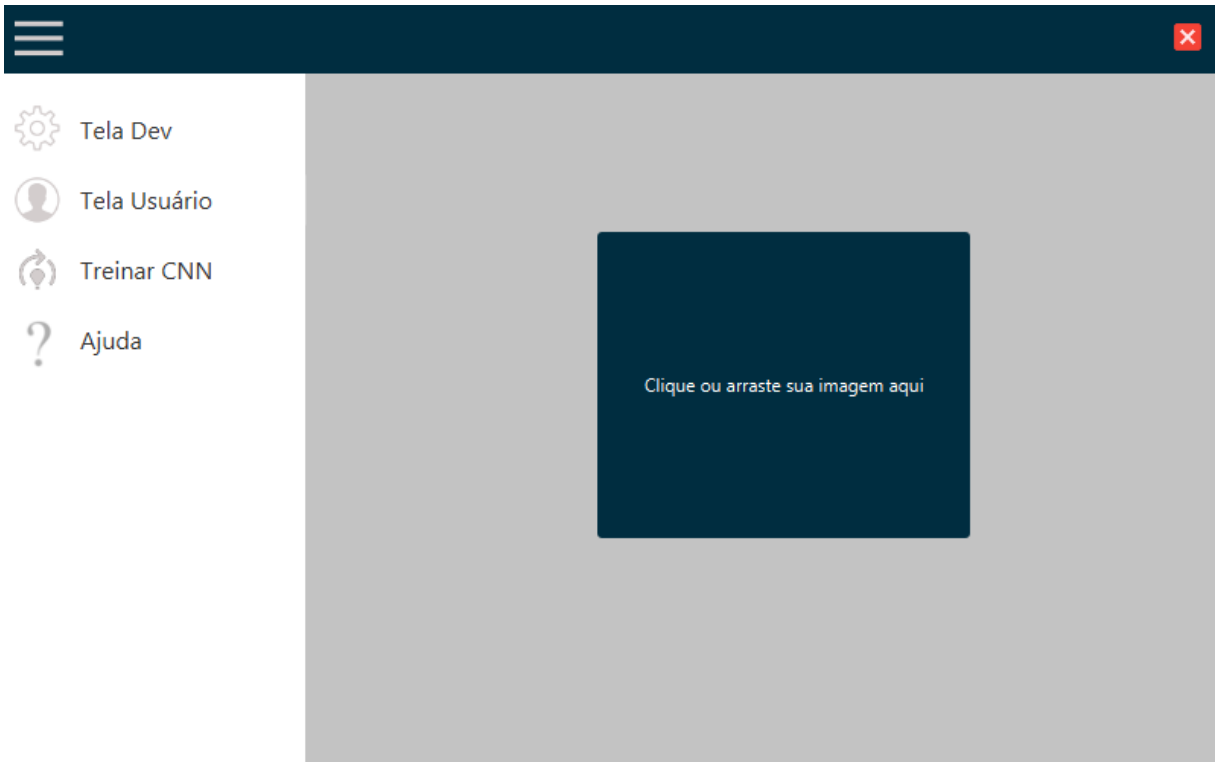
4.1 INTERFACE DO USUÁRIO

Após passado a fase inicial, algumas alterações significativas foram implementadas nas telas do usuário: agora, uma barra lateral expansível exibe quatro opções distintas, representando o *software* de desenvolvimento e o *software* do usuário (Figura 29), ambos eram apresentados juntos, visto que o *software* do usuário é o *software* de desenvolvimento sem duas abas. Ao clicar na interface do usuário (ícone da silhueta de uma pessoa), surge um botão na tela, orientando ao usuário a selecionar ou arrastar a imagem desejada (Figura 30).

Figura 29: Interface principal após alterações.

Fonte: Elaborado pelo autor (2023).

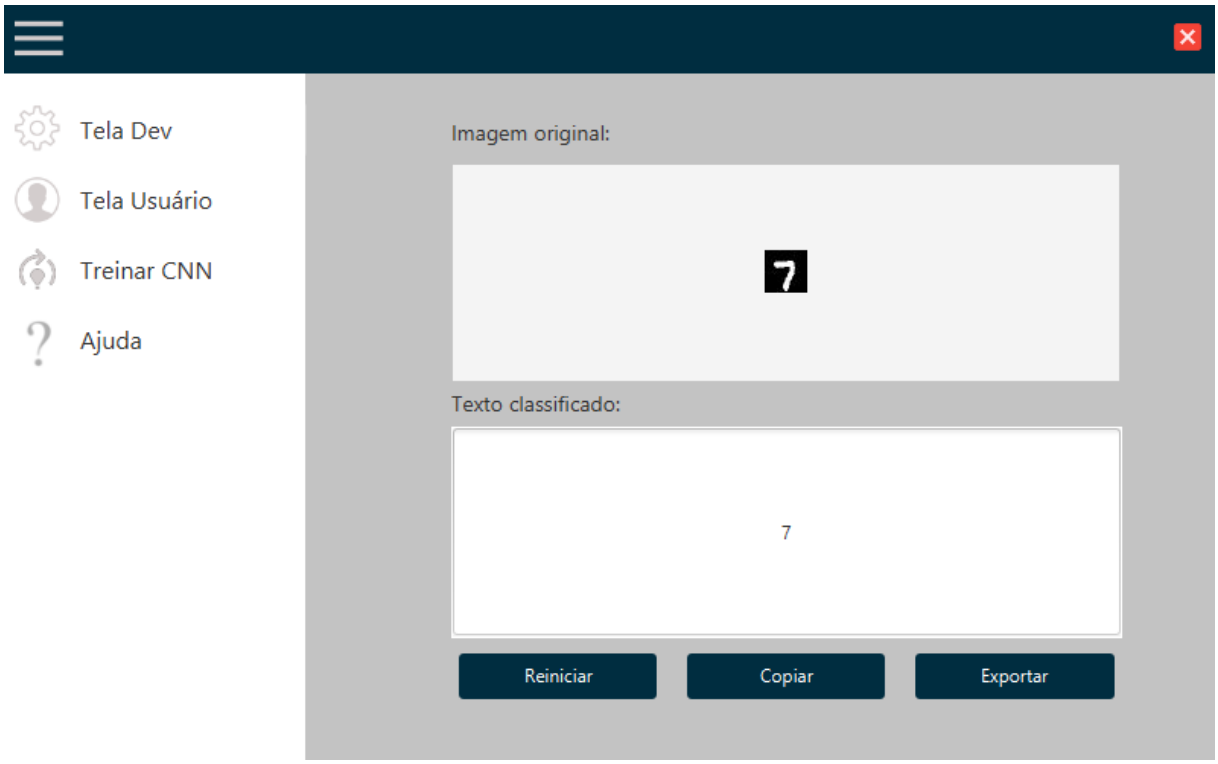
Figura 30: Interface principal após selecionar a “Tela Usuário”.



Fonte: Elaborado pelo autor (2023).

Uma vez que a imagem é carregada, ela é exibida na tela seguinte. Assim que o processamento de identificação dos números é concluído, ele é apresentado juntamente com a imagem (Figura 31), ou seja, na mesma interface. Além das funcionalidades anteriores que foram mantidas, uma nova foi adicionada: o botão “Reiniciar”, que permite que o usuário retorne à interface de seleção de imagem (Figura 30).

Figura 31: Interface principal após finalizar o processamento da imagem.



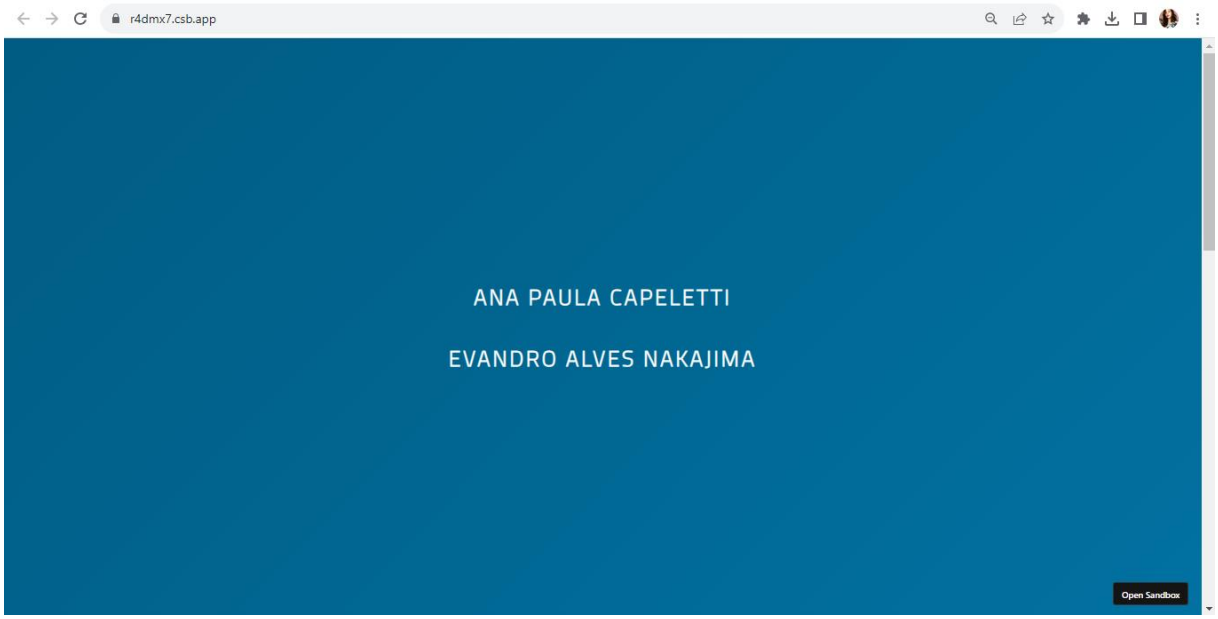
Fonte: Elaborado pelo autor (2023).

Levando em consideração os protótipos do projeto, onde havia a opção do botão de "ajuda" no menu principal para auxiliar nas dúvidas dos usuários, foi desenvolvido um site simples, utilizando HTML 5 e CSS 3, hospedado em uma plataforma de desenvolvimento em nuvem chamada CodeSandBox. O site tem como intuito o mesmo do proposto no protótipo, auxiliar os usuários a entender como funciona o *software*. A opção de ajuda pode ser observada na Figura 32, ao clicar sobre a opção, o navegador padrão do usuário é aberto com o site apresentado na Figura 33 e na Figura 34.

Figura 32: Interface principal com a opção de ‘ajuda’ em destaque.

Fonte: Elaborado pelo autor (2023).

Figura 33: Cabeçalho do site de ajuda para o usuário.



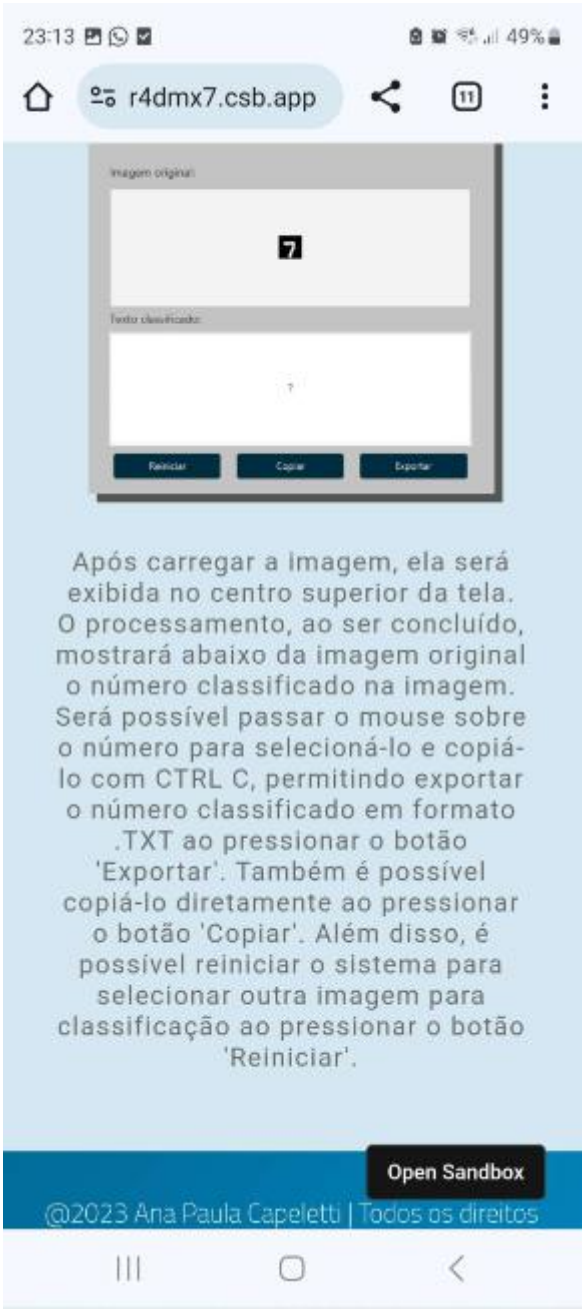
Fonte: Elaborado pelo autor (2023).

Figura 34: Site de ajuda ao usuário.

Fonte: Elaborado pelo autor (2023).

Vale ressaltar que o *site* possui um gradiente de cores em tons azuis que fica trocando em seu cabeçalho e em seu *footer*, além de possuir uma opção de bordas nas imagens ao passar o mouse sobre as figuras (como pode ser visto na Figura 34) e, ser responsivo, como é possível verificar na Figura 35.

Figura 35: *Site* de ajuda ao usuário em uma tela *mobile*.



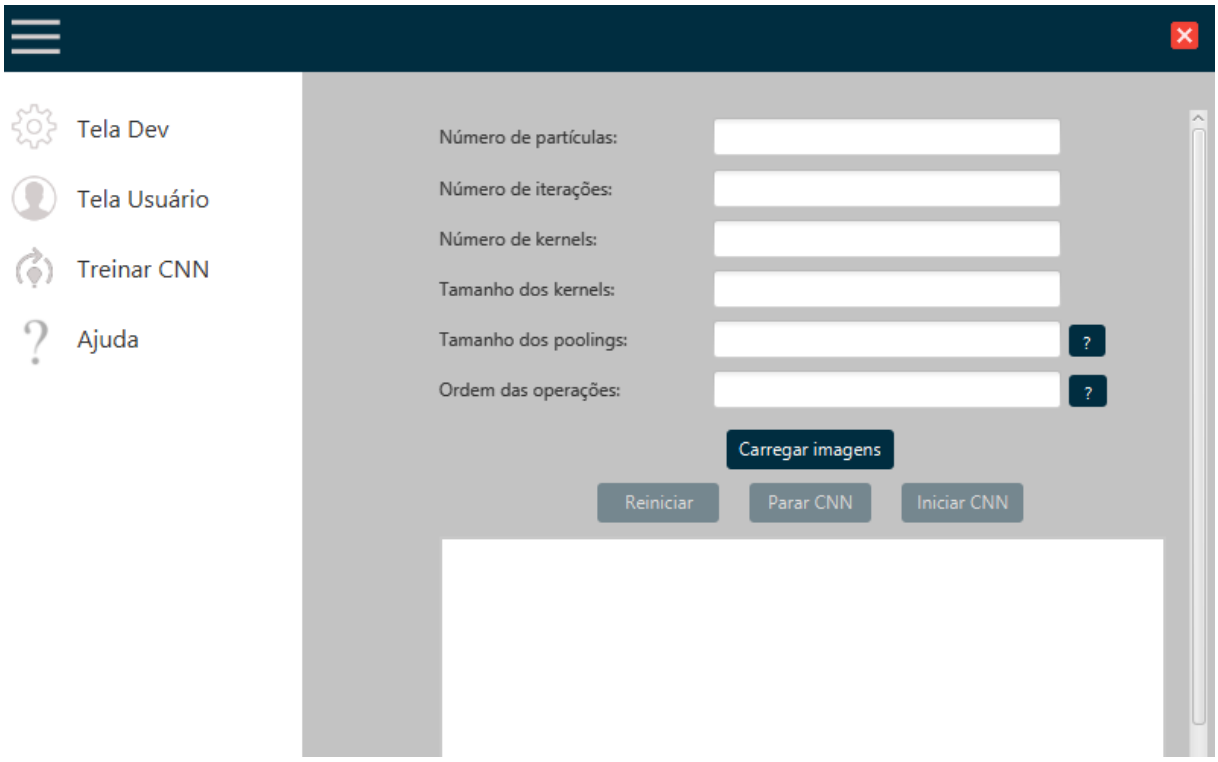
Fonte: Elaborado pelo autor (2023).

4.2 INTERFACE DO DESENVOLVEDOR

Uma das interfaces do *software* do desenvolvedor é a denominada ‘Treina CNN’, apresentada na Figura 36. Esta interface requer informações específicas para conduzir a etapa

de treinamento da RNA, utilizando imagens (as quais são carregadas mediante o botão "carregar imagens"). Após carregar as imagens os demais botões são habilitados para utilização. O botão "reiniciar" tem a função de apagar todas as informações inseridas e preparar a interface para a entrada de novos dados e imagens. Já o botão "parar CNN" interrompe o processo de classificação das imagens de entrada, enquanto o botão "iniciar CNN" dá início ao procedimento de classificação das imagens, exibindo seus resultados no espaço delimitado abaixo dos botões. Tais resultados contemplam informações como o número da iteração atual, a melhor partícula e o menor erro obtido durante o processo.

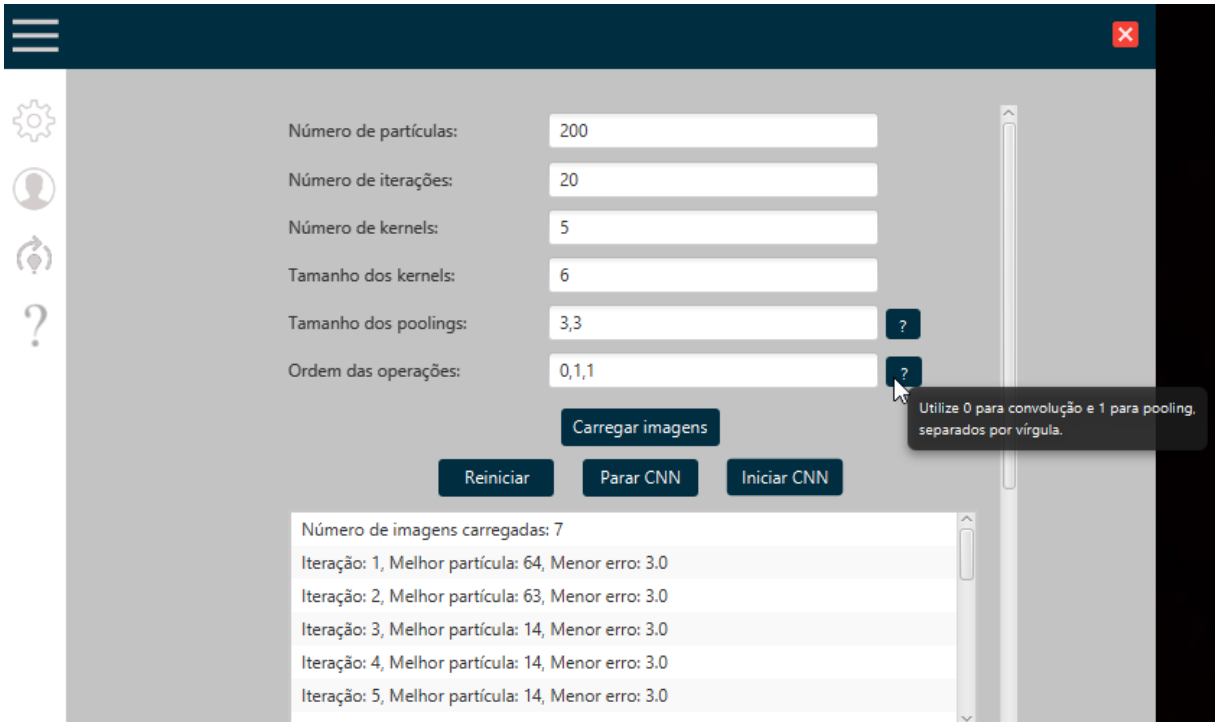
Figura 36: Interface do *software* do desenvolvedor, ‘treinar CNN’.



Fonte: Elaborado pelo autor (2023).

Vale ressaltar que as opções destacadas em azul com um ponto de interrogação nos campos “tamanho dos *poolings*” e “ordem das operações” correspondem a dicas interativas (*tooltips*) para orientar o usuário. Indicando que as informações devem ser inseridas separadas por vírgula. Além disso, no caso da "ordem das operações", o valor 0 refere-se à camada de convolução, enquanto o valor 1 corresponde à camada de *pooling*. Na Figura 37 é apresentado um exemplo das informações de retorno após a conclusão do processo de classificação.

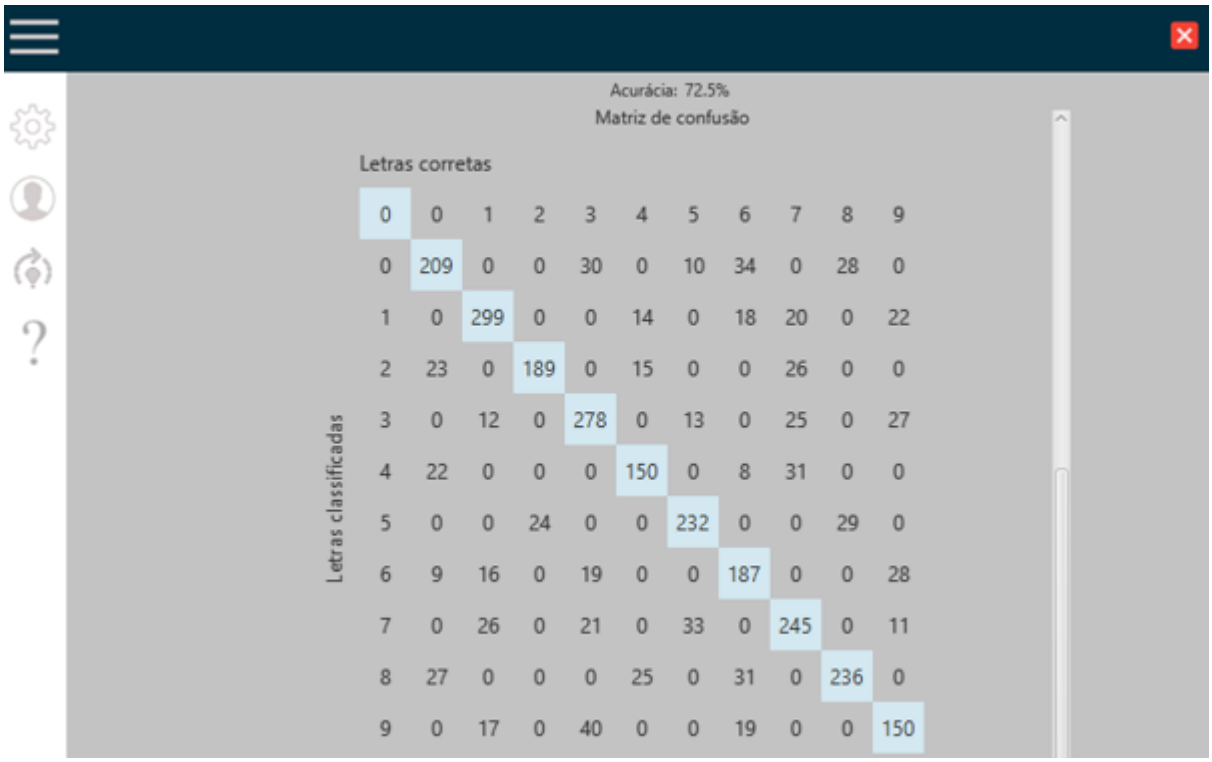
Figura 37: Exemplo das informações de retorno após finalizar o processo de classificação.



Fonte: Elaborado pelo autor (2023).

Após a conclusão do processo de classificação na interface "treinar CNN", os resultados podem ser analisados por meio de uma matriz de confusão, imediatamente disponibilizada após a exibição dos resultados. Essa matriz apresenta os números que foram classificados corretamente (diagonal principal) e os que não foram. Um exemplo dessa matriz encontra-se na Figura 38. A tela de "treinar CNN" emprega matrizes de *kernels* predefinidas para produzir os resultados, baseando-se nas informações especificadas nos campos apresentados na Figura 37, bem como nas imagens de entrada.

Figura 38: Interface do desenvolvedor, mostrando um exemplo de matriz de confusão.



Fonte: Elaborado pelo autor (2023).

O dígito “1” foi o melhor classificado possuindo, 74 Falsos Positivos (FP), 71 Falsos Negativos (FN), 299 Verdadeiros Positivos (TP) e 1876 Verdadeiros Negativos (TN). Onde, a sensibilidade ou taxa de verdadeiros positivos seria de: 0.8081, enquanto a especificidade ou taxa de verdadeiros negativos seria de: 0.9620, a acurácia ou taxa de casos corretamente classificados seria de: 0.9375 e a precisão ou taxa de positivos previsto é de: 0.8016.

A outra interface do software do desenvolvedor é denominada "tela dev", ilustrada na Figura 39. Essa interface realiza um *feedforward* para cada imagem de entrada, uma vez que os *kernels* são definidos e importados na entrada, juntamente com as imagens. Similar à interface "tela usuário", que possui *kernels* fixos, a "tela dev" requer informações específicas para conduzir a fase de treinamento da RNA. Essas informações incluem um arquivo no formato ".xlsx" (planilha *Excel*), contendo os melhores *kernels* da melhor partícula global, além do valor do *bias*, o qual se baseia no número de camadas de convolução a serem aplicadas e, por fim, contendo os valores da camada totalmente conectada. Também solicita o

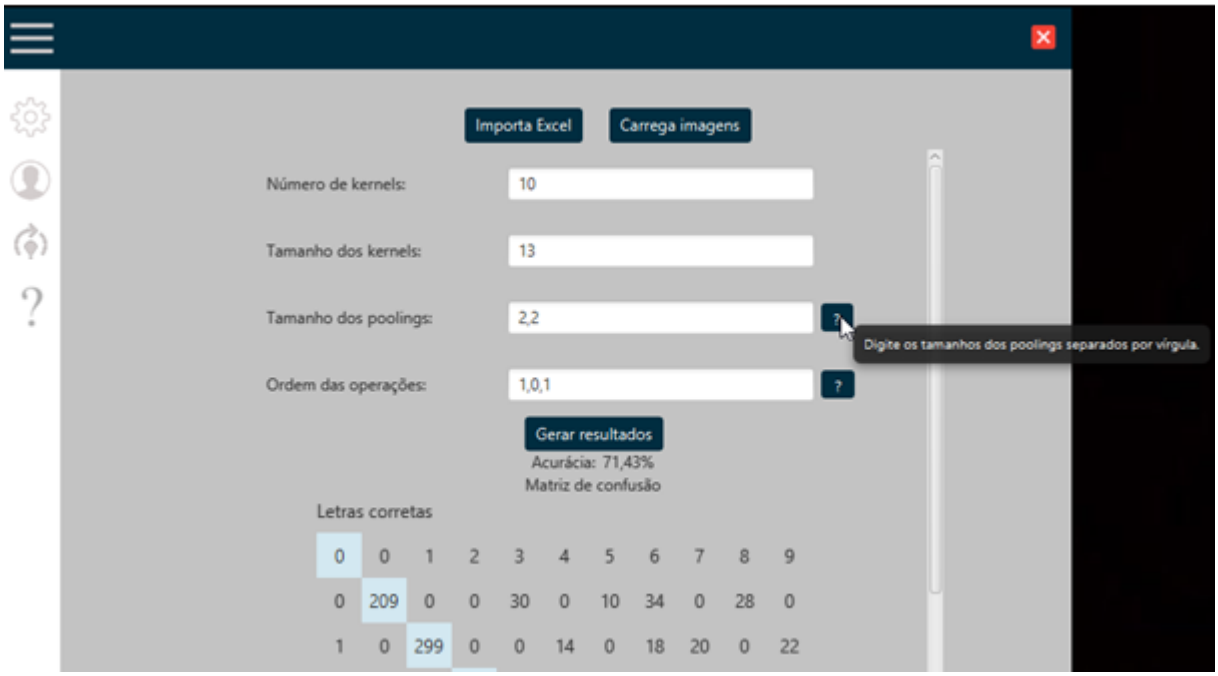
carregamento de determinadas imagens para fins de classificação, as quais são carregadas por meio do botão "carrega imagens". Após o carregamento das imagens, o botão para gerar os resultados torna-se habilitado para uso. Os resultados são exibidos abaixo do botão de “gerar resultados”, fornecendo informações como o valor da acurácia e uma matriz de confusão.

Figura 39: Interface do *software* do desenvolvedor, ‘Tela dev’.

Fonte: Elaborado pelo autor (2023).

Assim como na interface "treina CNN", as opções destacadas em azul com um ponto de interrogação nos campos "tamanho dos *poolings*" e "ordem das operações" na interface "tela dev" são representadas como dicas interativas (*tooltips*) para orientar o usuário. Elas fornecem as mesmas orientações presentes na outra tela. Na Figura 40 é exibido um exemplo das informações de retorno após a conclusão do processo de classificação. Já na Figura 41, é apresentada a matriz de confusão completa, sendo esta inicialmente mostrada na Figura 40.

Figura 40: Exemplo das informações de retorno após gerar os resultados.



Fonte: Elaborado pelo autor (2023).

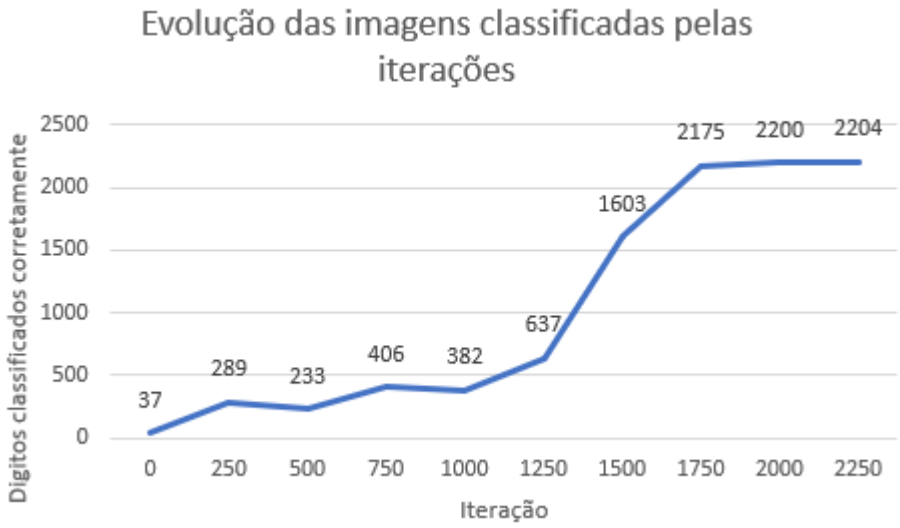
Figura 41: Matriz de confusão completa.

Fonte: Elaborado pelo autor (2023).

4.3 TREINAMENTO DA CNN

A Figura 42 ilustra a evolução dos dígitos classificados corretamente nas imagens, em relação ao número de iterações. Observa-se que, aproximadamente em 1750 iterações, atinge-se a marca de 2175 dígitos classificados corretamente, representando uma acurácia de 72.5%. É perceptível que, após esse ponto, os resultados se estabilizam, mantendo a taxa de acurácia em 72.5%. O gráfico foi elaborado nas seguintes condições: 100 partículas, 3000 imagens, a rede neural utilizada para gerar os resultados foi a do PSO, possuindo: 1 camadas de *pooling* máximo (com tamanho 2x2), uma camada de convolução (com 10 *kernels*, com tamanho 13x13) e outra camada de *pooling* máximo (com tamanho 2x2) e, por fim, a camada totalmente conectada.

Figura 42: Evolução dos dígitos classificados nas imagens baseado no número de iterações.



Fonte: Elaborado pelo autor (2023).

4.4 CROSS VALIDATION

Conforme informado anteriormente, para analisar o desempenho do classificador construído para este projeto, foi utilizada uma técnica de testes, que também foi utilizada no trabalho de Barella (2016), que é o método de validação cruzada ou *cross validation*. Os

resultados apresentados na Figura 43 possuem as seguintes condições: K = 6, onde a base de imagens era composta por 100 imagens no total e, dessas, 80% (8 imagens de cada dígito) foram selecionadas para o treino e 20% (2 imagens de cada dígito) para teste. Visto que, a base de imagens utilizada é pequena resultados abaixo de 70% eram esperados.

Figura 43: Resultados do *Cross Validation*.

-	Acurácia	Melhor dígito	Pior dígito
K(1)	45%	9, 6 e 3 (2/2)	0, 2, 5 e 8 (0/2)
K(2)	50%	5 e 8 (2/2)	3 e 7 (0/2)
K(3)	55%	1, 3 e 6 (2/2)	5 e 9 (0/2)
K(4)	55%	0, 4 e 9 (2/2)	2 e 7 (0/2)
K(5)	40%	0, 2 e 3 (2/2)	4, 5, 6, 7 e 8 (2/2)

Fonte: Elaborado pelo autor (2023).

5 CONCLUSÃO

O objetivo deste trabalho foi desenvolver um sistema para reconhecer palavras em imagens, através de uma rede neural, com o propósito de otimizar o tempo gasto. Além disso, foram avaliados diferentes métodos no próprio desenvolvimento da rede neural utilizada, comparando-se os resultados obtidos.

É relevante mencionar que os valores dos *kernels*, *bias* e os valores da camada totalmente conectada, foram gerados por meio do comando '*random.nextGaussian()*', o qual produz números aleatórios com média 0 e desvio padrão 1. Durante os testes, o algoritmo também foi executado utilizando o método do gradiente descendente. Os resultados obtidos, quando comparados à utilização do PSO, foram semelhantes, embora o PSO tenha demonstrado um desempenho ligeiramente superior, alcançando uma acurácia de 72.5%, conforme evidenciado na Figura 42.

É válido ressaltar também que esses testes foram conduzidos em conjuntos de imagens em pequena escala. O PSO mostrou limitações quando aplicado a conjuntos de imagens maiores. Uma possível solução para lidar com essa questão seria a melhoria do *hardware* da

máquina de testes, por meio da adição de uma GPU para processamento. Este fator foi identificado como um dos principais obstáculos que inviabilizaram a classificação de letras.

Pode-se concluir, através dos dados exibidos na Figura 42, o algoritmo PSO alcançou uma taxa de acurácia de 72.5%, após 1750 iterações, no processo de classificação. Os resultados deste estudo indicam a viabilidade do método PSO como um algoritmo de aprendizagem de redes neurais, com potencial aplicação em várias áreas.

Sugere-se como trabalho futuro o desenvolvimento de um sistema integrado que combine as funcionalidades de identificação de dígitos em imagens para sugestão de palavras-chave e a capacidade de identificar textos em imagens, oferecendo também a tradução instantânea para outros idiomas. Tal sistema poderia aprimorar a forma como lidamos com o conteúdo visual, facilitando a categorização e compreensão de imagens em diferentes línguas, ampliando significativamente sua acessibilidade e utilidade global.

REFERÊNCIAS

ADENIYI, P. O. Digital analysis of multitemporal Landsat data for land-use/landcover classification in semi-arid area of Nigeria. **Photogrammetric Engineering and Remote Sensing**, v.51, n. 1l, p. 1761-1774, 1985.

AGUIAR, F. G. **Utilização de Redes Neurais Artificiais para detecção de padrões de vazamento em dutos**. 2010. Dissertação (Mestrado em Engenharia Mecânica) - Universidade de São Paulo, São Carlos, p.95. 2010. Disponível em: < https://www.teses.usp.br/teses/disponiveis/18/18147/tde-17012011-160008/publico/Dissertacao_Revisada.pdf>. Acesso em: 16 de ago. de 2022.

ALBUQUERQUE, M. P. A.; CANER, E. S.; MELLO, A. G.; ALBUQUERQUE, M. P. Análise de Imagens e Visão Computacional. In: V Escola do Centro Brasileiro de Pesquisas Físicas. 1. 2012. Rio de Janeiro. **Anais...**, Rio de Janeiro: CBPF, 2012. p.145- 176. Disponível em: <https://mesonpi.cat.cbpf.br/e2012/arquivos/g06/CursoE2012-PI.pdf>. Acesso em: 31 de jul. 2022.

ALMEIDA, A. P. C. R. A.; NAKAJIMA, E. A. Enxame de partículas aplicado a uma rede neural como backpropagation para otimização de tempo de jogo. **Revista Brasileira de Iniciação Científica**, v. 8, p.21050-21050, 2021.

ARYA, S.; SINGH, R. A Comparative Study of CNN and AlexNet for Detection of Disease in Potato and Mango leaf. In: International conference on issues and challenges in intelligent computing techniques (ICICT), 2019. IEEE. **Anais...**, IEEE: 2019. p.1-6. Disponível em: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8977648>. Acesso em: 19 jul. 2022.

ÁVILA, E. A.; FARIAS, P. L. Letreiramentos da Liberdade: um método para identificação de caracteres em inscrições japonesas no espaço urbano. In: 4° SPDesign – Seminário de Pesquisas do Programa de Pós-graduação em Design da FAU USP, 2021. São Paulo. **Anais...**, São Paulo: Blucher, 2021. Disponível em: <http://pdf.blucher.com.br.s3-sa-east-1.amazonaws.com/designproceedings/4spdesign/4spdesign_03.pdf>. Acesso em: 06 ago. 2022.

BANT. **Filtragem no domínio da frequência**. 2014. Disponível em: < https://www.facom.ufu.br/~backes/gsi058/Aula07-FiltragemFrequencia.pdf >. Acesso em 20 nov. 2023.

BARELLA, C. F. **Abordagens estatísticas aplicadas ao mapeamento de susceptibilidade a movimentos de massa: Análise de diferentes técnicas no contexto do quadrilátero ferrífero**. 2016. Tese (Doutorado em Geotecnia) – Núcleo de Geotecnia, Universidade Federal de Ouro Preto, Ouro Preto, 2016. Disponível em: <https://www.repositorio.ufop.br/handle/123456789/6519>. Acesso em: 12 ago. 2022.

BAXES, Gregory A. **Digital Image Processing: Principals and Applications**. New York: John Wiley & Sons, 1994.

BIANCHI, M. F. **Extração de características de imagens de faces humanas através de wavelets, PCA e IMPCA**. 2006. Dissertação (Mestrado em Engenharia Elétrica) - Universidade de São Paulo. São Carlos, 2006. Disponível em: <https://www.teses.usp.br/teses/disponiveis/18/18133/tde-10072006-002119/en.php>. Acesso em: 03 set. 2022.

BORTH, M. R.; LACIA, J. C.; PISTORI, H.; RUVIANO, C. F. A visão computacional no agronegócio: Aplicações e direcionamentos. **Encontro Científico de Administração, Economia e Contabilidade (ECAECO)**, 7, out. 2014. Disponível em: <http://www.gpec.ucdb.br/pistori/publicacoes/borth_ecaeco2014.pdf>. Acesso em: 31 ago. 2022.

CARVALHO, M. M. Identificação de caracteres apresentados a uma câmera por meio do movimento dos dedos da mão de um ser humano. In: Congresso Nacional de Iniciação Científica. 13., 2013, Campinas, SP. **Anais…**, Campinas: Instituto Federal de Educação, ciência e tecnologia do Espírito Santo. Disponível em: <https://www.conic-semesp.org.br/anais/files/2013/trabalho-1000016127.pdf>. Acesso em: 26 ago. 2022.

CASTRO, C. L.; BRAGA, A. P. Supervised learning with imbalanced data sets: an overview. **Sba**: Controle & Automação Sociedade Brasileira de Automática, Campinas, SP, n. 5, p. 441, 2011.

CAVANI, F. A. **Análise de cenas de pomares de laranjeiras através de segmentação de imagens e reconhecimento de padrões**. 2007. Dissertação (Mestrado em engenharia mecânica) – Universidade de São Paulo. São Carlos, 2007. Disponível em: <https://www.teses.usp.br/teses/disponiveis/18/18145/tde-17012011-123819/publico/2007FelipeACavaniAnalisedecenas.pdf>. Acesso em: 15 set. 2022.

CUNHA, C. H. O. **Redes ReLU como modelos inteiros-mistos aplicadas à sintonia de simuladores de poços de petróleo**. 2022. Dissertação (Mestrado em Engenharia de

Automação e Sistemas) – Universidade Federal de Santa Catarina, Florianópolis, 2022. Disponível em: < https://repositorio.ufsc.br/bitstream/handle/123456789/240854/PEAS0411-D.pdf?sequence=-1&isAllowed=y>. Acesso em: 16 nov. 2022.

ELSAYED, G. F.; SHANKAR, S.; CHEUNG, B.; PAPERNOT, N.; KURAKIN, A.; GOODFELLOW, I. SOHL-DICKSTEIN, J. Adversarial Examples that Fool both Computer Vision and Time-Limited Humans. **ArXiv**, arXiv:1802.08195, maio 2018. Disponível em: <https://arxiv.org/pdf/1802.08195.pdf>. Acesso em: 27 jun. 2022.

FARHANI, N.; TERBEH, N.; ZRIGUI, M. Image to text conversion: state of the art and extended work. In: 2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA), 2017. IEEE. **Anais...**, IEEE: 2017. p.937-943. Disponível em: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8308390>. Acesso em: 10 jun. 2022.

FAWCETT, T. An introduction to ROC analysis. **Pattern recognition letters**, v. 27, n. 8, p. 861-874, 2006.

FENG, Y.; LAPATA, M. How many words is a picture worth? automatic caption generation for news images. In: Proceedings of the 48th annual meeting of the Association for Computational Linguistics, 2010. Uppsala. **Anais...**, Uppsala: 2010. p.1239-1249. Disponível em: <https://aclanthology.org/P10-1126.pdf>. Acesso em: 24 jun. 2022.

FILHO, O. M.; NETO, H. V. **Processamento Digital de Imagens**, Rio de Janeiro: Brasport, 1999. ISBN 8574520098.

FRANCO, J. R. **Método computacional para identificação do fungo Cercospora Kikuchii em sementes de soja**. 2017. Dissertação (Mestrado em Computação Aplicada) - Universidade Estadual de Ponta Grossa. Ponta Grossa, 2017. Disponível em: <https://tede2.uepg.br/jspui/handle/prefix/146>. Acesso em: 03 jun. 2022.

GODOI, M. A. D. **Aplicativo para identificar ocorrência de doença na soja por meio de reconhecimento de imagem e visão computacional.** 2019. Trabalho de Conclusão de Curso (Graduação em Ciências da Computação) - Faculdade Meridional – IMED, Rio Grande do Sul, [*s. l.*] 2019. Disponível em: <https://www.imed.edu.br/Uploads/APLICATIVO%20PARA%20IDENTIFICAR.pdf>. Acesso em: 27 ago. 2022.

GHARAT, S. **What, Why and Which?? Activation Functions**. 2019. Disponível em: <https://medium.com/@snaily16/what-why-and-which-activation-functions-b2bf748c0441>.

Acesso em: 30 out. de 2022.

GOMES, O. F. M. **Processamento e análise de imagens aplicados à caracterização automática de materiais**. 2001. Dissertação (Mestrado em Ciências da Engenharia Metalúrgica) - Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2001. Disponível em: < https://www.researchgate.net/publication/337963794_Processamento_e_Analise_de_Imagens_Aplicados_a_Caracterizacao_Automatica_de_Materiais >. Acesso em: 03 ago. 2022.

GOMES, O. F. M. **Microscopia Co-Localizada: Novas Possibilidades na Caracterização de Minérios**. 2007. Tese (Doutorado em Engenharia Metalúrgica e de Materiais) - Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2007. Disponível em: <https://doi.org/10.17771/PUCRio.acad.11498>. Acesso em: 02 set. 2022.

GU, J.; WANG, Z.; KUEN, J.; MA, L.; SHAHROUDY, A.; SHUAI, B.; LIU, T.; WANG, X.; WANG, L.; WANG, G.; CAI, K; CHEN, T. Recent advances in convolutional neural networks. **Pattern recognition**, v. 77, p.354-377, 2018. Disponível em: <https://arxiv.org/pdf/1512.07108.pdf >. Acesso em: 08 jul. 2022.

HAYKIN, S. **Redes Neurais**: princípios e prática. Porto Alegre: Bookman Editora, 2007.

HUBEL, D. H.; WIESEL, T. N. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. **The Journal of physiology**, [*s. l.*], v. 160, n. 1, p.106, 1962. Disponível em: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1359523/pdf/jphysiol01247-0121.pdf >. Acesso em: 04 jul. 2022.

KEEP, G. **Google Keep**. 2022. Disponível em: <https://www.google.com.br/keep/>. Acesso em: 01 nov. 2022.

KENNEDY, J.; EBERHART, R. Particle swarm optimization. In: Proceedings of ICNN'95-international conference on neural networks, 1995. IEEE. **Anais...**, IEEE: 1995. p.1942-1948. Disponível em: <https://ieeexplore.ieee.org/document/488968>. Acesso em: 04 out. 2020.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. ImageNet classification with deep convolutional neural networks. **NIPS'12 Proceedings of the 25th International Conference on Neural Information Processing Systems**, [*s. l.*], v. 1, p. 1097-1105, 2012.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. ImageNet classification with deep convolutional neural networks. **Communications of the ACM**, [*s. l.*], v. 60, n. 6, p. 84-90,

2017. Disponível em: <https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf>. Acesso em: 02 out. 2022.

LECUN, Y.; BOTTOU, L.; BENGIO, Y.; HAFFNER, P. Gradient-based learning applied to document recognition. **Proceedings of the IEEE**, v. 86, n. 11, p. 2278-2324, 1998. Disponível em: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=726791>. Acesso em: 10 set. 2022.

LENS, G. **Google Lens**. 2022. Disponível em: <https://lens.google/intl/pt-BR/howlensworks/>. Acesso em: 01 nov. 2022.

LINS, R. A. S. **Sistema inteligente para o processamento de imagens digitais intrabucais oclusais**. 2015. Dissertação (Mestrado em Ciências) - Universidade Federal do Rio Grande do Norte, Natal, 2015. Disponível em: <https://repositorio.ufrn.br/bitstream/123456789/21042/1/RamonAugustoSousaLins_DISSERT.pdf>. Acesso em: 15 set. 2022.

LUQUE, A.; CARRASCO, A.; MARTÍN, A.; HERAS, A. L. The impact of class imbalance in classification performance metrics based on the binary confusion matrix. **Pattern Recognition**, [s. l.], v. 91, p. 216-231, 2019.

LUQUE, L.; SILVA, R R. **Builder e Composite: padrões para a sua caixa de ferramentas**. 2014. Disponível em: <https://www.researchgate.net/profile/Leandro-Luque/publication/261026285_Padroes_de_Projeto_Builder_e_Composite/links/0deec53306c5287805000000/Padroes-de-Projeto-Builder-e-Composite.pdf >. Acesso em: 07 out. 2022.

LULIO, L. C. **Técnicas de visão computacional aplicadas ao reconhecimento de cenas naturais e locomoção autônoma em robôs agrícolas móveis**. 2011. Dissertação (Mestrado em Engenharia Mecânica) –Universidade de São Paulo, São Carlos. 2011. Disponível em: <https://www.teses.usp.br/teses/disponiveis/18/18145/tde-28112011-233750/publico/LCLulio2011.pdf>. Acesso em: 01 ago. 2022.

MARENGONI, M.; STRINGHINI, D. Tutorial: Introdução à visão computacional usando opencv. **Revista de Informática Teórica e Aplicada**, v. 16, n. 1, p.125-160, 2009. Disponível em: <https://www.seer.ufrgs.br/rita/article/view/rita_v16_n1_p125/7289>. Acesso em: 04 ago. 2022.

MARINI, F.; WALCZAK, B. Particle swarm optimization (PSO). A tutorial. **Chemometrics and Intelligent Laboratory Systems**, v. 149, p.153-165, 2015. Acesso em: 07 out. 2020.

MARR, D. Analyzing natural images: A computational theory of texture vision. In: Cold Spring Harbor symposia on quantitative biology, 1975. Cold Spring Harbor Laboratory Press. **Anais...**, Cold Spring Harbor Laboratory Press: 1975. p.647-662. Disponível em: <https://dspace.mit.edu/bitstream/handle/1721.1/6235/AIM-334.pdf?sequence=2&isAllowed=y>. Acesso em: 28 jun. 2022.

MARTINIANO, A.; FERREIRA, R.P.; FERREIRA, A.; FERREIRA, A.; SASSI, R.J. Utilizando uma rede neural artificial para aproximação da função de evolução do sistema de Lorentz. **Revista Produção e Desenvolvimento**, v. 2, n. 1, p.26-38, 2016.

MILANO, D.; HONORATO, L. B. Visão computacional**. Universidade Estadual de Campinas-(Unicamp)**, São Paulo, 2010. Disponível em: <https://docplayer.com.br/3058305-Visao-computacional-danilo-de-milano-luciano-barrozo-honorato-unicamp-universidade-estadual-de-campinas-ft-faculdade-de-tecnologia.html>. Acesso em: 27 ago. 2022.

MIYAMOTO, H. K.; FERNANDES, U. P.; IDE, W. T. S. T. Projeto de futebol de robôs na categoria IEEE Very Small Size Soccer da Equipe GER. **IEEE**. 2015. Disponível em: <http://sistemaolimpo.org/midias/uploads/124d7cc5559f5b464bd61dfbf4090bd9.pdf>. Acesso em: 15 out. 2022.

MOHAMMAD, A. A.; SOHRAB, Z.; ALI, L.; ALI, E.; IOANNIS, C. Reservoir permeability prediction by neural networks combined with hybrid genetic algorithm and particle swarm optimization. **Geophysical Prospecting**, v. 61, n. 3, p.582-598, 2013.

NEVES, L. A. P.; NETO, H. V.; GONZAGA, A. **Avanços em visão computacional**. Curitiba: Omnipax, p.27, 2012. Disponível em: <http://omnipax.com.br/livros/2012/AVC/avc-livro.pdf>. Acesso em: 27 jul. 2022.

OCAZIONEZ, S. A. C. **Processamento no domínio da freqüência de sinais de eletroencefalografia coletados durante protocolo de estresse moderado**. 2009. Dissertação (Mestrado em Engenharia Elétrica) – Universidade de Brasília, Brasília. 2009. Disponível em: <https://repositorio.unb.br/bitstream/10482/4213/1/2009_SergioAndresCondeOcazionez.pdf>. Acesso em: 23 set. 2022.

OLIVEIRA, F. M. **Ferramentas de Pré-processamento de Imagens**. 2003. Trabalho de Conclusão de Curso (Bacharelado em Engenharia Elétrica) - Universidade de Brasília, Brasília. 2003. Disponível em: <https://bdm.unb.br/bitstream/10483/801/1/2003_Fabr%c3%adcioMendesdeOliveira.PDF>. Acesso em: 25 set. 2022.

OLUDARE, I. A.; AMAN, J.; ABIODUN, E. O.; KEMI, V. D.; NACHAAT, A. M.; HUMAIRA, A.State-of-the-artin artificial neural network applications: Asurvey. **Heliyon**, v. 4, n. 11, p. e00938, 2018.

POLAK, E., **Computational Methods in Optimization**: A Unified Approach. Cambridge: Academic Press, 1971.

RAGHUNANDAN, K. S.; KUMARA, C. B. M.; KUMAR, G. H.; SUNIL, C. Convolution Neural Network Based Deep Features for Text Recognition in Multi-Type Images. In: 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2018. IEEE. **Anais…**, IEEE: 2018. p.502-507. Disponível em: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8554890>. Acesso em: 25 set. 2022.

RICHARDS, J. A. **Remote sensing digital image analysis: an introduction**. Berlin: Springer-Verlag, 1986. p. 281.

RIZZO, I. V.; CANATO, R. L. C. Inteligência artificial: funções de ativação. **Prospectus**, v. 2, n. 2, p.51-65, 2020.

ROSA, R. **Introdução ao sensoriamento remoto**. Uberlândia: EDUFU, 1990.

RUMELHART D. E.; HINTON G. E.; WILLIANS R. J. Learinig representations by back propagation error. **Nature**, v. 323, nº. 9, p. 533-536, 1986. Acesso em: 22 out. 2022.

SCHWAAB, M.; BISCAIA. J. E. C.; MONTEIRO, J. L.; PINTO, J. C. Nonlinear parameter estimation through particle swarm optimization. **Chemical Engineering Science**, v. 63, n. 6, p. 1542-1552, 2008.

SGARBOSA, P.; VECHIO, G. H. D. V. Inteligência artificial e suas implicações: como os dispositivos inteligentes e assistentes virtuais influenciam o cotidiano das pessoas. **Revista Interface Tecnológica**, [S. l.], v. 17, n. 2, p. 193–205, 2020. DOI: 10.31510/infa.v17i2.936. Disponível em: https://revista.fatectq.edu.br/interfacetecnologica/article/view/936. Acesso em: 25 set. 2022.

SHARMA, S.; SHARMA, S. Activation functions in neural networks**. International Journal of Engineering Applied Sciences and Technology**, v. 4, n. 12, p. 310–316, 2020.

SILVA, A. M. M.; PATROCÍNIO, A. C.; SCHIABEL, H. Processamento e análise de

imagens médicas. **Revista Brasileira de Física Médica**, v. 13, n. 1, p.34-48, 2019. Disponível em: <https://doi.org/10.29384/rbfm.2019.v13.n1.p34-48>. Acesso em: 28 set. 2022.

SLOWIK, A.; BIALKO, M. Training of Artificial Neural Networks Using Differential Evolution Algorithm, **IEEE**, p. 60-65. 2008. Acesso em: 02 out. 2020.

SOUZA, J. A. **Reconhecimento de padrões usando indexação recursiva**. 1999. Tese (Doutorado em Engenharia de Produção) - Universidade Federal de Santa Catarina, Florianópolis, 1999. Disponível em: <https://repositorio.ufsc.br/bitstream/handle/123456789/80484/144135.pdf?sequence=1>. Acesso em: 03 jun. 2022.

SZELISKI, R. **Computer vision**: algorithms and applications. Ithaca: Springer Science & Business Media, 2010.

TANGERINO, D. F.; LOURENÇO, R. T. Comparação da exatidão de métodos de classificação supervisionada e não supervisionada a partir do índice kappa na microbacia do Ribeirão Duas Águas em Botucatu/SP. In: XVI Simpósio Brasileiro de Sensoriamento Remoto-SBSR, 2013. Foz do Iguaçu. **Anais…**, Foz do Iguaçu: 2013, p.4093-4100. Disponível em: <http://marte2.sid.inpe.br/col/dpi.inpe.br/marte2/2013/05.29.00.32.43/doc/p1193.pdf>. Acesso em: 31 jun. 2022.

TEIXEIRA, J. P.; BATISTA, J.; TOCA, A.; GONÇALVES, J.; PEREIRA, F. Reconhecimento de Caracteres com Rede Neuronal Artificial para o Reconhecimento de Caracteres com Interface Gráfica. Inovação e Desenvolvimento, In: 5.ª Conferência de Engenharia, 2009. Covilhã. **Anais…**, Covilhã: 2009, [6] p. Disponível em: <https://bibliotecadigital.ipb.pt/bitstream/10198/1864/1/Artigo2_final.pdf>. Acesso em: 23 out. 2022.

VOULODIMOS, A.; DOULAMIS, N.; DOULAMIS, A.; PROTOPAPADAKIS, E. Deep learning for computer vision: a brief review. **Computational intelligence and neuroscience**, v. 2018, n. 7068349, 2018. Disponível em: <https://doi.org/10.1155/2018/7068349>. Acesso em: 27 jun. 2022.

WIZBICKI, A. S.; BATTISTI, G. Reconhecimento de padrões em imagens aplicando visão computacional. **Revistas Eletrônicas Unijuí - Salão do Conhecimento**, Rio Grande do Sul, set. 2014. Disponível em:

<https://publicacoeseventos.unijui.edu.br/index.php/salaoconhecimento/article/view/3408/2813>. Acesso em: 27 jul. 2022.

WOS. **Trust the Difference**. Web of Science Fact Book, 2021. Disponível em: <https://www.webofknowledge.com>. Acesso em: 09 jul. 2022.

WOS. **Web of Science**. 2022. Disponível em: <https://www.webofknowledge.com/>. Acesso em 11 out. 2022.

YANG, Y.; Teo, C. L.; DAUMÉ III, H.; ALOIMONOS, Y. Corpus-guided sentence generation of natural images. In: Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing. 2011. Edimburgo. **Anais…**, Edimburgo: Association for Computational Linguistics, 2011. p.444-454. Disponível em: <https://aclanthology.org/D11-1041.pdf>. Acesso em: 21 out. 2022.

YE, Q.; DOERMANN, D. Text detection and recognition in imagery: A survey. **IEEE transactions on pattern analysis and machine intelligence**, v. 37, n. 7, p.1480-1500, 2014. Disponível em: <https://ieeexplore.ieee.org/ielaam/34/7116666/6945320-aam.pdf >. Acesso em: 31 out. 2022.

YILDIRIM, Melih. Analog circuit architecture for max and min pooling methods on image. **Analog Integrated Circuits and Signal Processing**, v. 108, n. 1, p.119-124, 2021.

ZHENYA, H.; CHENGJIAN, W.; LUXI, Y.; XIQI, G.; SUSU, Y. Extracting rules from fuzzy neural network by particle swarm optimisation, **IEEE**, p. 74-77. 1998. Disponível em: <https://ieeexplore.ieee.org/abstract/document/699325>. Acesso em: 09 jul. 2022.