

Identificação de componentes de esquemas elétricos através de técnicas de classificação de imagem

EMANUEL JORGE PEREIRA DA CUNHA

Junho de 2024

POLITÉCNICO DO PORTO
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

Identificação de componentes de esquemas elétricos através de técnicas de classificação de imagem

Emanuel Jorge Pereira da Cunha

Mestrado em Engenharia Electrotécnica e de Computadores
Área de Especialização em Automação e Sistemas



DEPARTAMENTO DE ENGENHARIA ELETROTÉCNICA
Instituto Superior de Engenharia do Porto

Junho, 2024

*Esta dissertação satisfaz, parcialmente, os requisitos que constam da Ficha de
Unidade Curricular de Tese/Dissertação, do 2º ano, do Mestrado em
Engenharia Electrotécnica e de Computadores, Área de Especialização em
Automação e Sistemas.*

Candidato: Emanuel Jorge Pereira da Cunha, N.º 1151134,
1151134@isep.ipp.pt

Orientação Científica: Mário Jorge de Andrade Ferreira Alves,
mjf@isep.ipp.pt

Coorientação Científica: Lino Manuel Cerqueira Sousa, sss@isep.ipp.pt



DEPARTAMENTO DE ENGENHARIA ELETROTÉCNICA
Instituto Superior de Engenharia do Porto
Rua Dr. António Bernardino de Almeida, 431, 4200-072 Porto

Junho, 2024

Agradecimentos

Em primeiro lugar, gostaria de agradecer à minha noiva pela motivação e apoio prestados durante os meses de desenvolvimento desta tese. Do mesmo modo, gostaria também de agradecer aos meus pais e ao meu avô cujo ânimo dado foi essencial para o desenvolvimento deste projeto.

Em segundo lugar, gostaria de agradecer ao meu orientador, o Professor Mário Alves, e ao meu coorientador, o Professor Lino Sousa, pela oportunidade de integração no grupo da *framework* U=RI solve. O apoio e ajuda destes foram essenciais à perseverança face às dificuldades encontradas durante o desenvolvimento desta tese. Neste seguimento, apresento também a minha gratidão ao professor André Rocha pelas apontamentos e propostas que me foi apresentando.

Agradeço também aos professores da escola profissional Cenatex, Paulo Silva e Filipe Martins, pois, para além dos conhecimentos transferidos durante o meu período de formação, apresentaram também a disponibilidade de me permitir recolher amostras de simbologia eletrónica pelas suas turmas. Consequentemente, apresento a minha gratidão à Diretora Pedagógica da escola profissional Cenatex, Dra. Arlete Milhão, pela receção, carinho e força que me deu no dia em que visitei a escola para efetuar a recolha junto aos alunos. Do mesmo modo, agradeço também aos alunos do Curso Técnico de Eletrónica, Automação e Instrumentação pela disponibilidade em preencher as tabelas de simbologia eletrónica.

Desta forma, gostaria de agradecer aos professores e alunos do primeiro ano de licenciatura do curso Engenharia Eletrotécnica e de Computadores, que durante as aulas práticas e as aulas de preparação para o exame de Teoria de Circuitos, se disponibilizaram a preencher as mesmas tabelas. De seguida, apresento o meu apreço a dois colegas que me acompanharam durante todo o mestrado, o Rúben Alves e o Pedro Lemos, e cuja entre-ajuda durante o mesmo foi essencial.

Por último, gostaria de agradecer ao Engenheiro Esaú Cardoso do Instituto Eletrotécnico Português, que após vários contactos sem sucesso, foi a pessoa que respondeu de forma clara, breve e concisa às minhas questões e que permitiu a validação de algumas pesquisas efetuadas.

Resumo

Um engenheiro eletrotécnico, principalmente ao longo do seu primeiro ano académico, é introduzido à análise de circuitos elétricos. Esta análise é iniciada com a apresentação aos componentes elétricos e eletrónicos e aos seus respetivos símbolos.

De seguida, após este primeiro contacto, o aluno começa a associar alguns símbolos e a formar os seus primeiros circuitos elétricos. Tradicionalmente, o aluno começa por desenhar o circuito em papel. Após isto, é feita a primeira análise e os cálculos necessários. A simulação do circuito é, por vezes, uma forma complementar de validar e analisar outros parâmetros, como a tensão e a corrente. A simulação é um passo importante na aprendizagem, pois possibilita a validação dos cálculos realizados em papel e flexibilidade a nível de projeto e pré-validação de circuitos elétricos e eletrónicos.

No estudo, e em forma de treino, os alunos analisam vários circuitos que, por vezes, não apresentam qualquer solução que permita validar os resultados. Por norma, são então introduzidas ferramentas de simulação de modo a obter a validação. Esta é uma tarefa árdua e que consome bastante tempo de estudo dos alunos.

De modo a ultrapassar esta dificuldade, foi criada a plataforma U=RI solve [1] que, num dos seus futuros modos de operação, permitirá ao aluno economizar o seu tempo na tarefa de transferir o circuito para o simulador. Para isto é utilizada uma fotografia ou digitalização do esquema do circuito para introduzir do circuito no simulador.

Primeiramente, a imagem é submetida num algoritmo de visão computacional. Este é capaz de interpretar a topologia de um esquema elétrico e construir o seu modelo a partir da análise dessa imagem. Este módulo de software, desenvolvido pelo colega Hugo Barbosa [2] no âmbito do seu projeto de tese, permite a deteção e extração dos elementos presentes no circuito. Estes são, por exemplo, componentes, os nós, as interligações entre elementos, os valores e os seus identificadores.

Assim sendo, os *outputs* deste módulo são um conjunto de ficheiros com a informação extraída da imagem. Nos excertos da imagem original, foram introduzidas pequenas caixas delimitadoras à volta dos elementos dos circuitos. Deste modo, estes excertos serão utilizados por outro módulo de software capaz de os classificar.

Consequentemente, este projeto/tese consistiu em criar um algoritmo de inteligência artificial que utiliza os outputs do módulo anterior e classifique cada elemento do circuito. Para isto, serão utilizadas técnicas de *Machine Learning*, como *Deep*

Learning, e redes neuronais. No entanto, devido à pouca oferta de conjuntos de imagem, ou seja, *datasets*, que preenchessem todos os requisitos propostos, foi necessária a recolha de amostras de símbolos para treinar o modelo de *Machine Learning*. Em suma, a finalidade deste modulo de *software* é o reconhecimento dos caracteres alfanuméricos contidos nos identificadores de cada componente do circuito. Deste modo, com todas estas partes interligadas integradas na *framework* U=RIolve, o aluno conseguirá validar os seus resultados utilizando esta ferramenta de auto-aprendizagem na análise de circuitos elétricos(segundo diferentes métodos).

Palavras-Chave: Análise circuitos, processamento imagem, machine learning, deep learning, redes neuronais, e-learning, classificação imagens, esquemas elétricos, U = RIolve

Abstract

An electrical engineering student, especially during their first academic year, is introduced to the analysis of electrical circuits. This analysis begins with the introduction to electrical and electronic components and their respective symbols.

Next, after this initial contact, the student starts to associate some symbols and form their first electrical circuits. Traditionally, the student begins by drawing the circuit on paper. After this, the first analysis and necessary calculations are made. Circuit simulation is sometimes used as a complementary way to validate and analyze other parameters, such as voltage and current.

Simulation is an important step in learning, as it allows for the validation of calculations done on paper and offers flexibility in project design and pre-validation of electrical and electronic circuits.

In the study, and as a form of practice, students analyze various circuits that sometimes do not provide any solution to validate the results. Normally, simulation tools are then introduced to obtain validation. This is a tedious task that consumes a lot of the students' study time.

To overcome this difficulty, the platform U=RI solve [1] was created, which, in one of its future modes of operation, will allow the student to save time on the task of transferring the circuit to the simulator. For this, a photograph or scan of the circuit diagram is used to input the circuit into the simulator.

First, the image is submitted to a computer vision algorithm. This algorithm can interpret the topology of an electrical diagram and build its model from the analysis of that image. This software module, developed by colleague Hugo Barbosa [2] as part of his thesis project, allows the detection and extraction of elements present in the circuit. These elements are, for example, components, nodes, interconnections between elements, values, and their identifiers.

Thus, the outputs of this module are a set of files with the information extracted from the image. In the excerpts of the original image, small bounding boxes were introduced around the circuit elements. In this way, these excerpts will be used by another software module capable of classifying them.

Consequently, this project/thesis consisted of creating an artificial intelligence algorithm that uses the outputs of the previous module and classifies each element of the circuit. For this, *Machine Learning* techniques, such as *Deep Learning*, and neural networks will be used. However, due to the lack of image sets, or datasets, that

met all the proposed requirements, it was necessary to collect samples of symbols to train the *Machine Learning* model.

In summary, the purpose of this software module is to recognize the alphanumeric characters contained in the identifiers of each circuit component. Thus, with all these interconnected parts integrated into the U=RI solve framework, the student will be able to validate their results using this self-learning tool in circuit analysis (according to different methods).

Keywords: Circuit analysis, image processing, machine learning, deep learning, neural networks, e-learning, image classification, electrical diagrams, U = RI solve

Índice

Lista de Figuras	vii
Lista de Tabelas	ix
Lista de Código	xi
Lista de Acrónimos	xiii
1 Introdução	1
1.1 Contextualização	1
1.2 Objetivos e Abordagem	2
1.3 Plano de Trabalho	2
1.4 Organização da Dissertação	3
2 Contexto Científico e Tecnológico	5
2.1 Processamento de Imagem	5
2.1.1 O que é uma Imagem Digital	5
2.1.2 Segmentação de Imagem	6
2.2 Aprendizagem Computacional	8
2.2.1 Machine Learning	9
Supervisionado	10
Não Supervisionado	18
Semi-Supervisionada	20
2.2.2 Deep learning	21
Como Treinar e Criar Modelos de Deep Learning	21
Aceleração de modelos de Deep Learning	22
2.3 Detecção de Objetos em Tempo Real	23
2.3.1 Yolo	23
2.3.2 Roboflow Universe	24
2.3.3 Edge AI	27
3 Simbologia de componentes elétricos e o "dataset"	29
3.1 Estudo de simbologia e normalização	29
3.2 Técnicas de classificação de símbolos de circuitos elétricos	31

3.3	Preparação e Construção do Dataset	36
3.4	Preparação do Dataset	36
4	Abordagem de implementação	39
4.1	Linguagens de Programação e Bibliotecas	39
4.1.1	Frameworks para Classificação de Imagem	41
	Tensor Flow	41
	Keras	41
	PyTorch	43
4.2	Arquitetura de software	44
4.2.1	Início do Programa	46
4.3	Treino do Modelo	47
4.4	Classificação de Componentes	52
4.5	Preenchimento do modelo de dados	52
5	Avaliação de resultados	55
5.1	Métrica de Avaliação	55
5.2	Resultados do Treino do Modelo	56
5.3	Resultados do Teste do Modelo	61
6	Conclusões	63
	Referências	66
	Anexo A Tabelas com classificações em todas as classes	75
	Anexo B Modelo de Dados JSON	85

Lista de Figuras

1.1	Calendarização	4
2.1	<i>Pixels</i> [5]	6
2.2	Limitação Binária	7
2.3	Deteção de Limites	7
2.4	Imagem dividida em regiões semelhantes[7]	8
2.5	Estrutura <i>Machine Learning</i> [10]	9
2.6	Algoritmo Inteligência Artificial[10]	10
2.7	Algoritmo KNN [15]	11
2.8	Técnica de redução de dimensionalidade[16]	12
2.9	Bagging[21]	15
2.10	Stacking[21]	16
2.11	Boosting[21]	16
2.12	Yolo V7[42] na deteção de componentes de um circuito elétrico	23
3.1	Amostra de símbolos da base de dados da IEC	31
3.2	Funcionamento do Método HOG	32
3.3	Demonstração de resultados do modelo R-CNN	34
3.4	Tabela Criada para Recolha de Símbolos	37
4.1	Estrutura Keras e Tensor Flow[63]	42
4.2	Funcionalidades e vantagens de usar Pytorch[67]	43
4.3	Fluxograma principal do modelo de ML	46
4.4	Operação convolução	49
4.5	Representação gráfica da função ReLU	49
4.6	Training	51
4.7	Fluxograma do módulo Predict	53
4.8	Fluxograma do preenchimento dos dados no modelo JSON	54
5.1	Demonstração de resultados com todas as classes	57
5.2	Demonstração de resultados com redução de classes	57
5.3	Demonstração de resultados com redução de classe e aumento de resolução	59
5.4	Demonstração de resultados com aumento de iterações	60

5.5	Demonstração resultados Teste	62
A.1	Demonstração resultados - parte 1	76
A.2	Demonstração resultados - parte 2	77
A.3	Demonstração resultados com redução classes IP:124; BS:64; Epochs:45	78
A.4	Demonstração resultados com redução classes IP:196; BS:20; Epochs:45	79
A.5	Demonstração resultados com redução classes IP:64; BS:128; Epochs:200	80
A.6	Demonstração resultados Teste - parte 1 IP:64; BS:32; Epochs:200 .	81
A.7	Demonstração resultados Teste - parte 2 IP:64; BS:32; Epochs:200 .	82
A.8	Conjunto de imagens-teste	83

Lista de Tabelas

2.1	Resumo categorias e classes RF100	26
2.2	Resultados Experimentais RF100	26
5.1	Matriz de Confusão	58
5.2	Matriz de confusão com redução de classes	59
5.3	Matriz de confusão com redução de classes e aumento de resolução	60
5.4	Matriz de confusão com aumento de iterações	61
5.5	Matriz de confusão com imagens de teste	61

Lista de Código

4.1	Rotina de validação dos rótulos de cada imagem no conjunto de dados	41
4.2	Rotina de Treino usando Keras	42
4.3	Rotina de Treino usando Pytorch	43
4.4	Função seleção de modo no programa principal	46
4.5	Função onde é construído o modelo da rede neuronal	48
B.1	Exemplo Estrutura de dados do Modelo	85

Lista de Acrónimos

AC	<i>Alternate Current</i>
AP	<i>Average Precision</i>
BMP	<i>Basic Metabolic Panel</i>
CEN	<i>European Committee for Standardization</i>
CENELEC	<i>European Committee for Electrotechnical Standardization</i>
CNN	<i>Convolutional Neural Networks</i>
COCO Microsoft	<i>Common Objects in Context (Microsoft)</i>
CPU	<i>Central Processing Unit</i>
CSV	<i>Comma Separated Values</i>
CUDA	<i>Compute Unified Device Architecture</i>
DC	<i>Direct Current</i>
DL	<i>Deep Learning</i>
Fast R-CNN	<i>Faster Regions with Convolutional Neural Networks</i>
GLM	<i>Generalized Linear Model</i>
GND	<i>Ground (Electrical)</i>
GPU	<i>Graphics Processing Unit</i>
HMM	<i>Hidden Markov Model</i>
HOG	<i>Histogram of Oriented Gradients</i>
IA	<i>Inteligência Artificial</i>
IEC	<i>International Electrotechnical Commission</i>
IEP	<i>Instituto Eletrotécnico Português</i>
IoT	<i>Internet of Things</i>

IoU	<i>Intersection over Union</i>
IPQ	<i>Instituto Português da Qualidade</i>
JPEG	<i>Joint Photographic Expert Group</i>
JPG	<i>Joint Photographic Group</i>
K-NN	<i>K-Nearest Neighbors</i>
LDA	<i>Linear Discriminant Analysis</i>
mAP	<i>mean Average Precision</i>
ML	<i>Machine Learning</i>
NLL	<i>Negative Log-Likelihood</i>
NN	<i>Neural Network</i>
PIL/PILLOW	<i>Python Imaging Library</i>
ReLU	<i>Rectified Linear Unit</i>
RGB	<i>Red Green Blue</i>
SGD	<i>Stochastic Gradient Descent</i>
SVM	<i>Support Vector Machine</i>
TIFF	<i>Tag Image File Format</i>
XML	<i>Extensible Markup Language</i>
YOLO	<i>You Only Look Once</i>

Capítulo 1

Introdução

O trabalho proposto para esta Tese engloba o estudo e implementação de algoritmos de classificação de imagem, utilizando inteligência artificial, com o intuito de analisar a aplicabilidade e eficácia de cada um deles no contexto da identificação de componentes de circuitos elétricos. Deste modo, este estudo dá continuidade ao trabalho que tem sido desenvolvido na *framework* U=RIolve [1], designadamente no âmbito da interpretação e modelação de circuitos elétricos.

1.1 Contextualização

A análise de circuitos elétricos é, de certa forma, a base de formação de um engenheiro eletrotécnico. Por norma, o aluno começa por desenhar um circuito em papel e aplicar metodologias de análise que caracterizam matematicamente o circuito, e permitem obter as correntes e tensões do mesmo, como por exemplo, o *Método das Correntes nas Malhas* e o *Método das Tensões nos Nós* [3].

A U=RIolve é uma ferramenta de ensino e auto-aprendizagem que auxilia os alunos na implementação destes métodos de análise de circuitos elétricos, disponibilizando resoluções passo-a-passo através de uma interface web simples e intuitiva.

Assim sendo, o projeto apresentado tem como objetivo criar um algoritmo de inteligência artificial utilizando os outputs de outro módulo desenvolvido anteriormente [2]. Nesse módulo, é introduzida uma imagem, fotografia ou digitalização de um circuito elétrico desenhado à mão ou num computador. De seguida, essa mesma imagem é submetida a um processo de segmentação por visão computacional, onde

serão detetados e extraídos os diversos elementos presentes no circuito, como componentes, nós, interligações entre os elementos, valores e os seus identificadores.

Em suma, o projeto desenvolvido terá como função a classificação de cada elemento do circuito extraído por esse módulo anterior [2]. Nesta classificação, serão utilizadas técnicas de *Machine Learning*, como *Deep Learning* e redes neuronais, de modo a classificar a classe a que pertence cada elemento.

1.2 Objetivos e Abordagem

Este projeto tem como objetivo, através da utilização de algoritmos de Inteligência Artificial, criar um modelo de *Machine Learning* para identificar componentes de circuitos elétricos. Com vista à sua concretização, deverão ser testados vários algoritmos de *Machine Learning* que irão servir de base de comparação e, desta forma, permitir a escolha da melhor implementação. Para isto, em primeiro lugar, deve ser obtido um *dataset* para treinar o modelo. Este *dataset*, por sua vez, deve ser composto por imagens-exemplo que devem estar divididas por classes.

De modo a treinar o modelo, é necessário criar uma rede neuronal com várias camadas de abstração. Estas estão normalmente conectadas por canais de ligação que se associam por determinados pesos. Reunidas as condições necessárias, deverão ser efetuados vários treinos, alterando os pesos das redes neuronais, até atingir o melhor resultado. De seguida, após obter um bom resultado, deverão ser testadas as classificações do modelo. Em suma, todos estes passos irão permitir que, após a apresentação de uma imagem, o algoritmo consiga classificar e preencher, com as devidas classes que o mesmo classificou, o modelo JSON criado pela aplicação realizada no projeto anterior [2].

1.3 Plano de Trabalho

Este projeto foi desenvolvido de acordo com o horizonte temporal apresentado no diagrama de Gantt (Figura 1.1). Numa primeira fase, foram debatidos e traçados os objetivos a executar. Neste âmbito, foi realizado um estudo prévio das tecnologias existentes de modo a identificar a melhor metodologia. Após este estudo inicial, procedeu-se ao estudo do Estado da Arte com intuito de conhecer e aprofundar os conhecimentos sobre algoritmos de inteligência artificial.

Em seguida, iniciaram-se os primeiros testes na construção de um modelo de aprendizagem computacional capaz de classificar uma imagem com base num pequeno conjunto de dados formatado para poder começar a testar e otimizar os algoritmos.

Posteriormente à obtenção dos primeiros resultados satisfatórios, surgiu a necessidade de obter um *dataset* mais completo que permitisse validar os resultados

obtidos. A aquisição do *dataset* foi uma tarefa que despendeu mais tempo do que previsto, mas fundamental para os objetivos desta tese.

Com a aquisição e processamento das amostras de simbologia elétrica, iniciaram-se os testes do *dataset* no software desenvolvido anteriormente. [2] Os resultados obtidos não foram de todo os inicialmente previstos, e assim tiveram de ser feitos ajustes consideráveis na estrutura do software e nas camadas da rede neuronal anteriormente desenvolvidas. Por fim, foram validados os resultados obtidos e finalizou-se o conteúdo desta dissertação.

1.4 Organização da Dissertação

A presente dissertação estará organizada em 5 capítulos. O Capítulo 2 apresenta noções básicas de visão computacional, assim como a introdução a conceitos e técnicas de *Inteligência Artificial* (IA). É ainda apresentada uma tecnologia que utiliza caixas delimitadoras e respetivas probabilidades de classificação de cada objeto em tempo real. Por fim, são demonstrados alguns trabalhos realizados neste domínio.

O Capítulo 3 foca no estudo da simbologia que foi utilizada e é onde se encontra a construção da tabela e o resultado da recolha dos símbolos elétricos desenhados pelos alunos.

O Capítulo 4 aborda a *framework* Pytorch, comumente utilizada em algoritmos de *Deep Learning* devido à sua flexibilidade e facilidade de uso. Isto deve-se também à sua compatibilidade com a linguagem de programação Python. Neste são ainda abordados os motivos pela qual esta foi escolhida para realizar o projeto. Além destes, é apresentada a arquitetura de software, explicando assim todas as etapas e a forma em que as mesmas foram abordadas.

O Capítulo 5 apresenta os resultados e as validações dos testes realizados. Neste serão demonstrados os resultados de classificação de treino do modelo e os resultados de teste usando imagens novas, ou seja, nunca antes vistas pelo modelo. Por último, o Capítulo 6 analisa as conclusões retiradas deste trabalho, incluindo as dificuldades sentidas, os objetivos alcançados e possíveis melhorias futuras.

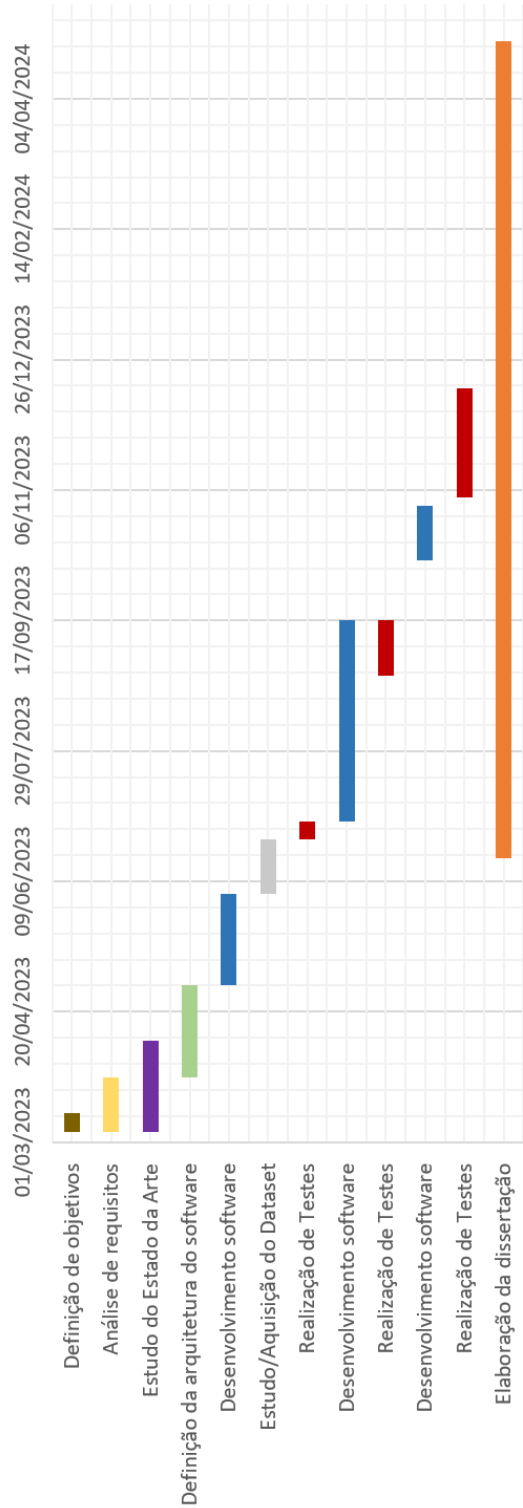


Figura 1.1: Calendarização

Capítulo 2

Contexto Científico e Tecnológico

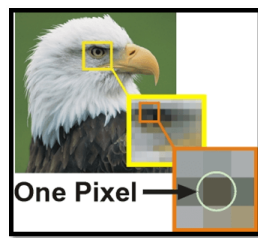
Este capítulo apresenta as noções básicas de visão computacional com vista a introduzir conceitos como, por exemplo, imagem digital e segmentação de imagem. O processo de classificação de imagens possui estes dois conceitos como base.

Neste capítulo é também apresentada a introdução à *Inteligência Artificial* (IA), conceitos como *Machine Learning* (ML), *Deep Learning* (DL) e *Neural Network* (NN), de forma a contextualizar algumas técnicas usadas ao longo deste projeto. Por fim, é apresentada uma breve introdução à deteção de objetos em tempo real, de forma a demonstrar novas abordagens na identificação de componentes em esquema elétricos.

2.1 Processamento de Imagem

2.1.1 O que é uma Imagem Digital

Uma imagem digital[4] é uma representação de uma imagem real na forma de números. Por sua vez, estes números serão armazenados e manipulados por um computador. Para obter uma imagem em formato de números esta é dividida em pequenas áreas, denominadas de *picture elements*, mais conhecidos por pixels.

Figura 2.1: *Pixels*[5]

Cada pixel pode conter várias informações sobre aquela área como, por exemplo, a cor. Numa imagem na escala de cinzentos, os pixels tem valor de um inteiro de 0-255, onde 0 é preto e 255 é branco.

Numa imagem a cores (RGB), cada pixel é composto por uma matriz de 3 elementos e cada elemento também varia entre 0-255. Esta variação representa a quantidade de vermelho, verde e azul, presentes naquele pixel em específico.

Numa imagem com a dimensão de 600×480 , no total temos 288000 pixels. Toda esta informação do valor dos pixels é armazenada em forma de matriz, linhas e colunas, que correspondem a coordenadas horizontais e verticais dos pixels da imagem. Matematicamente, também podemos dizer que uma imagem é uma função tipo $f(x, y)$, onde cada posição definida por (x, y) terá uma característica da imagem naquele ponto.

Em suma, o processamento de imagens digitais utiliza algoritmos e modelos matemáticos para processar e analisar uma imagem, de forma a extrair delas algumas informações úteis. Com isto, os objetivos do processamento digital são melhorar a qualidade da imagem, extrair informações e automatizar tarefas baseadas em imagens.

2.1.2 Segmentação de Imagem

A segmentação de imagens[6] é um subdomínio da visão computacional e do processamento digital de imagem. Assim sendo, a segmentação de imagem é uma das etapas fundamentais do *Machine Learning* que tem por objetivo o agrupamento de regiões ou segmentos semelhantes numa imagem.

Deste modo, este processo consiste em dividir uma imagem digital em subgrupos. Estes subgrupos, denominados por segmentos de imagem, reduzem a complexidade da imagem, permitindo assim o processamento ou análise adicional de cada segmento. Assim sendo, o processo de segmentação trata da atribuição de rótulos a pixels com o intuito de identificar objetos, pessoas ou outros elementos importantes na imagem.

Por outras palavras, uma prática comum no processo de segmentação é a utilização de um algoritmo para encontrar objetos de imagem em vez de processar a imagem por inteiro. Deste modo, o algoritmo pode operar numa caixa delimitadora

já definida pelo algoritmo de segmentação. Isto evita que o algoritmo processe toda a imagem e, assim, melhore a precisão e reduza o tempo de processamento.

Existem diversas formas de segmentar uma imagem. A abordagem mais utilizada baseia-se nas fronteiras, onde são identificadas as regiões onde as variações são significativas na análise dos pixels. A título de exemplo, analisemos o seguinte caso: quando se pretende fazer a limitação binária, é escolhido um limite. Todos os pixels com o valor abaixo desse limite são convertidos a preto e todos os pixels com valor acima do limite são convertidos a branco. Deste modo, concluímos que esta abordagem é mais utilizada para segmentar objetos que tenham maior intensidade que outros ou objetos que se encontram num plano de fundo. Consequentemente, é possível identificar regiões distintas numa imagem.

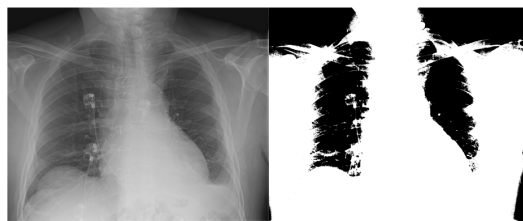


Figura 2.2: Limitação Binária

Uma outra abordagem é a da segmentação baseada nos limites dos objetos. Esta permite localizar os recursos dos objetos, isto é, possibilita a remoção de informações redundantes das imagens, permitindo uma análise mais eficiente. Isto é possível uma vez que o tamanho é inferior em relação ao da imagem total. Assim sendo, estes algoritmos baseiam-se nas variações de contraste, textura, cor e saturação.



Figura 2.3: Detecção de Limites

Existe ainda a segmentação baseada na divisão da imagem em regiões com características semelhantes. Neste tipo de segmentação, cada região é um conjunto de pixels que o algoritmo identifica através de uma *seed* (semente). Após encontrada a *seed*, o algoritmo pode adicionar ou remover pixels da região com o intuito de encontrar padrões idênticos.

Outro método usado para a segmentação consiste na divisão da imagem, agrupando pixels (denominados por *clusters*), com características semelhantes. Portanto, os *clusters* são complementares à visão humana, já que possuem a capacidade de isolar estruturas não visíveis, como sombras e estruturas.

Por último, temos as *watershed* que são um tipo de segmentação em que a imagem é transformada numa escala de cinzentos. Após esta transformação, as imagens são

tratadas como mapas topográficos, onde o brilho do pixel determina a elevação no mapa. Deste modo, o algoritmo divide as imagens em várias regiões com base na altura do pixel, agrupando desta forma os pixels com o mesmo valor de cinza.

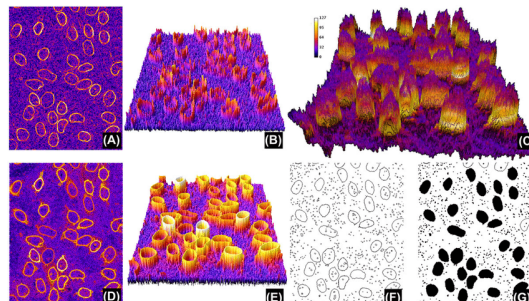


Figura 2.4: Imagem dividida em regiões semelhantes[7]

2.2 Aprendizagem Computacional

A inteligência artificial[8] consiste na simulação da inteligência humana em máquinas que são, deste modo, programadas para pensarem como seres humanos e, de certa forma, imitarem as suas ações. O termo “inteligência artificial” também pode ser aplicado a qualquer máquina ou *software* capaz de desenvolver funções que se assemelham à mente humana. No entanto, estes mecanismos de inteligência artificial não se limitam a métodos biologicamente observáveis.

Nos últimos anos, tem sido verificada uma enorme evolução nesta área, nomeadamente em progressos na realização de tarefas como a imitação de atividades, raciocínio e percepção. Alguns responsáveis deste desenvolvimento acreditam que é possível superar as capacidades do ser humano, enquanto que outros defendem que a atividade cognitiva do homem está ligada a julgamentos de valor que estão sujeitos à experiência humana.

Em suma, atualmente encontramos a inteligência artificial aplicada a quase tudo, desde a condução autónoma e reconhecimento facial, até às recomendações nas redes sociais e a utilização de assistentes de voz. Apesar disto, podemos dividir a inteligência artificial em dois tipos de categorias: os sistemas fortes e os sistemas fracos. [9]

Os sistemas fortes são sistemas que desenvolvem tarefas consideradas humanas. Estes tendem a ser sistemas mais complexos e difíceis. Neste tipo de sistemas a interação humana pode ser nula, como é o caso da condução autónoma.

Os sistemas fracos são projetados para realizar tarefas específicas, como é o caso dos assistentes pessoais.

Alguns conceitos importantes são:

- IA Reativa: usa algoritmos para otimizar as saídas com base nas entradas. Esta IA tende a ser bastante estática, isto é, não é capaz de aprender ou adaptar-se a novas situações. A título de exemplo, temos o jogo do galo, cujo objetivo é sempre vencer a mesma partida.
- IA de memória limitada: adaptam-se a novas situações a partir da experiência de situações passadas ou se a sua base de dados for atualizada. Geralmente, a sua memória é pequena, o que resulta na sua principal limitação. Por exemplo, o veículo autónomo é capaz de adaptar-se a novas situações na leitura da estrada.
- IA teoria da mente: é totalmente adaptável e tem a capacidade de aprender e guardar experiências passadas. A título de exemplo, temos o *chatbot* que simula conversas.
- IA auto-consciente: torna-se consciente da sua própria existência.

2.2.1 Machine Learning

No *Machine Learning*, os algoritmos são treinados através de métodos estatísticos para encontrarem padrões e recursos numa enorme quantidade de dados. O âmbito principal é tomar decisões e obter previsões com base em novos dados. Deste modo, a precisão dos algoritmos melhora à medida que as decisões são tomadas e as previsões obtidas. Assim sendo, podemos dividir os métodos de *Machine Learning* em três tipos de métodos de aprendizagem[10]: supervisionado, não supervisionado e semi-supervisionado.

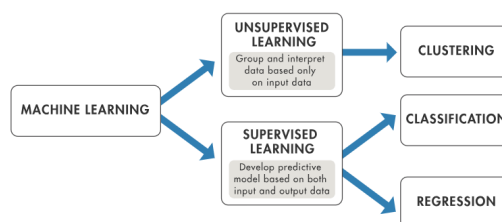


Figura 2.5: Estrutura *Machine Learning*[10]

Ao longo deste tema, os algoritmos de *Machine Learning* serão repetidamente mencionados. Estes algoritmos são, na realidade, equações matemáticas e, por este motivo, a maioria deles baseia-se em variáveis dependentes e independentes. Alguns destes algoritmos utilizam métodos estatísticos nas suas tomadas de decisão, sendo este o motivo pela qual existem diferentes tipos de algoritmos para lidar com diferentes tipos de dados.

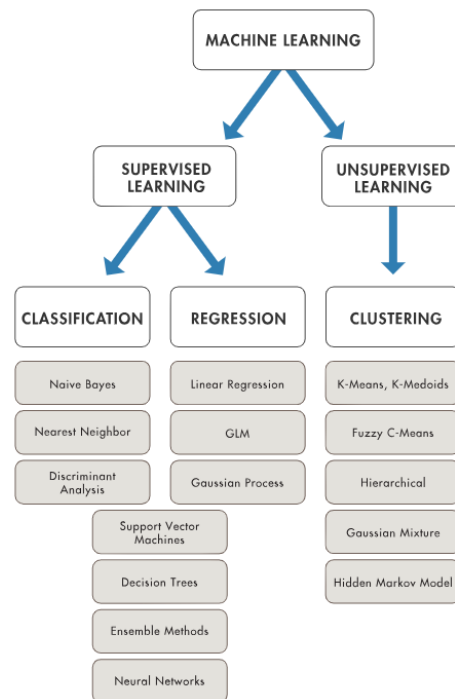


Figura 2.6: Algoritmo Inteligência Artificial[10]

Supervisionado

O método de aprendizagem supervisionado [11] é definido pelo uso de um conjunto de dados rotulados para treinar algoritmos. Este é, frequentemente, apelidado de modelo, quando utilizado para classificar dados ou prever resultados com precisão. Deste modo, sempre que são introduzidos dados na entrada do modelo, este ajusta os seus parâmetros adequadamente com o intuito de dar a resposta mais precisa. O método de aprendizagem supervisionado usa técnicas de classificação e regressão para desenvolver modelos de *Machine Learning*. Como exemplos destes algoritmos, temos as árvores de decisão, as florestas aleatórias, o *Support Vector Machine* (SVM) e Naive Bayes.

1. Técnicas de classificação

As técnicas de classificação são utilizadas sempre que os dados possam ser rotulados, categorizados ou separados em grupos/classes específicos. Nestas, as aplicações de processamento de imagem e visão computacional podem ser utilizadas na deteção de objetos e segmentação de imagem.

(a) Naive Bayes

O Naive Bayes é um algoritmo que pertence às técnicas de classificação de dados[12]. Este é derivado da probabilidade de Bayes [13] e é geralmente utilizado na classificação de texto. O algoritmo em causa pode

ser aplicado na classificação de artigos e também na filtragem de spam, uma vez que apresenta capacidade de análise sentimental. Em suma, este algoritmo analisa o texto e tenta identificar a emoção expressada, tendo como respostas as opções de neutro, positivo ou negativo.

Para isso, o algoritmo cria duas tabelas, sendo uma de frequência de todas as classes e outra de probabilidade. A partir da análise de uma tabela de dados, ele cria classes e apresenta uma resposta a partir dos critérios estabelecidos.

O Naive Bayes é um algoritmo que aprende a probabilidade de cada objeto, as suas características e os grupos a que pertencem. Tratando-se de um algoritmo supervisionado, este apenas classifica por intermédio de um profissional que o vai guiando na construção do modelo.

(b) K-NN

O *K-Nearest Neighbors* (K-NN) (K vizinhos mais próximos) tem no seu âmbito classificar uma amostra de um conjunto de dados através da avaliação da distância em relação aos vizinhos mais próximos. [14] Se estes forem maioritariamente de uma classe, então a amostra será classificada como pertencente a esta. Assim sendo, este é um algoritmo supervisionado utilizado na classificação que parte do pressuposto de que pontos semelhantes possam ser encontrados próximos uns dos outros.

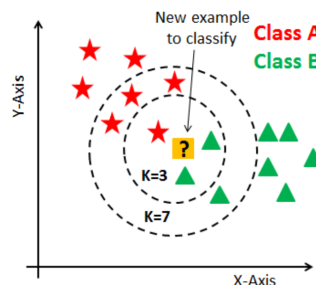


Figura 2.7: Algoritmo KNN [15]

(c) Técnicas de redução de dimensionalidade

As técnicas de redução de dimensionalidade têm um papel crucial nas aplicações de *Machine Learning*. Estas consistem no processo de reduzir o número de variáveis aleatórias que serão inseridas num modelo de treino. Deste modo, estas reduzem a quantidade de informação através da diminuição dos recursos redundantes e dependentes.

Estas técnicas são utilizadas para modelar diferenças em grupos, isto é, separar duas ou mais classes.

A redução da dimensionalidade também reduz drasticamente o custo da aprendizagem da máquina e permite resolver problemas complexos com modelos simples.

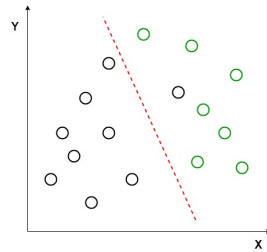


Figura 2.8: Técnica de redução de dimensionalidade[16]

A *Linear Discriminant Analysis* (LDA) [17] é um exemplo de uma técnica de redução de dimensionalidade comumente utilizada em técnicas de classificação de padrões. Esta trata-se de uma técnica de pré-processamento utilizada sempre que existe a necessidade de separar duas ou mais classes com múltiplas características de modo eficiente.

O objetivo da Análise Discriminante Linear é encontrar a linha ou plano que melhor separa os pontos de dados pertencentes a várias classes. Este limite de decisão deve ser escolhido com o intuito de aumentar a distância entre as várias classes e, ao mesmo tempo, diminuir a distância entre os dados dentro da mesma classe.

2. Técnicas de regressão

A regressão em *Machine Learning* é uma técnica muito aplicada em modelos supervisionados. Nestes os algoritmos são treinados para entender a relação entre variáveis independentes e um resultado, ou seja, $y = f(x)$. Em suma, estes modelos são utilizados para preencher o conteúdo em falta ou para prever novos dados. Por sua vez, os resultados podem ser previstos uma vez que a relação entre variáveis independentes e dependentes tenha sido estimada. A análise de regressão é um campo de estudo estatístico. Contudo, os modelos de regressão de *Machine Learning* são usados para prever a relação entre recursos e variáveis de resultado. Portanto, os dados de treino necessitam de ser rotulados.

(a) Regressão linear

O algoritmo de regressão linear é o algoritmo mais simples existente no *Machine Learning*, tendo como intuito resolver o problema de regressão utilizando uma linha reta para descrever um conjunto de dados. Através dele é possível especificar a relação entre um ou mais recursos numéricos num único alvo. Esta técnica usa apenas duas variáveis, onde a variável

x é a variável independente, visto que não vai depender do que estamos a prever, e a variável y é a variável dependente, uma vez que vai depender daquilo que iremos prever. Com base na equação da reta $y = a_0 + a_1x$, concluímos que os dados da entrada irão ser o x , o y é a resposta do algoritmo e os parâmetros a_0 (interceção) e a_1 (coeficiente) são as variáveis que o algoritmo terá de aprender e ajustar de forma a obter a melhor resposta. Esta deve ter o menor erro, o que significa que o erro entre os valores previstos e os valores reais deve ser minimizado. [11]

(b) Modelos Lineares Generalizados

Os *Generalized Linear Model* (GLM) são uma classe a que pertencem as técnicas de regressão que podem ser usados para modelar uma grande variedade de relações entre variáveis de resultado e uma ou mais variáveis de treino. [18]

Os GLM são uma técnica de modelação paramétrica, isto é, que fazem suposições sobre a distribuição dos dados.

Face ao modelo exposto anteriormente que tinha relações lineares, este modelo permite relações mais flexíveis e não lineares através da utilização de distribuições estatísticas, como o Exponencial e o *Poisson*. [19]

No caso dos modelos lineares generalizados, existe uma função de ligação que define a relação entre as entradas e as saídas. Um exemplo padrão de uma função de ligação é uma função exponencial em função de densidade da resposta que permite que a variação de cada observação seja uma função do seu valor previsto.

(c) Regressão logística

A regressão logística[11] é semelhante à regressão linear, salvo na forma como é aplicada. A regressão linear é utilizada para resolver problemas de regressão, enquanto que a regressão logística é aplicada na resolução de problemas de classificação, tendo como intuito a previsão da probabilidade de uma variável pertencer a uma determinada classe.

Na regressão logística, em vez de se ajustar uma reta de regressão, é ajustada uma função logística em forma de “S” que pode usar qualquer número real e mapeá-lo num valor entre 0 e 1.

$$y = \frac{1}{1 + e^{-x}}$$

Esta também tem a capacidade de classificar através da probabilidade novos dados, utilizando conjuntos de dados contínuos discretos.

3. Support Vector Machine

O algoritmo *Support Vector Machine* é um algoritmo de *Machine Learning* aplicado tanto para técnicas de regressão como de classificação. O objetivo deste é criar a melhor linha de limite de decisão num espaço N-dimensional, de modo a dividi-lo em classes. Esta divisão é classificada como hiper-plano.

No hiper-plano a margem entre dois pontos de diferentes classes é sempre o máximo possível, sendo que a dimensão do hiper-plano depende do número de características. Assim sendo, se o número de características de entrada for dois, então o hiper-plano é apenas uma linha. Por outro lado, no caso de o número de características de entrada for três, o hiper-plano torna-se num plano de duas dimensões.

O SVM escolhe os pontos ou vetores das extremidades que auxiliam na criação do hiper-plano de vetores de suporte. [20] Por este motivo, este algoritmo é denominado de *Support Vector Machine*.

4. Árvore de Decisão ou Florestas Aleatórias

A árvore de decisão é uma técnica de *Machine Learning* que representa o processo de tomada de decisão através de uma estrutura ramificada que é semelhante a uma árvore. As árvores de decisão são semelhantes a fluxogramas, sendo que iniciam na sua raiz com uma condição específica de dados que, por sua vez, leva às ramificações que contêm respostas. As ramificações resultam em nós de decisão internos que possuem outras condições, o que origina mais resultados. Consequentemente, isto continua até que os dados atinjam o que é denominado por nó terminal e finalizem. [11]

As florestas aleatórias são um simples conjunto de árvores de decisão, cujos resultados são agregados num resultado final. Este algoritmo resulta em árvores de decisão bastante extensas, mas sem que estas aumentem substancialmente o erro.

5. Ensemble Methods

Os modelos individuais tendem a ter baixa precisão nas suas previsões. Consequentemente, de modo a atenuar esse problema, o *Ensemble Methods*[21] é um algoritmo que procura um melhor desempenho de previsão através da combinação das previsões de vários modelos.

Estes modelos que irão ser combinados são modelos individuais denominados por *weak learners*. Os mesmos possuem este nome devido à variação da sua resposta. No entanto, os *weak learners* conseguem ter uma boa eficiência, apesar do seu desempenho ser baixo, o que origina a que, após algumas previsões, as suas respostas não sejam fiáveis.

Por exemplo, quando usamos uma árvore de decisão, existem vários fatores que devemos ter em consideração, como os recursos usados e o limite da resposta

(SIM ou NÃO) em cada bifurcação. Se for adicionada uma pergunta extra na nossa decisão, podemos esperar que esta seja melhor definida pela nossa classe. Em suma, neste tipo de algoritmos, não podemos confiar apenas na resposta obtida da árvore de decisão, necessitamos esperar pela obtenção do melhor resultado possível ou melhor decisão em cada ramo.

É possível calcular quais os recursos a usar ou quais as melhores perguntas que podem ser feitas a cada ramo de decisão. Assim sendo, é possível fazer uma avaliação no final, não apenas com base na resposta obtida, mas com todos os resultados em cada ramo. Neste tipo de métodos existem três classes principais: *Bagging*, *Stacking* e *Boosting*.

O *Bagging* [22] combina *weak learners* de elevada variação através de um processo de média determinística, e tenta produzir um modelo com menor variação. Os *weak learners* são do mesmo tipo, ou seja, homogêneos. Geralmente, este processo é feito em duas etapas, denominadas por *bootstrap* e *aggregation*. A primeira é um método de amostragem aleatória utilizado para derivar amostras dos dados através do procedimento de substituição, já a segunda é um método que envolve o processo de juntar todas as saídas de todos os modelos e prever um resultado com maior precisão e menor variação.

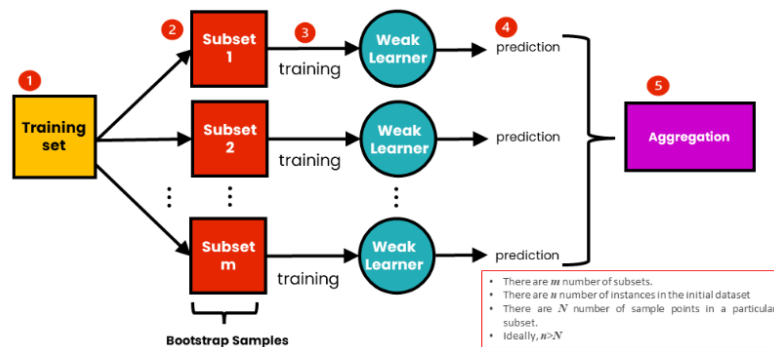


Figura 2.9: Bagging[21]

O *Stacking* [22] visa criar um único modelo robusto através do agrupamento de vários *weak learners*, de forma paralela, para que ao combiná-los com o novo conjunto de dados possamos obter melhores previsões. O processo de *stacking* (empilhar) utiliza as saídas de submodelos como entrada e tenta aprender como combinar melhor as previsões de entrada para realizar uma melhor previsão de saída.

O *Boosting* [22] recorre aos erros gerados por cada *weak learner* para introduzir na entrada do próximo. Deste modo, os modelos posteriores podem melhorar os erros dos modelos anteriores. Ao contrário do *Bagging* que conjugava os resultados apenas no final, o *Boosting* agrega os resultados a cada etapa por meio da média ponderada. Esta média consiste em atribuir pesos diferentes a

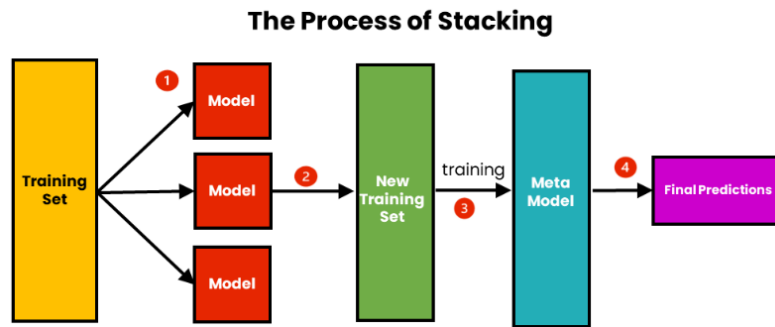


Figura 2.10: Stacking[21]

todos os modelos, dependendo do seu poder preditivo, ou seja, consiste em dar mais peso ao modelo com maior poder preditivo. Em suma, isto ocorre porque a aprendizagem com maior poder preditivo é considerada a mais importante.

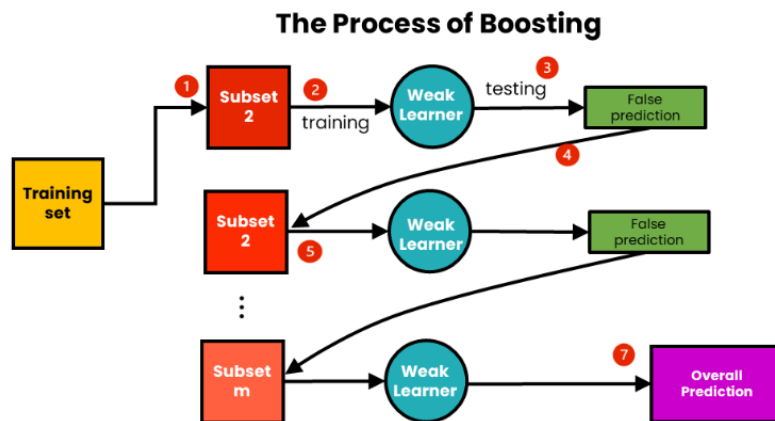


Figura 2.11: Boosting[21]

6. Redes Neurais

As *Neural Network*[23], também denominadas por redes neurais artificiais ou redes neurais simuladas, são um subconjunto do *Machine Learning* e a base do *Deep Learning*. Uma rede neuronal é composta por vários algoritmos que podem interpretar dados, rotulá-los e agrupá-los. Consequentemente, o seu nome resulta da semelhança a um cérebro humano que possui neurónios interconectados entre si.

As redes neurais também são constituídas por neurónios ligados entre si em várias camadas. Geralmente, uma rede neuronal tem mais de três camadas, sendo normalmente constituída por uma camada de saída e uma ou mais camadas ocultas. Quanto mais camadas tiver a rede neuronal, mais profunda é a rede.

As redes neurais dependem de dados de treino para aprender, ou seja, estas dependem de um conjunto de dados já agrupados e rotulados. Estes dados de treino auxiliam o algoritmo a melhorar a sua precisão.

Em suma, as redes neurais são uma ferramenta poderosa na ciência de computação e inteligência artificial. Após treinos, estas conseguem classificar e agrupar dados a grande velocidade. Por exemplo, o reconhecimento de imagem pode levar minutos, enquanto um ser humano manualmente faz o mesmo processo em horas.

(a) Redes neurais *Feedforward*

As redes neurais *feedforward* [11] são o algoritmo mais utilizado nas redes neurais. Este algoritmo apenas tem um único sentido, isto é, a informação apenas é processada numa única direção, começando pela camada de entrada, passando pelas camadas ocultas e terminando nas camadas de saída. Consequentemente, é um tipo de algoritmo muito utilizado no processamento fácil.

(b) Redes neurais *Backpropagation*

Este é um processo utilizado no treino de uma rede neuronal, [24] consistindo em recorrer ao erro das redes *feedforward* para realizar todo o processo em sentido contrário, desde a camada de saída até à camada de entrada, ajustando os seus parâmetros.

(c) Redes neurais Recorrentes

A redes neurais recorrentes agregam os processos anteriores, ou seja, estes são reutilizados durante outro processo. Em suma, à saída de cada nó é guardada a informação que é reintroduzida na rede. Este processo resulta numa aprendizagem mais teórica, recorrendo aos dados aprendidos para reutilizá-los mais tarde. [25] As redes neurais recorrentes são um conjunto conciso de redes neurais *feedforward*, com a capacidade de transmitir informações em intervalos de tempo, introduzindo desta forma uma noção de tempo ao modelo.

(d) Redes neurais Convolucionais

As redes convolucionais, mais comumente denominadas por CNN, possuem várias camadas nas quais os dados são classificados em categorias. [11] Estas redes possuem uma camada de entrada, uma camada de saída e várias camadas ocultas. Estas últimas, por sua vez, mapeiam recursos e registam as áreas de imagem que vão sendo divididas até obterem dados valiosos sobre as mesmas. Este tipo de redes são muito utilizados no processamento de imagem.

(e) Redes neurais Modulares

As redes neurais modulares contêm várias redes ocultas que funcionam independentemente. [26] Assim sendo, não existe qualquer tipo de interação entre as redes durante o processo. Esta tipologia é utilizada com o intuito dos processos mais complexos melhorarem a sua eficiência. Em suma, como o próprio nome indica, cada módulo é responsável por uma parte específica.

Não Supervisionado

O método de aprendizagem não supervisionado recorre a algoritmos para analisar e agrupar dados não rotulados. Assim sendo, estes algoritmos descobrem padrões ocultos sem a necessidade de intervenção humana, tendo como intuito encontrar as semelhanças e diferenças nas informações que permitem agrupá-las.

Este método de aprendizagem não pode ser diretamente aplicado a um problema de regressão ou classificação, uma vez que nos dois existem dados de entrada e no método não supervisionado não possuímos nenhum dado de saída.

Deste modo, o objetivo é encontrar uma estrutura implícita e agrupar de acordo com as semelhanças, representando o conjunto num formato compactado. Contudo, necessitam sempre de intervenção humana para validar as variáveis de saída.

Consequentemente, os modelos de aprendizagem não supervisionado são utilizados para três tarefas principais: agrupamento (*Clustering*), associação (*Association*) e redução de dimensionalidade (*Reduction*).

- O **Clustering** [27] é uma técnica de análise de dados não rotulados que os agrupa nas suas diferenças e semelhanças. O algoritmo mais conhecido que utiliza este método é o K-Means.
- A **Association** [27] utiliza diferentes regras para descobrir as relações entre as variáveis, verificando a dependência de um item com o outro, tendo por intuito encontrar relações ou associações. O exemplo mais comum da aplicação deste método pode ser encontrado em sites de venda online, nomeadamente quando é sugerido um artigo com base na compra de outras pessoas que adquiriram o mesmo artigo pesquisado, sendo que um complementaria o outro.
- **Reduction** [27] é um método utilizado para reduzir a dimensão de um enorme conjunto de dados de entrada, tendo por objetivo manter a mesma informação numa dimensão de dados mais reduzida. Portanto, é uma técnica de pré-processamento de dados.

De seguida, serão analisados brevemente os algoritmos de *Machine Learning* não supervisionados[28] que são mais recorrentemente usados, evidenciando as suas principais características.

1. K-Means

O K-Means é um algoritmo de *clustering* [27] que recorre a um enorme grupo de dados não rotulados para agrupá-los em *clusters*. Este algoritmo iterativo possui um K no seu nome que define o número de *cluster* pré-definidos que necessitam de ser criados no processo. Cada *cluster* K é criado com o intuito de cada conjunto de dados pertencer a um grupo de características semelhantes. Assim sendo, este é um algoritmo baseado em centroides, onde cada *cluster* está associado a um centroide. O objetivo principal do algoritmo é reduzir a distância de cada dado singular ao centro do *cluster* corresponde.

2. K-Medoids

O princípio de funcionamento do algoritmo K-Medoids baseia-se no algoritmo K-Means, em que cada objeto é atribuído a *clusters*. Neste caso, os objetos não são agrupados de acordo com a sua semelhança, mas sim baseado na sua distância. O *Medoid* do *cluster* é o objeto mais central, ou seja, é o objeto cuja média de distância para com os outros objetos é mínima. Por recorrer a esta forma, é pouco sensível aos *outliers*. Estes últimos são dados que se diferenciam drasticamente de todos os outros, sendo esta uma das vantagens deste algoritmo em relação ao K-Means.

3. Fuzzy C-Means

O algoritmo Fuzzy C-Means [27] é uma técnica de agrupamento de dados em que um conjunto de dados é agrupado em N *clusters*. Cada dado pertencente a um *cluster* é definido com um grau de associação a esse *cluster*. Assim sendo, o dado que fica mais longe de um *cluster* terá um grau mais baixo de associação a esse *cluster*, uma vez que o dado que fica mais próximo do centro tem um maior grau de associação para com esse *cluster*. A associação de um *cluster* ao seu grau de associação é feita aleatoriamente no início do processo. De seguida, estas associações serão atualizadas iterativamente até que os centros dos *cluster* se movam para o local correto.

4. Hierarchical

O algoritmo *Hierarchical* é um *cluster* hierárquico [27] que agrupa os dados em escalas, com o intuito de criar uma árvore de agrupamento (dendrograma). Esta representa graficamente o relacionamento hierárquico entre os *clusters* subjacentes. Uma abordagem hierárquica do agrupamento é baseada na determinação de *clusters* previamente definidos. Os algoritmos de agrupamento hierárquico enquadram-se em duas categorias:

- Aglomerativos, onde cada ponto de dados é tratado como um único *cluster* que, de seguida, aglomera sucessivamente, de baixo para cima, os pares de

clusters. A hierarquia dos *clusters* é representada por um dendrograma ou estrutura de árvore.

- Divisivos, onde todos os pontos de dados são tratados como um grande *cluster* e o processo de agrupamento envolve a divisão, de cima para baixo, de um *cluster* grande em vários pequenos *clusters*.

5. Gaussian Mixture

Os algoritmos de mistura gaussiana são modelos probabilísticos que representam uma amostra normalmente distribuída dentro de uma população. Este algoritmo não necessita de conhecer previamente a amostra que representa, o que permite que esta aprenda automaticamente. Por este motivo, trata-se de um modelo não supervisionado. Os modelos de mistura gaussiana são relativamente robustos para *outliers*, o que resulta na produção de resultados precisos mesmo que alguns pontos de dados não se encaixem perfeitamente em nenhum dos *clusters*. Assim sendo, os conjuntos de dados são representados por curvas gaussianas, e quanto mais semelhantes forem duas curvas gaussianas, mais próximas estarão as médias desses dois conjuntos. Deste modo, o número de curvas gaussianas irá indicar o número de *clusters* existentes em cada grupo. [29]

6. Hidden Markov Model

O algoritmo *Hidden Markov Model* (HMM) [29] também é um modelo estatístico utilizado para descrever uma relação probabilística entre uma sequência de observações e uma sequência de dados ocultos. Em suma, este modelo é usado para prever observações futuras ou classificar sequências com base no seu estado atual.

A cadeia de *Markov* é uma máquina de estados finita onde cada estado tem uma probabilidade associada. O principal objetivo da cadeia de *Markov* passa por identificar a probabilidade de transição de um estado para outro. Desta forma, prevê-se que o estado futuro de uma variável dependa apenas do seu estado atual. Assim sendo, o algoritmo HMM consiste em dois tipos de variáveis: os estados ocultos e as observações. Os estados ocultos são as variáveis subjacentes que geram os dados observados, mas não são diretamente observados. Já as observações são as variáveis que são medidas e observadas.

Semi-Supervisionada

O método semi-supervisionado apresenta uma abordagem híbrida de *Machine Learning*, isto é, ele combina dados rotulados e não rotulados para o treino do modelo. [11] Assim sendo, este método usa dados não rotulados para obter informação extra

e melhorar o processo de aprendizagem. Esta é uma abordagem útil devido ao facto da aquisição de dados rotulados ser um processo demorado e dispendioso.

As técnicas de aprendizagem semi-supervisionada podem ser aplicadas em várias tarefas, como a classificação, a regressão e a deteção de anomalias, permitindo que os modelos façam previsões mais precisas e generalizem melhor em cenários reais.

2.2.2 Deep learning

O *Deep Learning*[30] pode ser considerado como uma evolução sofisticada e matematicamente complexa dos algoritmos de *Machine Learning* e é completamente baseada em redes neuronais com múltiplas camadas. Em suma, este é essencialmente uma rede neuronal com três ou mais camadas.

As redes neuronais *Deep Learning*, também denominadas por redes neuronais artificiais, tentam imitar o cérebro humano na forma de aprendizagem, isto é, tentam reconhecer, classificar e descrever objetos através de associações de informação recolhida.

Os modelos de *Deep Learning* são capazes de aprender automaticamente, ou seja, conseguem reconhecer padrões complexos de imagens, o que os torna adequados para tarefas como o reconhecimento de imagem, reconhecimento de fala e o processamento de linguagem natural.

No geral, os algoritmos de *Deep Learning* necessitam de pouca intervenção humana. Embora isto demonstre um grande potencial, apenas agora foi alcançada tanta usabilidade. Isto sucede por duas principais razões: o *Deep Learning* necessita de enormes quantidades de dados para treinar as suas redes e, para poder processar essa quantidade enorme de dados, também necessita de um poder de computação substancial. [31]

No processo de aprendizagem do *Deep Learning* tradicional, o procedimento é supervisionado e o programador necessita de ser extremamente específico, indicando o tipo de coisas que este deve procurar para decidir se uma imagem contém ou não a informação desejada. Deste modo, esta é uma operação exigente que é denominada por extração de recursos. A taxa de sucesso do computador depende inteiramente da capacidade do programador de definir com precisão um conjunto de recursos. A vantagem do *Deep Learning* é que o programa cria um conjunto de recursos sozinho, ou seja, sem supervisão. O *Deep Learning* não supervisionado não é apenas mais eficiente, mas também é mais preciso.

Como Treinar e Criar Modelos de Deep Learning

Geralmente existem três maneiras mais comuns que são usadas para realizar a classificação de objetos para modelos de *Deep Learning* [32], sendo estas o treino a partir do zero, a transferência de conhecimento e a extração de recursos.

No treino a partir do zero, necessitamos de reunir um conjunto de dados rotulados muito numeroso e projetar uma arquitetura de rede que irá aprender todos os recursos. Isto é positivo para novas aplicações ou para aplicações que necessitam de um enorme número de categorias de saída. Assim sendo, esta abordagem é menos comum, já que necessita dessa grande quantidade de dados que levam geralmente vários dias até serem treinados.

Por outro lado, a transferência de conhecimento envolve o ajuste de um modelo pré-treinado. Geralmente, este inicia-se por uma rede existente (*GoogleNet*, *AlexNet* e *ImageNet*) [33] [34] que se alimenta com novos dados que contêm classes desconhecidas. Após o reajuste de alguns parâmetros com base na nova introdução de dados, podemos apenas categorizar as classes pretendidas em vez de possuir milhões de objetos diferentes. A maior vantagem deste método é que necessita de menos dados, não sendo necessário o processamento de milhões dados, e tornando assim menor o tempo de computação.

Por fim, a abordagem de extração de recursos é menos específica e, por este motivo, menos usada. Assim sendo, como todas as camadas têm a tarefa de aprender certos recursos das imagens, podemos retirá-los da rede a qualquer momento, até mesmo durante o processo de aprendizagem. Este elementos podem ser usados como uma entrada para um modelo de *Machine Learning*, como é o caso do *Support Vector Machine*.

Aceleração de modelos de Deep Learning

O tempo de treino de um modelo de *Deep Learning* pode ser de dias ou até semanas. No entanto, o uso de *Graphics Processing Unit* (GPU), pode reduzir significativamente este tempo. Nele, as unidades de processamento gráfico são originalmente desenvolvidas para aplicações que necessitam de um processamento gráfico elevado, como a renderização de imagens, vídeos e animações 2D e 3D. Assim sendo, renderizar é o processo de transformar um modelo computacional em imagens ou animação.

As GPU utilizam um processo paralelo, dividindo as tarefas em sub tarefas menores, que são distribuídas entre os vários núcleos de processamento na GPU. Isto resulta num processamento mais rápido das tarefas de computação especializada. [35]

Estes núcleos de processamento da GPU são denominados por *Tensor Cores*. Estes são núcleos pequenos que realizam multiplicações de matrizes eficientemente. Assim sendo, os *Tensor Cores* são úteis e rentabilizam o processo, tendo em conta que esta é a parte mais cara de qualquer rede de *Deep Learning*.

A NVidia foi a pioneira em unidades de processamento gráfico, lançando as suas primeiras unidades em 1999. Esta desenvolveu uma plataforma[36] de computação paralela que visava permitir ao utilizador o controlo do processamento paralelo das GPU.

Consequentemente, a *Compute Unified Device Architecture* (CUDA)[37] foi lançada em 2006 e permitiu aos cientistas e investigadores desenvolverem aplicações que podem ser locais ou em *cloud* com milhares de GPU. Esta é uma aplicação que inclui bibliotecas, *debugging* e otimização do código com um compilador de C/C++.

2.3 Detecção de Objetos em Tempo Real

2.3.1 Yolo

Na visão computacional, a deteção de objetos em tempo real é uma tarefa muito importante e é um dos pontos principais em sistemas de visão computacional. As aplicações que usam modelos de deteção de objetos em tempo real incluem a análise de vídeo, a robótica, a condução autónoma, a contagem de objetos e análise de imagens médicas.

O YOLO [38] é um modelo de deteção de objetos com uma elevada velocidade e precisão. Este foi apresentado em 2016 e, desde então, tem evoluindo em diversas versões. Ao longo deste desenvolvimento, surgiram versões não oficiais, como foi o caso do YOLO *V5* e *V6*[39][40]. Estas foram criadas por outras entidades e inspiradas em versões oficiais, mas não foram adequadamente revistas nem testadas nos mesmos padrões. A última versão oficial é a YOLO *V7*. [41]

You Only Look Once (YOLO) trata de localizar uma região de interesse dentro de uma imagem através de uma caixa delimitadora e classificá-la como um classificador de imagens comum. A maioria dos algoritmos usa uma *Convolutional Neural Networks* (CNN) para extrair recursos da imagem e prever a probabilidade de classes aprendidas. Os algoritmos de deteção de objetos podem ser divididos em duas categorias: detetores de um único estágio e detetores de dois estágios.

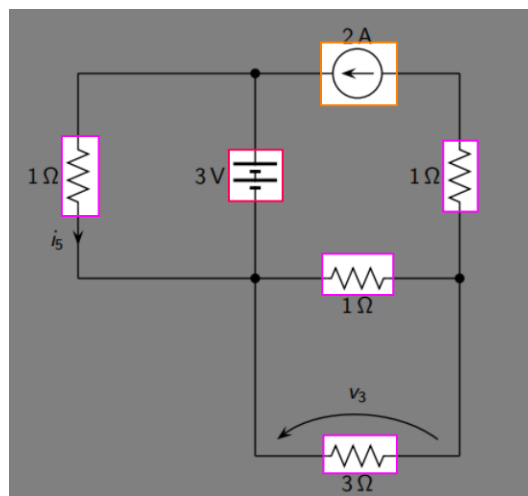


Figura 2.12: Yolo V7[42] na deteção de componentes de um circuito elétrico

Assim sendo, a detecção de dois estágios faz duas passagens na imagem. A primeira passagem é usada para fazer uma primeira aproximação da presença e localização dos objetos, enquanto a segunda passagem é utilizada para aperfeiçoar essas propostas e fazer as previsões finais. Esta técnica é mais precisa do que a de estágio único, mas o seu custo é obviamente mais elevado do ponto de vista computacional.

Por sua vez, a detecção de estágio único faz apenas uma única passagem da imagem de entrada para obter previsões sobre a presença e localização de objetos na imagem. Deste modo, este processa a imagem inteira numa única passagem, tornando-a computacionalmente mais leve. Esta técnica de estágio único é menos precisa e, portanto, é aplicada em detecções em tempo real com recursos computacionais limitados.

Em suma, a detecção de objetos de estágio único é mais adequada para aplicações em tempo real, enquanto a detecção de objetos de estágio duplo é melhor para aplicações onde a precisão é mais importante. Na prática, executar a detecção de objetos em aplicações de visão computacional é muito difícil. As principais adversidades passam pela alocação de recursos computacionais, a robustez do sistema, a eficiência e a latência. Com desenvolvimento das IoT, a capacidade de obter dados a partir de sensores, robôs, câmaras e outros atuadores industriais, possibilitou a implementação de modelos de IA na extremidade. Portanto, no *streaming*, as câmaras têm como *input* imagens e como *output* a detecção de um objeto. Originalmente, isto não acontecia, sendo que a função da câmaras era apenas obter imagens e transferi-las para uma unidade de computação remota.

2.3.2 Roboflow Universe

O Roboflow [43] é uma plataforma com um conjunto de ferramentas que são projetadas para profissionais de visão computacional e *Machine Learning*. Este é utilizado principalmente para gerir e aumentar conjuntos de dados que visam o treino de modelos de visão computacional, mas também fornece outros recursos relacionados ao fluxo de trabalho no desenvolvimento de visão computacional. As principais funcionalidades do Roboflow são:

- Gerir conjunto de dados: o Roboflow permite que os utilizadores façam upload, organizem e possam gerir os seus conjuntos de dados de imagem e vídeo. Isto pode incluir rotular imagens, anotar objetos de interesse e manipular dados em vários formatos.
- Aumento de dados: a plataforma fornece uma variedade de técnicas de aumento de imagem que visam aumentar a diversidade dos seus dados de treino. Isto auxilia no aperfeiçoamento da robustez e do desempenho dos modelos de visão computacional.

- Integração: o Roboflow pode ser integrado em estruturas populares de *Machine Learning*, como o TensorFlow e o PyTorch. Isto simplifica o processo de incorporação do conjunto de dados, aumentando o pipeline de treino do modelo.
- Implementação de modelo: embora o Roboflow se concentre principalmente na preparação de dados, este também oferece recursos para implementar e servir modelos de visão computacional. Assim sendo, este apresenta a capacidade de conseguir exportar modelos para serem usados em vários ambientes de desenvolvimento.
- Colaboração: este oferece suporte à colaboração entre membros da equipa, permitindo que vários utilizadores trabalhem num conjunto de dados simultaneamente e rastreiem alterações e anotações.
- Controlo de versão: tal como um sistema de controlo de versão de software, o controlo de alterações feitas no conjunto de dados ao longo do tempo pode ser gerido, facilitando as iterações e melhorias.
- Suporte de treino: embora não seja uma plataforma de treino completa, o Roboflow pode auxiliar os utilizadores a preparar os seus dados para o treino e conectá-los à infraestrutura de treino da sua preferência.

Por norma, a avaliação de modelos de deteção de objetos é concentrada num conjunto de dados muito genéricos. Assim sendo, por vezes, na sua constituição, encontramos imagens da Internet, como imagens de animais, plantas/flores e meios de transportes como carros, barcos e aviões. Assim sendo, existe uma maior quantidade de imagens disponíveis a menor custo.

Por outro lado, a rotulagem destes e a sua aquisição é um processo que despende tempo, trazendo consigo, deste modo, um custo associado. Desta forma, o Roboflow apresentou a Roboflow 100, RF100[44], que consiste em 100 conjuntos de dados, entre eles imagens de satélite, imagens de microscópio, imagens submarinas e de documentos.

Consequentemente, isto permitiu aos investigadores testar os seus modelos com dados mais concretos, sendo o intuito da RF100 disponibilizar um *benchmark* de um conjunto de dados semanticamente diversificados, com 7 domínios de imagem, o que totaliza 224.714 imagens e 805 rótulos. Por sua vez, estas imagens são rotuladas por profissionais de visão computacional que contam com cerca de 11.170 horas de rotulagem.

O RF100 está dividido em sete categorias: imagens aéreas, jogos, microscópio, subaquáticas, documentos, eletromagnético e imagens do mundo real. O resumo de classes e o conjunto de dados está resumido na tabela seguinte.

Tabela 2.1: Resumo categorias e classes RF100

Categoria	Dataset	Classes
Aéreas	7	24
Microscópio	11	28
Subaquáticas	5	39
Documentos	8	90
Eletromagnético	12	41
Mundo Real	50	495
Jogos	7	88

Existem algumas métricas utilizadas para determinar e comparar o desempenho de diferentes modelos de detecção de objetos, como o YOLO, sendo os mais comuns o *Intersection over Union* (IoU) e *Average Precision* (AP), tal como explicado a seguir.

A IoU calcula a área de intersecção entre as caixas delimitadores para o mesmo objeto, fornecendo assim uma boa estimativa de quão próxima a caixa delimitadora está face à original. Se o valor for 0, a classificação não é correta. Já se o valor for 1, trata-se de uma classificação totalmente correta.

Sendo esta uma comparação entre duas caixas delimitadoras, o valor intermédio não devolve grande informação acerca da qualidade da previsão. Mesmo no caso de ser encontrado outro valor acima deste, é difícil determinar que seria um bom ponto. Assim sendo, a maneira mais usual passa por definir um intervalo e desenhar os resultados num sistema de coordenadas que formam uma curva. No caso dos *dataset* do *Common Objects in Context (Microsoft)* (COCO Microsoft) [45], este valor é calculado entre os 0,5 e os 0,95[46]. O valor calculado abaixo dessa curva, denominado por *Average Precision* (AP), é geralmente definido desta forma: AP@[0,5:0,95].

Portanto, o AP é apenas o valor do IoU num dado ponto. De modo a calcular a média do intervalo, é introduzido o conceito de *mean Average Precision* (mAP). Em suma, este é a média ponderada de todos os valores de AP, nesse dado intervalo, para todas as classes. Na tabela a seguir está representado um resumo das performances obtidas no conjunto de dados RF100.

Tabela 2.2: Resultados Experimentais RF100

Categoria	YOLOv5	YOLOv7
Aéreas	0,636	0,504
Microscópio	0,650	0,591
Subaquáticas	0,560	0,662
Documentos	0,716	0,722
Eletromagnético	0,689	0,607
Mundo Real	0,752	0,899
Jogos	0,859	0,796

No modelo YOLOv5, os valores representados dizem respeito à métrica , assim como no modelo YOLOv7.

Assim sendo, verifica-se que existe uma variação no desempenho entre modelos, existindo também uma variação entre as classes dos modelos. Deste modo, conclui-se que a precisão não se mantém nas classes do mesmo conjunto de dados.

Segundo os criadores do RF100, existem ainda melhorias que podem ser feitas. Estes concluem também que uma das principais razões para as variações de precisão estará no tipo de imagens utilizadas e nos títulos das classes que estão fora do vocabulário existente na maioria dos conjuntos de dados disponíveis na Internet.

2.3.3 Edge AI

Com vista à superação do desafio da falta de recursos computacionais na classificação de dados no *streaming* de imagem, conforme referido anteriormente, foi introduzido o conceito de *Edge AI*[47]. Este consiste na utilização de *Edge Computing* e *Machine Learning*. Assim sendo, o *Edge AI* modera o processamento de *Machine Learning* no dispositivo mais próximo do utilizador, onde geralmente a enorme quantidade de dados está localizada.

Com os avanços da inteligência artificial e unidades de computações gráfica GPU, foram criadas oportunidades para que as máquinas e os dispositivos operem com inteligência. A quantidade de dados gerados para o controlo destes dispositivos é de tal ordem que tornar-se impensável processar os mesmos em *datacenters* remotos ou em *clouds*. Assim sendo, os dados podem ser processados localmente, reduzindo assim problemas relacionados com a latência, a largura de banda e a privacidade.

Consequentemente, a descentralização e os recursos *offline* tornaram o modelo mais robusto, uma vez que o acesso à Internet não é necessário para o processamento de dados. Isto, por sua vez, resulta numa maior disponibilidade e fiabilidade para aplicação. Como os dados são analisados localmente, o tempo de comunicação é reduzido e a resposta à necessidade do utilizador mais eficiente. Deste modo, a utilização da largura de banda diminui, assim como o custo de utilização.

O *Edge AI* aumenta a privacidade ao conter os dados localmente, ou seja, carrega apenas as análises e o conhecimento para a *cloud*. Assim sendo, mesmo que exista a necessidade de carregar alguns dados para o treinar, estes podem ser carregados de forma anónima.

A *cloud* oferece benefícios relacionados com custo da infraestrutura, o fácil acesso e a resiliência contra a falha do servidor. O *Edge Computing* oferece tempos de resposta mais eficientes, menores custos de largura de banda e resiliência contra falhas de rede.

Graças ao desenvolvimento das redes neuronais, aos avanços na computação paralela, às redes 5G e ao crescimento dos dispositivos IoT, existe atualmente uma

estrutura robusta para a aprendizagem local, sem a necessidade de acesso à Internet. Esta tecnologia é bastante útil em aplicações críticas, como a segurança, o transporte, a indústria 4.0 e automação residencial. Através da interligação com o IoT, o *Edge AI* apresenta potencial de se tornar uma ferramenta fundamental para o futuro.

Capítulo 3

Simbologia de componentes elétricos e o "dataset"

Este capítulo apresenta os passos para a criação do *dataset*, desde a recolha, ao recorte das imagens, à sua divisão em classes e o motivo pelo qual se procedeu à criação do mesmo, para que correspondesse às necessidades para atingir os objetivos propostos. Este capítulo introduz também o estudo prévio da simbologia e normas existentes de símbolos elétricos e eletrónicos, de modo a que a recolha deste símbolos siga as normas europeias (baseadas em normas internacionais).

3.1 Estudo de simbologia e normalização

Em 1904, no congresso elétrico internacional, as exposições dos vários membros demonstravam que a eletricidade possuía várias hipóteses de utilização. As tensões podiam variar, a corrente podia ser contínua ou alternada, as frequências e o número de fases eram variáveis e as fichas e tomadas utilizadas também podiam ser diferentes.

Anteriormente, na década 1880, já prevalecia a ideia entre os cientistas que a falta de terminologia, medições e classificações contribuíam para a demorada evolução da ciência elétrica, o que, consequentemente, dificultava o desenvolvimento dos mercados. Deste modo, o congresso de 1904 apresentou a proposta de criação de uma comissão internacional com a principal função de estabelecer os termos e as medidas necessárias às classificações de aparelhos e máquinas elétricas. Isto resulta

na criação da IEC (Comissão Eletrotécnica Internacional), fundada em 1906. Esta era uma organização de liderança mundial na preparação e publicação de normas internacionais para as tecnologias elétricas, sendo presidida por Alexander Siemens.

Consequentemente, com o aprofundamento do estudo desta norma, concluiu-se que a IEC60617:2012 [48] não era uma norma sobre símbolos gráficos para diagramas, mas sim uma base de dados. Esta base de dados é composta pela parte 2 a 13, que pertencia à versão anterior da norma, a edição 2. Deste modo, na atualidade, a base de dados conta com 1900 símbolos.

O acesso à mesma pode ser feito através de subscrição anual, pela quantia de 650 CHF (sensivelmente 678€). Esta pode ser subscrita na sua totalidade ou apenas numa parte, dependendo do interesse do utilizador, já que a base de dados está dividida por categorias.

Com o intuito de validar a sua acreditação em Portugal, foram contactadas duas entidades: o *Instituto Português da Qualidade* (IPQ), integrante da IEC, e o *Instituto Eletrotécnico Português* (IEP). Em resposta ao contacto, o IPQ informou que apesar da IEC60617 ser validada e creditada internacionalmente, esta não é aprovada pela União Europeia. Em suma, reforçaram a ideia que a mesma consistia numa base de dados com inúmeros símbolos que era possível subscrever anualmente, mas que da qual não possuíam acesso. Assim sendo, não era possível validar as últimas atualizações da base de dados.

Por outro lado, a resposta do IEP indicava que existia um acesso restrito, mas que através da loja virtual seria possível obter um *preview* da estrutura da base de dados.

A norma em análise foi desenvolvida pela comissão técnica TC3 da IEC. O organismo europeu da normalização corresponde ao CEN/CENELEC. Por norma, este organismo utiliza a mesma numeração das normas internacionais, adicionando apenas o prefixo EN. No entanto, neste caso, não foi encontrada nenhuma norma com a designação EN60617. Consequentemente, é possível presumir que as regras presentes na IEC60617 não tenham sido adotadas a nível europeu, indo contra o mencionado pelo IPQ.

Por fim, verificou-se que existe um conjunto de comissões técnicas sobre este tema, o CLC/SR3. Nele, existe uma norma, a EN62744:2015, que representa os estados dos objetos por símbolos gráficos padronizados na IEC60617, ISO14617 e IEC60417. Assim sendo, esta resulta numa versão mais completa em que, para além de estar presente a simbologia de componentes eletrónicos, possui também símbolos gráficos que identificam equipamentos e/ou partes de equipamento. Esta também pode indicar estados funcionais, conexões, informações sobre a embalagem e instruções de operação dos equipamentos.

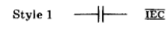
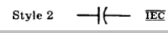
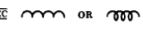
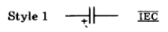
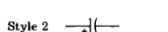
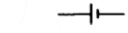
Resistência	Interruptor
 OR  *See Note 2.1A	
Condensador	Bobine
Style 1  OR Style 2 	 OR 
Condensador Polarizado	Instrumentos de Medida
Style 1  OR Style 2 	 *See Note 12.1A A Ammeter  V Voltmeter  W Wattmeter 
Pilha/Bateria	Massa (Ground)
   	 

Figura 3.1: Amostra de símbolos da base de dados da IEC

3.2 Técnicas de classificação de símbolos de circuitos elétricos

A deteção de componentes em imagens de circuitos eletrónicos é uma técnica que tem evoluído de forma a reduzir o tempo despendido na interpretação dos circuitos. Estes podem ser simulados e interpretados de forma digital, mas a sua interpretação em desenho em papel é a forma mais simples de iniciar uma primeira análise.

Assim sendo, após a interpretação na folha de papel, o seu resultado é validado no formato de simulação num software dedicado para esse fim. De seguida, é feita a sua implementação em ambiente de laboratório.

O processo de voltar a desenhar o circuito no programa de simulação é tedioso e pode levar a erros. Como tal, a existência de um método onde seja possível introduzir a imagem previamente desenhada à mão em folha de papel num formato digital, através de uma fotografia ou digitalização, é de uma enorme vantagem. O crescente desenvolvimento da inteligência artificial na área de visão computacional viabilizado a implementação destas técnicas. Deste modo, os métodos baseados em *Machine Learning* têm sido utilizados nos estudos de deteção e classificação de circuitos e componentes elétricos desenhados à mão.

O método *Histogram of Oriented Gradients* (HOG)[49] é um dos métodos geralmente utilizado no reconhecimento de imagens. Este consiste em dividir a imagem em pequenas regiões, em que em cada pixel de uma região são retiradas informações localizadas de intensidade de gradientes, assim como a sua forma e direção.



Figura 3.2: Funcionamento do Método HOG

No estudo desenvolvido por Yuchi Liu e Yao Xiao da Universidade de Stanford [50], após a segmentação do circuito realizada inicialmente, foram classificados os componentes segmentados utilizando o método de classificação SVM. Nestas classificações, obtiveram resultados positivos, com 90% de certeza, num conjunto de 26 componentes de teste. Os componentes escolhidos para a realização do treino deste estudo foram a resistência, o diodo, o amplificador, o condensador e a bateria. Neste dois últimos, foram utilizados mecanismos diferentes por se assemelharem bastante, ou seja, dilataram as duas áreas de contacto, isto é, os terminais, e compararam o seu tamanho. Se o tamanho fosse idêntico, tratava-se de um condensador. Caso contrário, era uma bateria.

Noutro estudo desenvolvido na Universidade de Bahria[49], utilizaram o mesmo método de classificação, assim como o método de reconhecimento. Neste obtiveram uma taxa de classificação de 92% num conjunto de 150 componentes de teste, divididos nas seguintes classes: fonte de tensão AC, fonte de tensão DC, lâmpada, diodo, resistência, bobine, bateria, interruptor, terra e condensador. Assim sendo, de modo a obter o melhor desempenho do sistema, foram testados vários tamanhos de janela para o método HOG, sendo que esta janela refere-se à escala de observação na computação de recursos. Um tamanho de janela mais pequeno capturava linhas e curvas que são muito semelhantes a vários componentes. Ao aumentar o tamanho da janela, eram capturadas informações discriminativas dos gradientes específicos do componente, aumentando também a taxa de classificação. Contudo, o aumento de janela apresentava também um limite: verificou-se que para tamanhos de janela superiores a 32x32, a taxa de classificação começava a descer, uma vez que é retirada pouca informação de uma área grande e existe informação que, consequentemente, acabava por ser perdida.

Outro estudo, realizado por Pablo Sala da Universidade de Toronto, utilizou o método de reconhecimento *Hidden Markov Model* (HMM)[51]. Neste estudo, o modelo está constituído por 4 camadas ocultas do método de HMM, por cada classe. Deste modo, as características básicas de cada símbolo foram analisadas com a área maior na horizontal, em que foram treinadas as 4 camadas, capturando os aspetos gerais das propriedades geométricas de cada símbolo. Assim sendo, foram utilizadas 15 imagens de cada símbolo em 4 posições, isto é, como se tivessem sido rodadas a 0°,

90°, 180° e 270°. Os símbolos utilizados foram: a antena, a bateria, o condensador, a bobine, o díodo, a GND chassi, o loop, a resistência, o transistor, o condensador variável, o díodo zener e o condensador polarizado. Neste estudo, verificou-se que os símbolos da bateria, do condensador e do condensador variável são mutuamente confundidos. Em suma, previu-se que uma maior amostra destes símbolos ajudaria na classificação, pois o classificador teria mais exemplos para aprender sobre os detalhes mínimos entre as categorias mencionadas anteriormente. Deste modo, a taxa de erro total deste modelo ronda os 6,94%.

No estudo desenvolvido por Mihriban Gunau e Murat Koseoglu[52], não foi seguida a mesma abordagem comparativamente aos estudos anteriores, ou seja, não utilizaram o método convencional de deteção de objetos HOG. Assim sendo, estes optaram pelo método *Faster Regions with Convolutional Neural Networks* (Fast R-CNN), sendo este capaz de detetar e classificar objetos. Neste método, o pré-processamento dos dados da imagem é reduzido e o processo de extração de características é feita automaticamente.

O número de camadas utilizadas pode variar dependendo do modelo a ser aplicado. Existem muitos modelos prontos baseados em CNN que são usados em vários estudos, sendo que estes diferem de acordo com os parâmetros, como o número de camadas usadas, a ordem das camadas e a taxa de aprendizagem. Contudo, alguns utilizadores projetaram os seus próprios modelos CNN em vez de usar modelos estruturados nas suas aplicações.

As principais razões para o aumento do uso das CNN em diversas áreas consistem na fácil aplicação, a rapidez entre as etapas de treino e teste e, como mencionado anteriormente, pelo facto do processo de extração de características ser feito automaticamente.

O algoritmo CNN estima a classe de uma determinada imagem, o que resulta numa saída. Por outro lado, também é possível estimar a classe do objeto de uma imagem usando CNN. Porém, não é possível determinar a posição do objeto na imagem. Consequentemente, foi desenvolvido um novo modelo denominado Fast R-CNN. Este é um método de *Deep Learning* baseado em CNN que foi criado para resolver o problema de deteção de objetos na imagem. No desenvolvimento deste estudo foram utilizados os símbolos de resistência, bobine, condensador e fonte de tensão. A inexistência de um conjunto de dados diversificados nesta área levou à criação de um *dataset*. Deste modo, foram recolhidas 800 imagens de circuitos eletrónicos desenhados por várias pessoas. Destas 800 imagens, separadas em duas partes, 80% foram utilizadas no treino do modelo, enquanto que as restantes 20% foram utilizadas para a validação.

O processo de rotulagem de cada símbolo no circuito desenhado foi por intermédio de um *software* criado para esse fim. Este *software* foi desenvolvido em Python e denomina-se por LabelImg. Assim sendo, antes de serem rotuladas, as imagens

foram redimensionadas para um tamanho de 200x200 pixels. A informação da classe de cada objeto e as suas coordenadas foram guardadas em dois ficheiros de formato XML, sendo que um deles estava destinado ao treino e o segundo à validação. Por fim, toda a informação contida nestes dois ficheiros foi convertida num ficheiro CSV.

Os resultados obtidos neste estudo são demonstrados através da função de perda (*loss function*). Este é um método que avalia o quão bem o algoritmo modelou o seu conjunto de dados. Se as suas previsões forem totalmente erradas, a função *loss* produz um número maior. Em suma, isto indica que os parâmetros devem ser ajustados de forma a melhorar o modelo, tendo como intuito a produção de números cada vez menores.

Quando examinamos os gráficos de perdas (Figura 3.3), estes mostram o desempenho do modelo. Neles, pode verificar-se que existe uma diminuição gradual da perda. Deste modo, verificou-se também que após um dado instante, o valor da perda acabou por estabilizar.

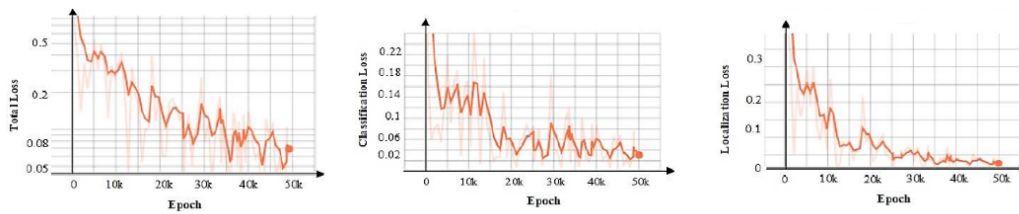


Figura 3.3: Demonstração de resultados do modelo R-CNN

Concluiu-se, considerando o gráfico de perda total, que o mínimo valor obtido foi de 0,048, terminando o processo com o valor de 0,06. Verificou-se também que o valor de perdas desce rapidamente abaixo de 0,1, o que demonstra que o modelo terá uma boa taxa de precisão nas deteções em tempo real.

No estudo realizado por Felix Thoma, Johannes e Yakun Li [53], na construção do *dataset* recolheram no total 1152 imagens. Nesta recolha de imagens pediram a 12 pessoas para desenharem duplicadamente 12 circuitos, considerando vários tipos de papel e lápis. No processo de conversão para formato digital fotografaram cada circuito 4 vezes, usando também diferentes tipos de luz. As imagens foram guardadas em formato JPEG com resoluções compreendidas entre 864x4160 e 396x4032 pixels.

O processo de anotação foi realizado manualmente através da ferramenta *labelImg*, sendo estas guardadas num ficheiro XML no formato *PascalVOC*. O *dataset* ficou com um total 45 classes e 48563 anotações. Para este estudo usaram um modelo *Faster Regions with Convolutional Neural Networks* (Fast R-CNN), com um *learning rate* de 0.007 e um *batch size* de 8, produzindo assim um resultado *mean Average Precision* de 52%.

Num outro estudo realizado por Mrityunjoy Dey, Shoif Md Mia, Navonil Sarkar e Archan Bhattacharya [54], decidiram realizar o mesmo, usando um método com

dois estágios. O primeiro estágio passa por separar os componentes por 4 grupos e apenas no segundo estágio era atribuída a categoria real do componente. Foram recolhidas 150 imagens com resolução de 64x64 pixels, de um conjunto de 20 circuitos desenhados à mão. Usaram um modelo simples baseado num modelo CNN, composta por um camada de convolução sucedida de uma *max-polling*, sendo estas repetidas 5 vezes. No final o mapa de características é achatado através de uma camada de ativação linear. Para finalizar, usaram duas *layers fully connected*, sendo uma delas a classificação de saída.

A precisão do primeiro estágio é de 86%. No segundo estágio a precisão varia entre os diferentes grupos, tendo o primeiro grupo uma precisão de 96.11%, e os restantes uma precisão, 100%, 91.85% e 98.78%. O primeiro grupo é composto na totalidade por portas lógicas, o segundo apenas por transístores, o terceiro é composto apenas com a simbologia do voltímetro, amperímetro e fonte de tensão AC, e o grupo 4 é o grupo composto por mais símbolos, sendo estes mais diversificados, como condensadores, bobines, resistências, interruptor, díodos e transformador. A precisão final do segundo estágio é de 97.33%.

No estudo realizado por Mihriban Gunay, Murat Koseoglu e Özal Yildirim[55], estes realizaram também uma recolha de 863 imagens, sendo que 709 delas foram usadas para treino e as restantes para teste final. Todas as imagens foram redimensionadas para uma resolução de 150x150 pixels e convertidas em escala de cinzas. Este processo foi realizado de forma a reduzir a complexidade computacional, economizar tempo e reduzir a área de cálculo. Os investigadores usaram um modelo CNN com 12 *layers* compostas por 5 *layers* convolucionais com um filtro de 3x3 pixels, em que cada uma das 5 *layers* anteriores é sucedida por layers pooling, tendo estas, por sua vez, um filtro de 2x2 pixels. O próximo conjunto de layers é composto por uma *layer fully connected*, denominada por *softmax*, e por uma função de ativação ReLu. No total foram usadas 100 iterações no ciclo de treino, finalizando com um precisão de 84.416%.

Noutro estudo, realizado por Aditi Kumar e Riyaz Merchant [56], foram recolhidos um conjunto de imagens com 8 classes, em que cada classe era constituída por cerca de 20 símbolos. No final, o conjunto era de 168 imagens. Todas estas foram recolhidas em papel branco e desenhadas com caneta preta para reduzir o pré-processamento quando convertidas em escala de cinzas. As imagens foram também redimensionadas numa resolução de 120x160 pixels. Para aumentarem o número de amostras, introduziram o processo de data augmentation, realizando rotações de 45 graus a cada imagem e introduzindo *gaussian blur*, isto é, uma técnica de desfoque suave através de uma convolução com uma função gaussiana. Após aplicadas estas técnicas, o *dataset* terminou com um total de 1760 imagens.

No total, 80% das imagens foram utilizadas como imagens de treino e as restantes 20% como imagens de teste. Neste estudo o ajuste dos hiper parâmetros tiveram por

base os presentes na documentação do Tensor Flow e os encontrados na literatura. Os investigadores optaram também por recorrer ao otimizador Adam devido aos seus benefícios ao lidar com gradientes escassos e com ruído. Este otimizador tem a vantagem de ser possível a atualização da taxa de aprendizagem por parâmetro, contrariamente à do SGD que utilizava a mesma taxa de aprendizagem em todos os parâmetros, resultando num treino mais longo, já que alguns parâmetros são atualizados mais rapidamente dos que outros. Consequentemente, os investigadores recorreram ao método CNN e o algoritmo conseguiu alcançar a precisão de 96%.

3.3 Preparação e Construção do Dataset

Ao longo da realização deste projeto, surgiu a necessidade de criar uma base de dados de símbolos de componentes elétricos. Esta necessidade advém da falta de um conjunto de imagens (*dataset*) formatado para os requisitos idealizados para esta aplicação, de modo a ser utilizado para treinar um modelo de *Machine Learning*.

Numa das pesquisas realizadas num dos maiores repositórios de *datasets*, foi encontrado um exemplar [57], porém este não foi selecionado por não possuir todos os símbolos pretendidos e que cumprissem os requisitos da nossa aplicação.[1] Este *dataset* continha outros que não se inseriam nesta primeira abordagem de classificação de símbolos eletrónicos. Um outro *dataset* [42] encontrado com simbologia eletrónica, também não foi considerado devido à sua aplicação ser mais direcionada para algoritmos que usam YOLO. Consequentemente, foi formulada uma tabela com os símbolos dos componentes elétricos necessários no nosso contexto.

Assim sendo, com o auxílio dos alunos da turma de Teoria dos Circuitos do primeiro ano de Licenciatura em Engenharia Eletrotécnica e de Computadores, e com os alunos do curso Técnico de Eletrónica, Automação e Instrumentação da Escola Profissional Cenatex no ano letivo de 2022/2023, foi possível obter 3441 desenhos de símbolos de componentes eletrónicos. Esta tabela foi estruturada de modo a que cada símbolo fosse desenhado em duas posições: horizontal e vertical, tal como exemplificado na Figura 3.4. Note-se que o desenho do símbolo horizontal é delimitado por uma caixa azul enquanto que na vertical a caixa é vermelha. Com isto, foi possível recolher duas configurações do mesmo símbolo desenhado pela mesma pessoa, tornando assim o modelo mais robusto.

3.4 Preparação do Dataset

Após o preenchimento da tabela por parte dos alunos e de alguns docentes, foi necessária a digitalização das 169 amostras recolhidas. De seguida, foi desenvolvida uma aplicação, usando Python, para a separação dos diferentes símbolos por classes.

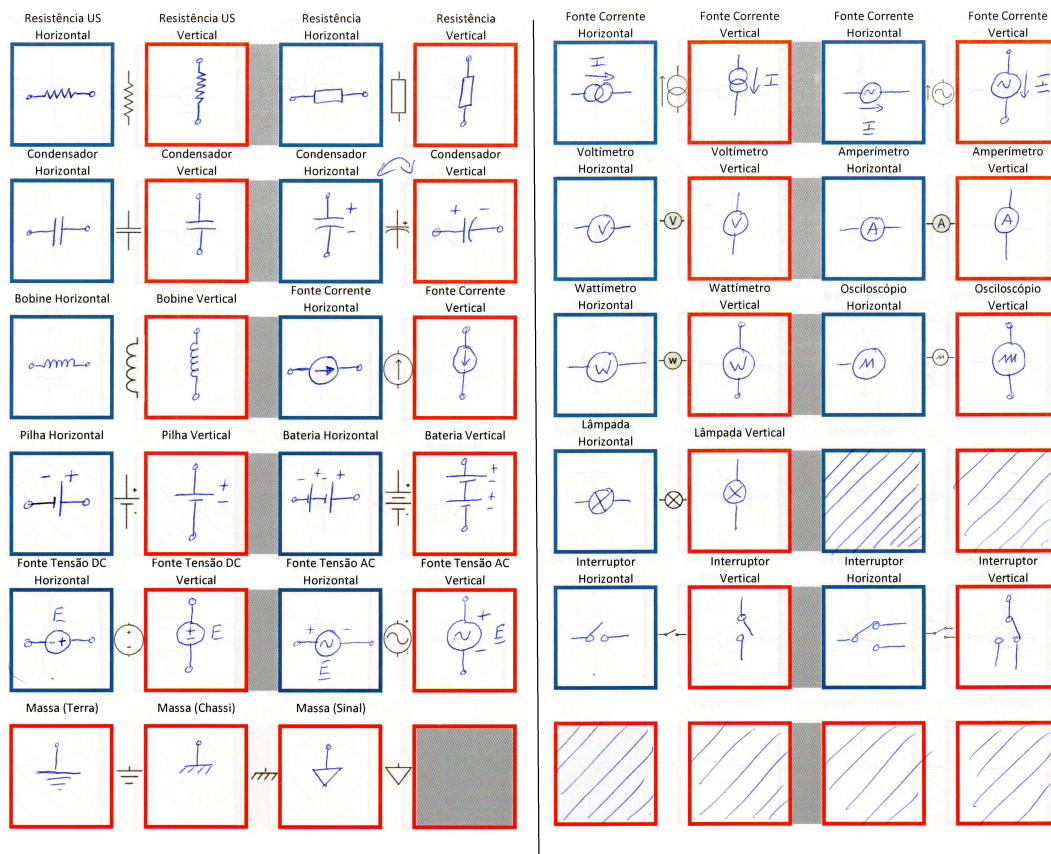


Figura 3.4: Tabela Criada para Recolha de Símbolos

O processo de digitalização requereu alguns cuidados, como a remoção de rasurados. Verificou-se que, em alguns casos, os alunos começavam por desenhar o símbolo na horizontal no lugar da vertical e que, quando se apercebiam, rasuravam e desenhavam novamente o símbolo abaixo. Outros dos cuidados exigidos foi o de manter o vidro do scanner o mais limpo possível, pois as impurezas podiam afetar na dimensão do documento de saída. Esta anomalia foi detetada num lote de documentos onde o vidro do scanner apresentava uma impureza fora da área da folha A4. Assim, o scanner fazia a leitura da impureza como informação, resultando numa leitura do scanner superior a uma folha A4. Como o ficheiro de saída seria do tamanho de folha A4, a informação contida no ficheiro final sofria uma redução de tamanho.

Assim sendo, as 169 tabelas foram digitalizadas e convertidas em formato JPG. Após a aquisição de todas as tabelas em formato digital, a aplicação desenvolvida era executada com a informação do caminho das tabelas em formato digital. Dentro desse caminho, a aplicação percorria todas as digitalizações e convertia as imagens em escala de cinza. Após esta conversão, a aplicação recortava a informação dentro de cada quadrado, vermelho ou azul.

Este quadrado foi utilizado inicialmente para limitar o espaço de desenho dos

símbolos, uniformizando assim o seu tamanho. Deste modo, foi permitido também definir uma coordenada fixa para cada componente. Por sua vez, a diferença de cor entre os quadrados auxiliou visualmente o aluno na diferenciação do desenho do símbolo na horizontal ou na vertical, isto é, serviu de referência visual. Após o recorte dos símbolos dentro de cada quadrado, a aplicação guardou todas as imagens recortadas dentro de uma pasta, com um nome de ficheiro sequencial. A divisão dos símbolos por classes foi efetuada manualmente, tendo como intuito a realização de um filtro das amostras recolhidas.

Capítulo 4

Abordagem de implementação

Neste capítulo será introduzida a *framework* em que se desenvolveu a parte fundamental deste projeto, o Pytorch. Utilizando o Pytorch foi possível implementar um algoritmo de *Machine Learning*, aproveitando várias das suas funcionalidades. Serão ainda apresentadas bibliotecas que serviram como teste e outras que tiveram um papel importante no processamento das imagens que compõem o *dataset*.

Será também apresentada a arquitetura de software, identificando os diferentes blocos e fases, desde o treino, classificação de componentes e o preenchimento do modelo de dados, JSON , que é transversal a outras aplicações da *framework* U=RI solve.

4.1 Linguagens de Programação e Bibliotecas

A biblioteca *Numpy*[58] é conhecida pela sua usabilidade na manipulação de *arrays* e matrizes de dados numéricos. As matrizes *Numpy* são otimizadas para operações matemáticas e estatísticas, possuindo a capacidade de serem muito mais rápidas do que os ciclos que são utilizados para realização de iterações em listas.

No processamento de imagem, a biblioteca *Numpy* armazena os valores dos pixels em matrizes, recorrendo ao formato de *bytes*. No caso da imagem ser colorida, cada pixel tem a informação das três componentes da cor RGB, de 0 a 255. Assim sendo, o valor de cada pixel é composto por 3 *bytes*. Por sua vez, estes são armazenados sequencialmente na memória, possibilitando um acesso mais rápido.

A *Python Imaging Library* (PIL/PILLOW) [59] é uma das bibliotecas de manipulação de imagem mais conhecidas e utilizadas em linguagem Python. Esta é uma opção muito recorrente para tarefas de alto nível que não exijam experiência prévia no processamento de imagem. Devido à sua fácil aprendizagem e utilização. Esta biblioteca suporta vários formatos de imagem, como o BMP, JPEG, TIFF, e apresenta comandos muito simples que permitem operações básicas como cortar, redimensionar, girar, inverter e ajustar as propriedades da imagem como desfoque, nitidez, ajuste de contraste e manipulação de cores.

As funcionalidades desta biblioteca tiveram um grande impacto na construção do conjunto de dados, foi uma ferramenta essencial no processamento das digitalizações das tabelas preenchidas pelos alunos com a simbologia dos componentes eletrônicos.

O OpenCV [60] é a biblioteca mais conhecida e fortemente utilizada para a manipulação de imagem no âmbito do processamento de imagem e *Machine Learning*. Esta é uma biblioteca *open source*, escrita em C/C++, que se encontra otimizada para a utilização de processadores *multicore*. O *OpenCV* é utilizado em processamento de imagem e vídeo, com o intuito de identificar objetos, rostos e até caligrafia, de modo fácil. Consequentemente, este possui várias funções na área da visão. Esta é uma biblioteca multiplataforma, que quando integrada com outras bibliotecas, como o caso da *Numpy*, é capaz de processar estruturas de *arrays*.

O *OpenCV* possui uma biblioteca desenvolvida especificamente para *Machine Learning* que é focada no reconhecimento e agrupamento de padrões estatísticos. As principais aplicações deste consistem na segurança e vigilância e na condução autónoma. No entanto, nos últimos anos, tem sido utilizada também no controlo de linhas de produção para a deteção de defeitos em produtos de diversas áreas. [61]

O excerto de código seguinte (Código 4.1), mostra uma aplicação das bibliotecas mencionadas anteriormente, nomeadamente o OpenCV (biblioteca importada como `cv2`), Numpy e PIL/PILLOW. Este excerto tem o âmbito de verificar se o método de rotulagem após o processo de *datasplitting* está a atribuir corretamente o rótulo à imagem correspondente. O código inicia com um redimensionamento da imagens do *dataset* utilizando a biblioteca PIL/PILLOW. Este não é um passo necessário, mas o conjunto de imagens redimensionadas seria usado mais à frente e, desta forma, permitiria visualizar a resolução da imagem com que se iria trabalhar. Após este redimensionamento, o próximo passo teve como função criar uma janela utilizando a biblioteca de processamento de imagem OpenCV, para mostrar a imagem. Normalmente o OpenCV utiliza o formato BGR para esta função, por isso verifica-se que existe uma conversão do esquema de cores da imagem. Para realizar este processo, a imagem necessita de estar no formato de array. Para isso utilizamos a biblioteca Numpy. Por fim, é introduzido como título da janela o rótulo da imagem guardado no vetor *label*. Esta imagem rotulada estará visível durante o período de 200 milissegundos e, posteriormente, a janela será fechada automaticamente. Este processo

repete-se até à última imagem do *dataset*.

```
1 import cv2
2 import numpy as np
3 from PIL import Image
4
5 target_width, target_height = 64, 64
6 resized_images = [img.resize((target_width, target_height)) for
    img in images]
7 for img, label in zip(resized_images, labels):
8     cv2.imshow("Image", cv2.cvtColor(np.array(img), cv2.
        COLOR_RGB2BGR))
9     cv2.setWindowTitle("Image", f"Label: {label}")
10    cv2.waitKey(200)
11    cv2.destroyAllWindows()
```

Código 4.1: Rotina de validação dos rótulos de cada imagem no conjunto de dados

4.1.1 Frameworks para Classificação de Imagem

Tensor Flow

O *Tensor Flow* [62] é uma plataforma completa para *Machine Learning*, criada pela *Google* em 2015 em código aberto. Esta plataforma torna fácil o processo de aquisição de dados, modelos de treino, previsões e refinamento de resultados futuros.

O *Tensor Flow* pode treinar e executar redes neuronais profundas na classificação de dígitos manuscritos, no reconhecimento de imagem e no processamento de linguagem natural. Deste modo, podem ser utilizados os mesmos ambientes de desenvolvimento da linguagem Python. Assim sendo, é possível utilizar o Python e *Tensor Flow* no mesmo código, recorrendo às várias funcionalidades de ambos. A título de exemplo, temos o facto de o *Tensor Flow* lidar com tensores e todas as operações matemáticas neles envolvidas, ao mesmo tempo que o Python direciona o tráfego entre as partes e fornece uma camada de abstração de uma programação de alto nível. Tensores podem ser classificados como uma generalização de vetores e matrizes para dimensões variáveis.

Keras

O Keras [64] é uma API de redes neuronais de alto nível escrita em Python e em código aberto. Foi projetada para ser fácil de usar, modular e extensível, permitindo facilmente a construção e experimentação de modelos de *Deep Learning*. Esta API disponibiliza uma interface simples que permite construir vários tipos de redes neuronais, com por exemplo, redes *feedforward*, CNN e RNN. O Keras permite a

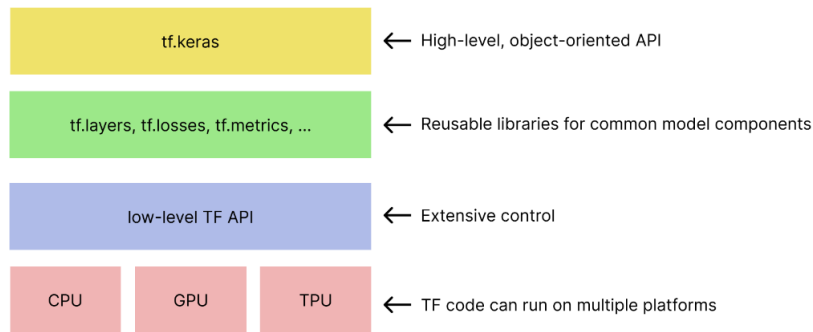


Figura 4.1: Estrutura Keras e Tensor Flow[63]

abstração de tarefas de computação de baixo nível, como o TensorFlow, tornando-o acessível a um público mais abrangente. Por outro lado, a personalização de certos aspectos de modelos ou processos de treino pode ser um desafio para aplicações mais específicas, o que pode limitar a configuração e manipulação das funções quando se pretende criar um modelo de *Deep Learning*.

O Keras segue um design modular, permitindo construir, modificar e experimentar facilmente diferentes arquiteturas de redes neurais. Esta arquitetura suporta a criação de modelos complexos através da combinação de vários blocos de construção, como pode ser visto no Código 4.2. Esta construção por blocos pode-se tornar crítica. O bloco pode não estar totalmente ajustado à aplicação pretendida, assim introduz sobrecarga no desempenho do modelo. Os optimizadores podem ser uma ferramenta valiosa no processo de treino, sendo que estes permitem ajustar a taxa de aprendizagem para que o sistema convirja mais rapidamente. Este ajuste fino da taxa de aprendizagem não é possível usando o Keras, portanto, o sistema demora mais tempo a convergir, criando assim uma sobrecarga no sistema. As mensagens de erro da biblioteca Keras são ineficientes, sendo o erro facilmente detetado se for claramente visível. Por vezes, encontrar a causa principal do erro não é uma tarefa fácil, o que torna a depuração uma tarefa crucial neste tipo de aplicações.

```

1 history = model.fit(
2     #imagens para treino do modelo
3     train_generator,
4     #tamanho dos mini lotes de imagens de treino
5     steps_per_epoch=train_generator.samples // batch_size,
6     #imagens para validacao do modelo
7     validation_data=validation_generator,
8     #tamanho dos mini lotes de imagens de validacao
9     validation_steps=validation_generator.samples // batch_size,
10    #numero total de iteracoes
11    epochs=500
12 )

```

Código 4.2: Rotina de Treino usando Keras

PyTorch

Para além das plataformas referidas anteriormente, existe uma outra que tem crescido exponencialmente devido à sua compatibilidade com o Python: o *Pytorch* [65]. Este está mais orientado para a aplicação de *Deep Learning*, sendo utilizado para criar redes neurais profundas no reconhecimento de imagem e processamento de linguagem [66].

O *Tensor Flow* e o *Pytorch* são muito semelhantes, já que ambos trabalham ao nível dos tensores. O *Pytorch* foi desenvolvido pela *Facebook AI*, atualmente denominada de *Meta AI*. Esta estrutura combina bibliotecas *back-end* aceleradas por unidades GPU provenientes do *Torch* e um *front-end* de fácil aprendizagem Python.



Figura 4.2: Funcionalidades e vantagens de usar Pytorch[67]

Esta plataforma distingue-se das outras pelo seu suporte de GPU[68]. Assim sendo, este calcula o gradiente das funções recorrendo ao *reverse-mode* e ao *auto-differentiation*, permitindo assim que as unidades gráficas sejam modificadas em tempo real. Essencialmente, o *reverse-mode* é como um gravador que regista as operações concluídas e as reproduz de trás para frente, calculando os gradientes. O *auto-differentiation* refere-se ao cálculo automático de gradientes ou derivadas de expressões matemáticas em relação às variáveis de entrada. Os gradientes são cruciais para algoritmos de otimização como SGD durante o treino de redes neurais. [69]

O *Pytorch* suporta as CPU's, GPU's e processamento paralelo, podendo o seu treino ser distribuído. Isto significa que o trabalho computacional pode ser distribuído pelos vários núcleos de GPU e CPU ou realizado em várias GPU [70] de diferentes unidades de computação.

Por fim, este possui muita documentação e tutoriais (disponíveis no seu *web-site*), o que resulta numa mais-valia para quem pretende iniciar-se em aplicações de inteligência artificial.

```

1  for epoch in range(num_epochs):
2      model.train()
3      running_loss = 0.0
4      for inputs, labels in train_loader:
5          optimizer.zero_grad()
6
7          #rede neuronal inputs-imagens, outputs-weights/biases
8          outputs = model(inputs)
9          #transformar o vetor de label em inteiro
10         labels = [label_to_int[label] for label in labels]
11         #transformar em tensor
12         labels = torch.tensor(labels, dtype=torch.int64)
13         #funcao de calculo de perda, compara os outputs do
            treino com as labels
14         loss = criterion(outputs, labels)
15         #calcula os gradientes dos parametros do modelo em
            relacao a perda
16         loss.backward()
17         #usa os gradientes calculados na funcao anterior
18         optimizer.step()
19         running_loss += loss.item()
20
21         epoch_loss = running_loss / len(train_loader)
22         #imprime o valor da funcao loss
23         print(f"Epoch {epoch+1}/{num_epochs}, Loss: {epoch_loss:.8
            f}")
24     #Guarda o modelo
25     torch.save(model.state_dict(), 'Model.pth')

```

Código 4.3: Rotina de Treino usando Pytorch

A natureza dinâmica e facilidade de implementação tornam o Pytorch adequado à prototipagem e experimentação de novos métodos. A depuração de erros é mais simples de solucionar, já que as ferramentas de depuração nativas do Python são as mesmas usadas no Pytorch. Consequentemente, é possível recorrer a ferramentas de mais baixo nível comparativamente ao Keras, onde maior parte das funções são construídas por blocos, tendo o Pytorch como vantagem a possibilidade de controlar todos os parâmetros de treino dos modelos[71], conforme o excerto de Código 4.3. Portanto, este foi o motivo pelo qual esta foi a biblioteca usada para este projeto.

4.2 Arquitetura de software

O algoritmo de inteligência artificial que tem como função a identificação de componentes de circuitos elétricos/eletrônicos está dividido em 3 partes principais, como apresentado na Figura 4.3.

Em primeiro lugar, existe um diretório com as imagens recolhidas para treinar o modelo. Estas imagens estão divididas em 22 classes, representando os símbolos de componentes elétricos/eletrônicos e instrumentos de medição: Amperímetro, Bateria, Condensador, Condensador Polarizado, Pilha, Fonte de Corrente AC, Fonte de Corrente DC, Fonte de Corrente Constante DC, GND Chassi, GND Digital, GND Terra, Bobine, Lâmpada, Osciloscópio, Resistência (símbolo Europeu), Resistência (símbolo Americano), Interruptor 1 polo, Interruptor 2 polos, Fonte Tensão AC, Fonte Tensão DC, Voltímetro e Wattímetro.

Existe também uma outra diretoria que é composta pelas imagens recortadas do circuito original, tendo estas sido segmentadas pelo módulo anterior, desenvolvido pelo colega Hugo Barbosa[2]. Dentro deste diretório, encontra-se também o arquivo JSON, também gerado pelo módulo anterior.

Este arquivo JSON engloba toda a informação de análise ao circuito (Modelo total nos anexos, Código B.1), nomeadamente os seguintes parâmetros:

- *Component*: Representa o tipo de componente do modelo;
- *Connection*: Representa o tipo de conceção do modelo;
- *ID*: Índice do objeto;
- *Node*: Nó de ligação no circuito;
- *Port*: Pontos de conceção do objeto no circuito;
- *Position*: Posição global e relativa do objeto.

Para além destes dois diretórios, existem os módulos de código desenvolvido. Estes encontram-se também divididos em 3 módulos, sendo que o último faz a interligação entre os dois anteriores. O módulo *training* tem como função treinar o modelo de *Machine Learning* através das imagens do *dataset*. Após o treino, este guarda o modelo tendo em vista o seu uso posterior. O módulo *predict* carrega o modelo pré-treinado pelo módulo *training* e realiza a previsão/classificação da imagem inserida, retornando a classe correspondente. O módulo *jsonfill*, inserido no módulo *predict*, utiliza o *id* da imagem e a classe prevista pelo módulo anterior para preencher os campos em falta no módulo JSON já existente.

Por fim, o módulo *main* permite ao utilizador escolher se pretender realizar o treino. Caso este não o pretenda, é feita a passagem através do argumento do caminho da imagem que se pretende classificar. A *main* passa este caminho para o *predict* e este classifica a imagem, retornando de volta a classe a que pertence a imagem passada.

Após a resposta deste, é encaminhada de seguida para o módulo *jsonfill* com o intuito deste preencher os campos em falta conforme a resposta obtida.

4.2.1 Início do Programa

O fluxograma seguinte (Figura 4.3) representa as etapas do programa principal do modelo. Este começa por analisar os parâmetros passados pela linha de comandos, aceitando apenas dois tipos de comandos: treino e classificação. No caso de se tratar de uma classificação, este comando deve ser acompanhado pelo caminho da imagem que se pretende classificar, como mostra o excerto de Código 4.4. Por outro lado, caso se pretenda treinar o modelo, este sinaliza outro programa que apenas treina o modelo e depois finaliza o programa.

No caso da classificação, este sinaliza o programa *predict* que retorna o valor da classificação. Por sua vez, esse valor será utilizado por outro programa para preencher os campos em falta num módulo JSON.

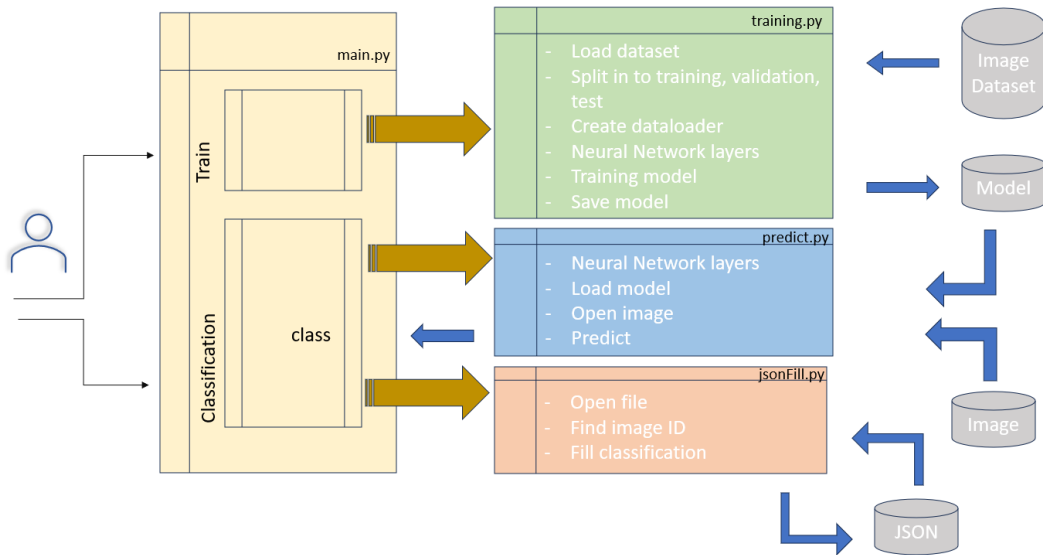


Figura 4.3: Fluxograma principal do modelo de ML

```

1 if __name__ == '__main__':
2     parser = argparse.ArgumentParser()
3     parser.add_argument('--image', type=str, help='enter the path
4       of the images to classify')
5     parser.add_argument('--training', type=str, help='training
6       model')
7     args = parser.parse_args()
8     main(args)
  
```

Código 4.4: Função seleção de modo no programa principal

4.3 Treino do Modelo

Conforme referido anteriormente, o modelo de treino é sinalizado pelo programa principal. Este começa por criar 2 vetores: o vetor imagens e o vetor *labels*.

Após esta criação, existe uma rotina que percorre todas as categorias do *dataset* e adiciona as imagens ao vetor imagens e no segundo vetor a *label* correspondente.

Na rotina seguinte, é criado o *dataset* propriamente dito. Este divide as imagens das categorias em 3 partes, sendo que para o treino são utilizadas 70% das imagens e as restantes 30% são divididas em partes iguais entre validação e teste.

Estas duas últimas são importantes, já que ao serem utilizadas durante o treino, permitem validar o ajuste dos pesos e avaliar a sua *performance*. Desta forma, o algoritmo consegue adaptar-se com base nas suas validações.

O processo de rotulagem também presente nesta rotina permite ao *dataset* recuperar o conjunto inicial de imagens e adicionar a cada uma delas uma *label*. Por norma, no treino de um modelo, pretendemos introduzir conjuntos de informações em mini-lotes. Deste modo, a reorganização e recuperação da informação em cada iteração do treino é acelerada. Consequentemente, *dataloader* é uma função que auxilia na abstração de criar um mini-lote a ser introduzido a cada iteração do treino.

Tratando-se de um *dataset* construído de raiz, é necessário ajustar alguns parâmetros com vista a que o treino convirja rapidamente. Assim sendo, estes parâmetros são ajustados com base nos valores de precisão e de perda retornados das iterações de treino. Se o tempo de treino for demasiado longo, o ajuste destes parâmetros e a aquisição de um modelo com a precisão pretendida pode demorar demasiado tempo. De forma a não ultrapassar o poder de processamento da máquina utilizada neste projeto, e segundo alguns estudos[72][73], concluiu-se que é possível treinar modelos fortes através da utilização de imagens mais pequenas de treino, reduzindo significativamente o tempo de treino. Por exemplo, em[72], obtiveram 77,1% de precisão usando o *dataset* ResNet-50, com imagens 128×128 pixels e 79,8% com imagens de 224×224 pixels.

Assim sendo, após alguns testes de treino, foi decidido reduzir o tamanho das imagens. Estas, que tinham inicialmente a resolução de 280x280, passaram para 64x64 pixels.

Para além do redimensionamento, as imagens foram normalizadas para escala de cinzentos, de modo a que todas as imagens apresentassem as mesmas características e, por sua vez, diminuir a quantidade de dados. Portanto, caso a imagem fosse a cores (RGB), cada pixel teria 3 Bytes. Sendo que na escala de cinzentos cada pixel terá apenas um Byte, este varia entre totalmente preto até totalmente branco.

De seguida, foi feita a estruturação das camadas da rede neuronal, utilizadas para treinar o modelo, como mostra no Código 4.5. Assim sendo, as camadas desta

rede foram selecionadas com base nos principais algoritmos direcionados para a classificação de imagem.

As camadas da rede neuronal são constituídas por uma camada convolucional, uma camada de ativação *Rectified Linear Unit* e uma camada totalmente conectada denominada por Linear.

```
1
2 class CustomModel(nn.Module):
3     def __init__(self, input_size, num_classes):
4         super(CustomModel, self).__init__()
5         self.conv1 = nn.Conv2d(1, input_size, kernel_size=3,
6                                 padding=1)
7         self.relu = nn.ReLU()
8         self.fc2 = nn.Linear(input_size * input_size * input_size,
9                               num_classes)
10
11     def forward(self, x):
12         x = self.conv1(x)
13         x = self.relu(x)
14         x = x.view(x.size(0), -1) # Flatten the feature map
15         x = self.fc2(x)
16         return x
```

Código 4.5: Função onde é construído o modelo da rede neuronal

As camadas encontram-se interligadas entre si. Consequentemente, o número de entradas de uma camada terá de ser igual ao número de saídas da camada anterior.

A primeira camada, a convolucional, definida pela função *Conv2D*, executa uma operação de convolução num determinado tensor. As *Convolutional Neural Networks* são muito utilizadas para o reconhecimento de imagem. Assim sendo, esta função recorre ao tensor de entrada e executa uma operação de convolução com o intuito de gerar um mapa de recursos.

O *kernel* é uma matriz pequena aplicada no tensor de entrada. O intuito desta é destacar os recursos que sobressaem da imagem de entrada. Esta janela pode ter a dimensão de 1x1, 3x3, 5x5 ou 7x7. Assim sendo, pelo facto das imagens utilizadas possuírem pequenos pormenores que as distinguem entre si, foi utilizado um *kernel* de dimensão 3x3, sendo este o mais usual neste tipo de aplicações.

No excerto de código acima (Código 4.5) é possível verificar que a função *Conv2D* é composta por 4 parâmetros. O primeiro identifica o número de componentes que constituem a imagem; como se trata de uma imagem na escala de cinzentos, este valor é de 1. O segundo parâmetro, denominado por *input_size*, representa o tamanho de entrada em pixels da imagem; inicialmente, foi utilizada a configuração 64x64 pixels. O parâmetro seguinte identifica o tamanho do *kernel*; conforme mencionado

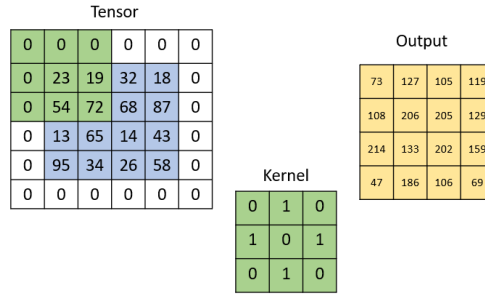


Figura 4.4: Operação convolução

anteriormente, foi escolhido o valor 3x3. Por último, o parâmetro *padding* adiciona zeros nas extremidades da imagem, com o objetivo de permitir ao *kernel* extrair a informação da imagem até às suas bordas. A função de convolução é definida pela seguinte expressão:

$$out(N_i, C_{outj}) = bias(C_{outj}) + \sum_{k=0}^{C_{in}-1} weight(C_{outj}, k) * input(N_i, k)$$

Nesta função, **N** representa a quantidade de informação inserida através dos mini-lotes; **C** determina o número de canais, tanto de saída como de entrada; *Bias* representa as *bias* aprendidas do formato dos canais de saída do modulo; e *Weight* determina os pesos aprendidos do formato dos canais de saída.

A segunda camada é uma função de ativação denominada por *Rectified Linear Unit* (ReLU). Esta camada aplicada após a convolução introduz não linearidade no modelo, permitindo que a rede aprenda e represente relações mais complexas nos dados. Sem essa não linearidade, a rede seria essencialmente um modelo linear, limitando a sua capacidade de aprender padrões complexos e hierárquicos em dados visuais. A ativação *ReLU* tem uma certa semelhança ao impulso nervoso dos neurónios no cérebro humano.

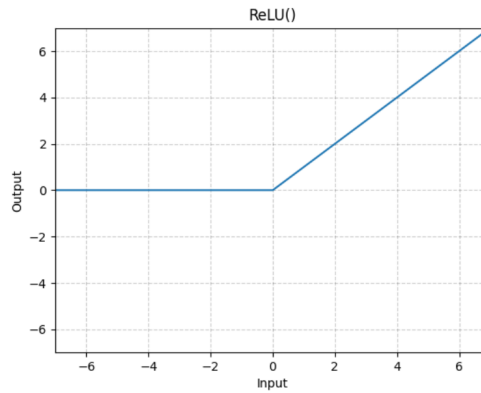


Figura 4.5: Representação gráfica da função ReLU

A principal função é retornar 0 (zero) caso receba um valor negativo na entrada.

Por outro lado, caso o valor seja positivo, devolve o mesmo valor de entrada. Assim sendo, se os valores negativos forem introduzidos no modelo, a capacidade de o modelo se ajustar e treinar adequadamente é diminuída.

Por fim, foi definida uma camada totalmente conectada, a *nn.Linear*, também conhecida por camada densa. Esta é um elemento fundamental em muitas arquiteturas de redes neurais. Matematicamente, esta função é definida por:

$$y = Wx + b$$

Nesta função, o y representa o tensor de saída, o x determina o tensor de entrada, o w representa a matriz de pesos e b determina o vetor de polarização.

O intuito principal desta função é realizar uma transformação linear nos dados de entrada e aplicar o conjunto de pesos e tendência aprendidos para produzir uma saída.

Após a estruturação da rede neuronal, foi iniciado o processo de treino do modelo. Durante este, os parâmetros do modelo são atualizados iterativamente com o intuito de minimizar a função de perdas. Estes parâmetros incluem pesos e *bias*. De seguida ao treino, esses parâmetros são guardados para serem utilizados em previsões futuras. Deste modo, para treinar o modelo foi utilizada uma função de perdas denominadas por *CrossEntropyLoss*, isto é, uma função projetada especificamente para problemas de classificação de várias classes, sendo aplicada quando se deseja medir a diferença de probabilidades de classes previstas e rótulos de classes reais. Esta função combina a ativação *softmax* e a função de perda *Negative Log-Likelihood*. Durante o treino, o objetivo é minimizar o *CrossEntropyLoss*, incentivando o modelo a fazer previsões corretas sobre as classes.

A função *softmax* recorre a um vetor de números reais para transformá-lo numa distribuição de probabilidade. Em suma, cada elemento de saída representa a probabilidade de pertencer a uma classe específica.

De seguida à aplicação *softmax*, a função de perda NLL calcula a probabilidade logarítmica negativa dos rótulos das classes verdadeiras. Essencialmente, isto resulta na análise que permite perceber a probabilidade de as classes previstas corresponderem aos rótulos de classe reais. Deste modo, a perda é maior quando as probabilidades previstas se encontram mais distantes dos rótulos reais da classe.

Portanto, é habitual combinar uma função de perda com um otimizador, pois juntos guiam eficazmente o modelo para melhorar a sua direção, convergência, estabilidade e robustez. Consequentemente, neste modelo foram experimentados 2 optimizadores: o *Adam* e o SGD. Neste caso, o *Adam* obteve melhores resultados e foi, por este motivo, o escolhido. Este apresentava uma evolução mais rápida que o SGD, reduzindo assim o número de iterações do treino (*epochs*).

O otimizador *Adam* apresenta como principais vantagens as suas taxas de aprendizagem adaptativa e a combinação de atualizações dinâmicas, permitindo o bom

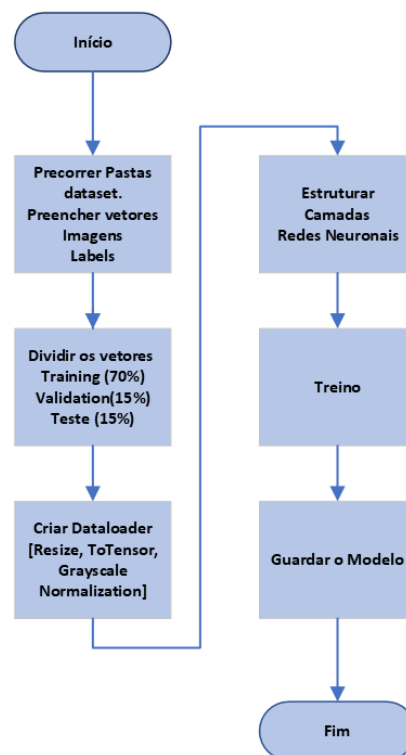


Figura 4.6: Training

desempenho numa ampla gama de tarefas de *Deep Learning*. Assim sendo, com o ajuste de parâmetros como o *learning rate*, foi possível ajustar o desempenho para obter aquele que seria ideal para a aplicação pretendida. O *learning rate* controla a magnitude dos ajustes dos pesos e dos *bias* a cada iteração do processo de treino. Se o *learning rate* for baixo, o algoritmo irá realizar atualizações menores, garantindo assim um processo de treino mais estável. No entanto, se o *learning rate* tiver um valor elevado, origina uma convergência mais rápida que pode ultrapassar os valores ideais dos parâmetros e resultar na divergência do processo de otimização. Deste modo, a escolha deste valor é uma das partes mais críticas do treino de modelos de *Deep Learning*, podendo variar entre 0,1 e 0,000001.

Terminado o treino deste modelo, este é guardado para ser carregado nas previsões futuras. Portanto, no módulo são guardados os parâmetros do modelo, a arquitetura do mesmo, o histórico de treino, os hiper-parâmetros, os pontos de verificação, as métricas de avaliação e as informações de pré-processamento.

Nos parâmetros do modelo estão incluídos os pesos e os *bias*, no caso das redes neurais, ou os coeficientes, no caso de outros modelos de *Machine Learning*.

Na arquitetura do modelo é guardada a estrutura do mesmo e a sua configuração, incluindo o número e tipo de camadas, as funções de ativação e quaisquer outros componentes que compõem o modelo.

O histórico de treino pode incluir a perda e a precisão de treino, assim como a

validação ao longo do tempo, podendo ser útil na visualização e análise do progresso do treino.

Os hiper-parâmetros são as configurações escolhidas antes do treino, como é o caso da taxa de aprendizagem, o tamanho do lote, o número de iterações (*epochs*) e quaisquer outras configurações que influenciam o processo de treino.

Os pontos de verificação permitem ajustar o modelo posteriormente ou continuar o treino a partir de um ponto específico, em vez de começar de novo. As métricas de avaliação são guardadas para os *datasets* de validação usados em aplicações específicas. Essas métricas fornecem *insights* sobre o desempenho do modelo em dados visuais não vistos.

No caso de existir algum pré-processamento de dados envolvido, como por exemplo, o dimensionamento de recursos, as informações sobre as etapas e os parâmetros de pré-processamento são guardados para garantir um pré-processamento de dados consistente durante a inferência.

4.4 Classificação de Componentes

As classificações são feitas com base num modelo pré-treinado. Como referido anteriormente, este programa recebe o caminho de uma imagem e devolve a classificação da classe que este previu que a imagem deve pertencer, como mostra a Figura 4.7.

Sendo carregado para este programa um modelo pré-treinado, este necessita que a estrutura das redes neuronais seja igual à do modelo pré-treinado. Deste modo, é inicialmente criado um modelo com as redes neuronais possuindo a mesma forma do modelo pré-treinado. Posteriormente, é aberta e carregada a imagem que de seguida irá ser processada. Este processamento é equivalente ao realizado anteriormente para as imagens do *dataset*.

Deste modo, a imagem é redimensionada, normalizada na escala de cinzas e convertida para tensor. Após o processamento da imagem, é realizada a previsão utilizando a função *softmax()*. Esta compara o tensor da imagem inserida com os *tensores* das classes do modelo pré-treinado, retornando o índice do valor mais alto, ou seja, retornando a classe a que a imagem pertence. Este valor da classe é retornado sucessivamente para o programa principal com o intuito de ser utilizado para preencher o módulo JSON.

4.5 Preenchimento do modelo de dados

Com o valor retornado do módulo anterior (*Predict*), o programa principal prossegue com o reenaminhamento do mesmo para o programa *jsonfill*. Assim sendo, o modelo JSON gerado pelo módulo de visão computacional[2] é preparado para a sua alteração.

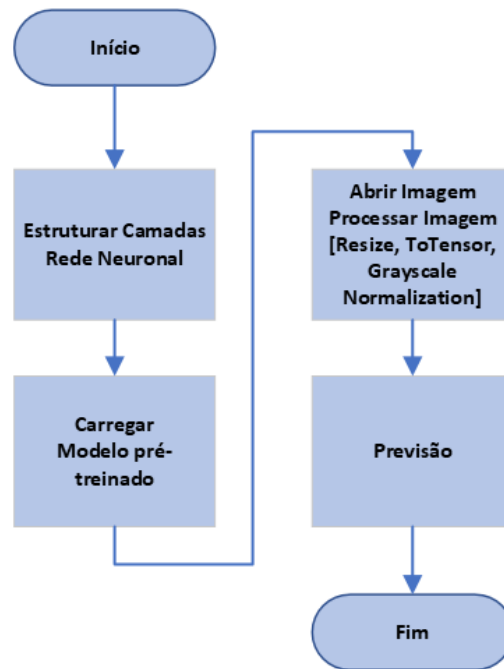


Figura 4.7: Fluxograma do módulo Predict

Neste modelo encontra-se toda a informação dos componentes existentes do circuito eletrônico segmentado, assim como dos nós e interligações entre esses componentes. Neste também se encontra registrada a posição global e relativa de cada componente no circuito.

Após a abertura do arquivo, procede-se à procura do identificador do componente, recorrendo a uma rotina. Sendo encontrado o identificador, preenche-se os valores em falta, como mostra a Figura 4.8. Neste caso, estes representam a classe e o identificador. Por exemplo, no caso de o componente ser uma resistência, serão preenchidos os campos *Component*, Resistência e R no campo *label*. Desta forma, são guardadas as alterações realizadas no arquivo e termina o programa.



Figura 4.8: Fluxograma do preenchimento dos dados no modelo JSON

Capítulo 5

Avaliação de resultados

Este capítulo pretende demonstrar inicialmente a ferramenta de análise usada para avaliar o modelo de classificação que basicamente, é uma tabela que mostra as frequências de classificação para cada umas das classes do modelo, conhecida por *matriz de confusão*. Após esta introdução serão apresentados os resultados obtidos, divididos em duas fases: a primeira referente aos resultados das classificações de imagens no processo de treino, recorrendo ao uso de imagens conhecidas pelo algoritmo para perceber se o seu processo de treino está a convergir; e a segunda referente aos resultados de teste do modelo, com o intuito de compreender a sua capacidade de classificar imagens novas.

5.1 Métrica de Avaliação

A *matriz de confusão* é uma tabela que permite extrair métricas para a avaliação dos modelos de classificação. As matrizes de confusão apresentam 4 resultados possíveis: Verdadeiro Positivo, Falso Negativo, Falso Positivo e Verdadeiro Negativo.

O Verdadeiro Positivo é o resultado de uma classificação correta para a classe correta. O Falso Positivo é o resultado de uma classificação correta para a classe errada. O Falso Negativo é o resultado de uma classificação errada para uma classe correta. Por fim, O Verdadeiro Negativo é o resultado de uma classificação errada para uma classe errada. Assim sendo, O Falso Positivo e o Falso Negativo são dos erros tipo 1 e tipo 2, respetivamente.

Desta tabela, podemos retirar métricas como a *exatidão*, responsável por medir a frequência com que o algoritmo faz uma previsão correta. Esta é a razão entre o número de previsões corretas e o número total de previsões.

$$Exatidao = \frac{VP + VN}{VP + VN + FP + FN}$$

Por sua vez, a *precisão* indica quantas previsões são realmente positivas em todo o total de previsões positivas, isto é, é a razão entre o número total de classes positivas classificadas corretamente e o total de classes classificadas de positivas.

$$Precisao = \frac{VP}{VP + FP}$$

A *sensibilidade* indica quantas previsões de classes positivas são realmente positivas, ou seja, é a razão entre o número total de classes positivas classificadas corretamente sobre o número total de classes classificadas corretamente.

$$Sensibilidade = \frac{VP}{VP + FN}$$

O *F1-score* é uma métrica que não é sensível a valores extremos (extremamente altos ou baixos), o que a torna muito útil. Como tal, esta resulta na média harmónica de precisão e sensibilidade.

$$F1 = 2 \times \frac{Sensibilidade \times Precisao}{Sensibilidade + Precisao}$$

Deste modo, este modelo apresenta percentualmente a possibilidade de a imagem analisada pertencer a cada uma das classes do modelo. Como tal, na tabela em anexo Figuras A.1 e A.2, por exemplo, são apresentadas todas as classificações realizadas com o intuito de construir a matriz de confusão. Após a extração dos dados, atingimos a matriz de confusão apresentada na Figura 5.1.

5.2 Resultados do Treino do Modelo

Nesta abordagem, foram utilizadas imagens redimensionadas com uma resolução de 64x64 pixels. Verificou-se que o treino foi realizado num total de 45 iterações, sendo que em cada iteração foram introduzidos mini-lotes de dados com o tamanho de 128 amostras. Como tal, obtiveram-se resultados com exatidão acima dos 94%. (A.1 A.2) As classes com a exatidão mais baixa são o Condensador, o Condensador Polarizado e a Fonte de corrente DC, por se tratarem de símbolos bastante semelhantes.

Com o intuito de testar e verificar a interferência dos vários parâmetros, foi decidido reduzir a quantidade de imagens do *dataset* para ser possível testar o modelo com imagens e quantidade de informação maiores. Isto permitiu perceber a evolução do algoritmo numa pequena amostra e depois replicá-la no modelo completo. Por

	Amperímetro	Bateria	Condensador	Condensador Polarizado	Pilha	Fonte C. AC	Fonte C. DC	Fonte C. Constante	GND Chave	GND Digital	GND Terra	Bobine	Lâmpada	Osciloscópio	Resistência EU	Resistência US	Interruptor SP	Interruptor 3P	Fonte T. AC	Fonte T. DC	Voltagem	Wattmetro
Amperímetro	4														1							
Bateria		4					1															
Condensador			2	2																	1	
Condensador Polarizado				3	2																	
Pilha					1	3														1		
Fonte C. AC						5																
Fonte C. DC							1	2						2								
Fonte C. Constante		1					1		3													
GND Chave									4	1												
GND Digital										4												
GND Terra			1								9						1					1
Bobine												6										
Lâmpada													4									1
Osciloscópio														5								
Resistência EU															5							
Resistência US			1													3	1					
Interruptor SP																	5					
Interruptor 3P															1			4				
Fonte T. AC																			5			
Fonte T. DC																				5		
Voltagem																					3	2
Wattmetro																						5

Figura 5.1: Demonstração de resultados com todas as classes

este motivo, foram escolhidas apenas 9 classes (Figura 5.2). Na escolha destas classes foram tidos em consideração os símbolos mais utilizados na iniciação da análise de circuitos.

As tabelas abaixo demonstram os resultados obtidos com as imagens de resolução 124x124 pixels e redução da quantidade dos mini-lotes, para 64. (A.3) Apesar disto, o número de iterações manteve-se, já que foi verificado que o modelo convergia de forma igual para o mesmo número de iterações.

	Condensador	Pilha	Fonte C. DC	GND Terra	Bobine	Resistência EU	Resistência US	Fonte T. AC	Fonte T. DC
Condensador	3				1		1		
Pilha		5							
Fonte C. DC		1	3						1
GND Terra	1			4					
Bobine				1	3		1		
Resistência EU						5			
Resistência US						1	4		
Fonte T. AC								5	
Fonte T. DC		1							4

Figura 5.2: Demonstração de resultados com redução de classes

O aumento das imagens resultou na alteração de alguns resultados. Assim sendo, verificou-se que a percentagem das classificações são mais elevadas, isto é, perto de 100%, e os resultados menos dispersos pelas classes. Na generalidade, verificou-se uma diminuição da exatidão, resultante da diminuição do número de amostras.

Com vista à obtenção de melhores resultados, foi decidido aumentar mais uma vez a imagem para uma resolução de 196x196 pixels e reduzir a quantidade de informação carregada a cada iteração através dos mini-lotes para o valor de 20. Apesar disto, o número de iterações manteve-se, isto é, foram utilizadas 45 iterações.

Tabela 5.1: Matriz de Confusão

Classe	Exatidão	Precisão	Sensibilidade	F1-Score
Amperímetro	98,18%	0,8	0,8	0,8
Bateria	98,18%	0,8	0,8	0,8
Condensador	94,54%	0,4	0,4	0,4
Condensador Pol	94,54%	0,4	0,4	0,4
Pilha	96,36%	0,6	0,6	0,6
Fonte Corrente AC	100%	1	1	1
Fonte Corrente DC	94,54%	0,4	0,4	0,4
Fonte Corrente Constante	96,36%	0,6	0,6	0,6
GND Chassi	98,18%	0,8	0,8	0,8
GND Digital	98,18%	0,8	0,8	0,8
GND Terra	96,36%	0,6	0,6	0,6
Bobine	100%	1	1	1
Lâmpada	98,18%	0,8	0,8	0,8
Osciloscópio	100%	1	1	1
Resistência EU	100%	1	1	1
Resistência US	96,36%	0,6	0,6	0,6
Interruptor 1P	100%	1	1	1
Interruptor 2P	98,18%	0,8	0,8	0,8
Fonte Tensão AC	100%	1	1	1
Fonte Tensão DC	100%	1	1	1
Voltímetro	96,36%	0,6	0,6	0,6
Wattímetro	100%	1	1	1

(A.4) Consequentemente, isto levou a que o processo de treino demorasse 4x mais tempo, devido ao facto da quantidade de informação de cada imagem ser maior.

Em suma, podemos verificar com as alterações realizadas que as classificações são ainda menos dispersas por entre as classes. Em relação aos dados obtidos, comparando as duas matrizes de confusão, podemos verificar um aumento da exatidão em duas classes, o Condensador e o GND Terra, e uma redução em outras duas, a Pilha e a Fonte de Tensão AC. As restantes classes mantiveram a exatidão.

Por outro lado, se compararmos os resultados com a primeira demonstração de resultados, que possuía todas as classes nos testes, verificamos que o aumento da resolução das imagens pouco interferia para uma melhor aprendizagem, concluindo-se assim que algumas dessas classes tinham resultados inferiores. Desta forma, foi utilizada a resolução de imagem inicial, 64x64 pixels, e mantida a quantidade de informação introduzida a cada iteração do treino através dos mini-lotes (128).

Portanto, para obter um resultado melhor, realizou-se o teste introduzindo um aumento do número de iterações. No teste inicial, com todas as classes, verificámos que esta solução não era vantajosa pois o treino acabava por convergir perto do valor de 45 iterações. Por vezes, quando usávamos um valor bastante elevado para o número de iterações, o treino parava, devolvendo uma mensagem que alertava que

Tabela 5.2: Matriz de confusão com redução de classes

Classe	Exatidão	Precisão	Sensibilidade	F1-Score
Condensador	91,11%	0,6	0,6	0,6
Pilha	100%	1	1	1
Fonte Corrente DC	91,11%	0,6	0,6	0,6
GND Terra	95,56%	0,8	0,8	0,8
Bobine	91,11%	0,6	0,6	0,6
Resistência EU	100%	1	1	1
Resistência US	95,56%	0,8	0,8	0,8
Fonte Tensão AC	100%	1	1	1
Fonte Tensão DC	95,56%	0,8	0,8	0,8

	Condensador	Pilha	Fonte C. DC	GND Terra	Bobine	Resistência EU	Resistência US	Fonte T. AC	Fonte T. DC
Condensador	4	1							
Pilha		4					1		
Fonte C. DC	1		3						1
GND Terra	1	1		3					
Bobine	1	1			3				
Resistência EU						5			
Resistência US							4		1
Fonte T. AC			1					2	2
Fonte T. DC				1					4

Figura 5.3: Demonstração de resultados com redução de classe e aumento de resolução

tinha sido atingido o limite de memória da máquina.

Quando aplicámos os mesmos parâmetros do teste inicial, tendo em consideração um menor número de imagens com redução a redução de classes, verificou-se que o treino do modelo já não convergia perto das 45 iterações. Após várias iterações concluímos que era possível aumentar significativamente o número de iterações. Foram necessários alguns treinos para tentar perceber qual o melhor valor deste parâmetro. No entanto, verificou-se que a convergência do treino variava com o aumento de iterações, o que não sucedia quanto estavam presentes todas as classes no processo de treino. Concluiu-se que o valor 45 era demasiado pequeno para o número de iterações. Devido ao facto de estar a ser usado um optimizador com *learning rate* adaptativo, ADAM [74], este adaptava-se para aprender detalhes maiores da imagem, verificando-se assim que o modelo tinha resultados satisfatórios quando apresentadas imagens do próprio *dataset* e resultados menos satisfatórios quando apresentadas imagens fora do *dataset*, ou seja, imagens de teste.

Nas tabelas abaixo, encontram-se demonstrados os resultados obtidos com 200 iterações no processo de treino.

Tabela 5.3: Matriz de confusão com redução de classes e aumento de resolução

Classe	Exatidão	Precisão	Sensibilidade	F1-Score
Condensador	95,56%	0,8	0,8	0,8
Pilha	95,56%	0,8	0,8	0,8
Fonte Corrente DC	91,11%	0,6	0,6	0,6
GND Terra	91,11%	0,6	0,6	0,6
Bobine	91,11%	0,6	0,6	0,6
Resistência EU	100%	1	1	1
Resistência US	95,56%	0,8	0,8	0,8
Fonte Tensão AC	86,67%	0,4	0,4	0,4
Fonte Tensão DC	95,56%	0,8	0,8	0,8

	Condensador	Pilha	Fonte C. DC	GND Terra	Bobine	Resistência EU	Resistência US	Fonte T. AC	Fonte T. DC
Condensador	5								
Pilha		5							
Fonte C. DC			4					1	
GND Terra	1			4					
Bobine	1			1	3				
Resistência EU						5			
Resistência US	1						4		
Fonte T. AC								4	1
Fonte T. DC							1		4

Figura 5.4: Demonstração de resultados com aumento de iterações

Portanto, com este ajuste de iterações, verificou-se um aumento de precisão nas classes Condensador, Pilha e Fonte de Corrente DC. Por outro lado, verificou-se uma diminuição em outras classes, nomeadamente na GND Terra, na Bobine, na Resistência US e nas Duas Fontes Tensão AC/DC. A queda mais significativa de precisão foi detetada na classe Bobine. Na tabela A.5 em anexo pode verificar-se que das classificações erradas, uma delas tem a percentagem maior na classe acima, ou seja GND Terra. Mas a segunda classificação errada tem a segunda percentagem maior na mesma classe. Apesar de uma das classificações estar completamente errada, a segunda, mesmo sendo errada tem uma percentagem considerável na classe correta.

Por fim, podemos chegar à conclusão que, com este ajuste final do número de iterações verificando a convergência do modelo, é possível obter uma resolução ótima da imagem, não sendo necessário sobrecarregar o modelo com imagens de resolução superior[72][73].

Tabela 5.4: Matriz de confusão com aumento de iterações

Classe	Exatidão	Precisão	Sensibilidade	F1-Score
Condensador	100%	1	1	1
Pilha	100%	1	1	1
Fonte Corrente DC	95,56%	0,8	0,8	0,8
GND Terra	95,56%	0,8	0,8	0,8
Bobine	91,11%	0,6	0,6	0,6
Resistência EU	100%	1	1	1
Resistência US	95,56%	0,8	0,8	0,8
Fonte Tensão AC	95,56%	0,8	0,8	0,8
Fonte Tensão DC	95,56%	0,8	0,8	0,8

5.3 Resultados do Teste do Modelo

Após chegar a um solução bastante aceitável dos resultados de treino, seguimos para o processo de teste, recorrendo a imagens exteriores ao *dataset* e verificando a capacidade de classificação do modelo.

Assim sendo, criou-se uma pequena amostra de imagens de teste. Esta amostra foi criada com 10 imagens de cada classe. Estas imagens (Tabela 5.5) de teste tiveram em consideração diferentes tamanhos e formas de desenho para que fossem minimamente diferentes entre si. Iniciámos com a mesma configuração dos dados de treino do último teste. Os resultados, como esperado, não foram satisfatórios nem comparáveis aos obtidos nos testes feitos com imagens do próprio *dataset*. (A.6 A.7)

Tabela 5.5: Matriz de confusão com imagens de teste

Classe	Exatidão	Precisão	Sensibilidade	F1-Score
Condensador	82,5%	0,3	0,3	0,3
Pilha	80%	0,2	0,2	0,2
Fonte Corrente DC	82,5%	0,3	0,3	0,3
GND Terra	77,5%	0,1	0,1	0,1
Bobine	77,5%	0,1	0,1	0,1
Resistência EU	85%	0,4	0,4	0,4
Resistência US	80%	0,2	0,2	0,2
Fonte Tensão AC	75%	0	0	0
Fonte Tensão DC	80%	0,2	0,2	0,2

Comparando a tabela anterior com a última obtida com as imagens de treino, observa-se que os valores da precisão, sensibilidade e F1-Score comprovam os resultados esperados.

A explicação para estes resultados deve-se ao facto do processo de treino convergir rapidamente. Esta conversão pode resultar de vários fatores, sendo o principal o tamanho do *dataset*, que apesar de permitir obter alguns resultados consideráveis com o número de imagens recolhidas, necessita de um tamanho superior para obter resultados melhores.

	Condensador	Pilha	Fonte C. DC	GND Terra	Bobine	Resistência EU	Resistência US	Fonte T. AC	Fonte T. DC
Condensador	4	3				3			
Pilha	6	1	1	1		1			
Fonte C. DC			2				1		6
GND Terra	5	1	2		1	1			
Bobine			2		2		3		
Resistência EU		2		2		5			1
Resistência US			1		2		6	1	
Fonte T. AC			5		1		3	1	
Fonte T. DC			4				3	1	2

Figura 5.5: Demonstração resultados Teste

Capítulo 6

Conclusões

No decorrer deste projeto foram adquiridas várias valências como resultado das exigências que o próprio trabalho reivindicou. Logo de início, para a construção de um *dataset*, foi necessário validar os símbolos, o que resultou no contacto com as entidades competentes que autenticaram a simbologia pretendida para esta aplicação.

No processo de construção deste *dataset*, foi indispensável formatar as imagens para que, desta forma, todo o *dataset* mantivesse a mesma estrutura. Assim sendo, foi necessário recorrer a ferramentas de processamento de imagem para recortar e fazer algum processamento de imagem como, por exemplo, redimensionar, transformar na escala de cinzas e fazer algumas rotações.

A tarefa na qual se despendeu mais tempo foi a implementação da rede neuronal no treino do modelo. Foi um processo iterativo em que se foi aprendendo à medida que as dificuldades o exigiam e os resultados obtidos não eram os esperados. No entanto, apesar de todos os desafios, foi possível desenvolver um modelo com resultados bastante satisfatórios, apresentando taxas de exatidão acima dos 90% e com tempos de teste bastante reduzidos. Isto deveu-se ao facto da estrutura ser baseada em *Convolutional Neural Networks*, responsáveis pela aprendizagem e análise dos pormenores através de pixels, ou seja, destas se adaptarem automaticamente às características evidentes das imagens, como bordas e formas. Por sua vez, isto reduziu significativamente a complexidade dos dados de entrada.

A utilização de um otimizador, como o ADAM [74], traduziu-se numa mais-valia no processo de treino, apresentando este a capacidade de ajustar a sua taxa de aprendizagem, denominada de *learning rate* e, assim, reduzir o tempo de treino. Em

suma, a aprendizagem computacional (*Machine Learning*) evoluiu nos últimos anos devido ao desenvolvimento da computação. Consequentemente, este foi um tema bastante cativante de desenvolver, uma vez que permitiu aprender e evoluir desde o primeiro até ao último dia do projeto.

No entanto, no decorrer deste projeto, foram encontradas várias dificuldades, nomeadamente a falta de um *dataset* que cumprisse os requisitos pretendidos. Os *datasets* existentes, nomeadamente em plataformas como o Kaggle [75], têm a vantagem de serem bem organizados, mais limpos e compostos por enormes quantidades de dados, resultando num aumento da qualidade e robustez de treino dos modelos.

Por outro lado, foi sentida também a dificuldade de não encontrar documentação clara para integrar o próprio *dataset* no código. A maioria da informação encontrada é baseada em *datasets* pré-existentes, como o caso do MNIST [76], CIFAR10 [77] e ImageNet [78], e com imagens mais comuns, como por exemplo animais, automóveis e dígitos que não faziam parte do propósito deste projeto. Assim sendo, para estes conjuntos de dados existem funções específicas que possibilitam efetuar o *download* e carregar diretamente o *dataset* apenas numa linha de código.

Conforme referido anteriormente, a aquisição de imagens no âmbito da construção do *dataset* foi um processo mais demorado que o previsto, tendo sido necessário validar os símbolos normalizados e não os geralmente utilizados. Contudo, o maior tempo despendido decorreu na tarefa de recolha entre os alunos, já que esta foi efetuada durante as aulas laboratórios onde estes se encontram em grupos menores.

Na aplicação do *dataset* com as imagens recolhidas entre os alunos foram descobertos alguns erros no processo de rotulagem do algoritmo. Desta forma, a parte inicial do software exigiu uma reconstrução total, que atrasou também no desenvolvimento do projeto.

O processo de ajuste de parâmetros foi também uma parte crucial no desenvolvimento da aplicação. Os tempos de treino eram longos, sendo necessário terminar um treino, ajustar parâmetros e testar novamente.

Tendo em vista um trabalho futuro, seria de interesse recorrer à utilização de uma máquina com maior poder de processamento, que possibilitasse usar processamento paralelo e, assim, efetuar o teste de ferramentas, como o caso do CUDA [37], e perceber o impacto no modelo pré-treinado. Com isto, seria possível comparar os resultados com os obtidos desta dissertação.

Por outro lado, seria também interessante utilizar outra plataforma como, por exemplo, o Databricks [79]. Esta é uma solução de computação em *cloud* que recorre a *clusters* para efetuar o processamento, transformação e exploração de grandes volumes de dados. Foi utilizada a plataforma Google Colab [80], semelhante à anterior, para realizar alguns testes. Verificou-se que os tempos de treinos eram maiores. Contudo, existia a possibilidade de adicionar mais dados, ou seja, mais imagens e resoluções maiores. Em contrapartida, muitas vezes os treinos eram suspensos por

motivos de segurança da plataforma, pois esta detetava inatividade e suspendia o treino para não gastar recursos.

De forma a complementar o projeto, existem algumas tarefas que poderiam ainda ser implementadas, como é o caso da classificação das *labels* de cada componente. Deste modo, o processo de interpretação e modelação de circuitos elétricos ficaria mais completo.

Por fim, seria também de interesse implementar um algoritmo recorrendo à ferramenta YOLO [39]. Apesar de, inicialmente, este não parecer integrar os objetivos pretendidos, veio a tornar-se uma ferramenta muito útil para analisar o problema de outra perspectiva. Em suma, esta ferramenta apresenta a capacidade de identificar vários objetos numa imagem.

Referências

- [1] L. Sousa, “U=risolve simulator.” <https://urisolve.pt/app/>, Setembro 2023. [Citado nas páginas i, iii, 1 e 36]
- [2] H. Barbosa, “Interpretação e modelação de esquemas elétricos com recurso a algoritmos de visão computacional, tese de mestrado em engenharia eletrotécnica e computadores, isep.” <http://hdl.handle.net/10400.22/21717>, 2022. [Citado nas páginas i, iii, 1, 2, 3, 45 e 52]
- [3] L. Sousa, M. Alves, and F. Pererira, “U=risolve: teaching & self-learning the nodal voltage analysis method the easy way.” https://www.isep.ipp.pt/files/CASHE_Talk11.pdf, February 2019. [Citado na página 1]
- [4] H. F. Mahmood, “What are digital images.” <https://www.educative.io/answers/what-are-digital-images>, 2023. [Citado na página 5]
- [5] T. Tech, “Concept of pixel.” <https://www.javatpoint.com/concept-of-pixel>, 2023. [Citado nas páginas vii e 6]
- [6] Datagen, “Image segmentation: The basics and 5 key techniques.” <https://datagen.tech/guides/image-annotation/image-segmentation/>, 2023. [Citado na página 6]
- [7] G. Landini, D. Randell, S. Fouad, and A. Galton, “Automatic thresholding from the gradients of region boundaries,” *Journal of microscopy*, vol. 265, 09 2016. [Citado nas páginas vii e 8]
- [8] J. FRANKENFIELD, “Artificial intelligence: What it is and how it is used.” <https://www.investopedia.com/terms/a/artificial-intelligence-ai.asp>, 2023. [Citado na página 8]
- [9] B. Liu, “”weak ai” is likely to never become ”strong ai”, so what is its greatest value for us,” 2021. [Citado na página 8]
- [10] M. Works, “What is machine learning.” <https://www.mathworks.com/discovery/machine-learning.html>, 2023. [Citado nas páginas vii, 9 e 10]
- [11] R. Y. Choi, A. S. Coyner, J. Kalpathy-Cramer, M. F. Chiang, and J. P. Campbell, “Introduction to Machine Learning, Neural Networks, and Deep Learning,”

- Translational Vision Science & Technology*, vol. 9, pp. 14–14, 02 2020. [Citado nas páginas 10, 13, 14, 17 e 20]
- [12] G. Sacramento, “Naive bayes: Como funciona esse algoritmo de classificação.” <https://blog.somostera.com/data-science/naive-bayes>, 2023. [Citado na página 10]
- [13] F.-J. Yang, “An implementation of naive bayes classifier,” in *2018 International Conference on Computational Science and Computational Intelligence (CSCI)*, pp. 301–306, 2018. [Citado na página 10]
- [14] J. Sun, W. Du, and N. Shi, “A survey of knn algorithm,” *Information Engineering and Applied Computing*, vol. 1, 05 2018. [Citado na página 11]
- [15] R. Shah, “Introduction to k-nearest neighbors (knn) algorithm.” <https://ai.plainenglish.io/introduction-to-k-nearest-neighbors-knn-algorithm-e8617a448fa8>, 2023. [Citado nas páginas vii e 11]
- [16] Raman, “ML | linear discriminant analysis.” <https://www.geeksforgeeks.org/ml-linear-discriminant-analysis/>, 2023. [Citado nas páginas vii e 12]
- [17] E. I. G. Nassara, E. Grall-Maës, and M. Kharouf, “Linear discriminant analysis for large-scale data: Application on text and image data,” in *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 961–964, 2016. [Citado na página 12]
- [18] Oracle, “Oracle - generalized linear model,” September 2023. [Accessed 10-03-2024]. [Citado na página 13]
- [19] M. Müller, “Generalized linear models,” 02 2004. [Citado na página 13]
- [20] S. Tong and D. Koller, “Support vector machine active learning with applications to text classification,” *The Journal of Machine Learning Research*, vol. 2, pp. 45–66, 12 2001. [Citado na página 14]
- [21] M. Kalirane, “Ensemble learning methods: Bagging, boosting and stacking.” <https://www.analyticsvidhya.com/blog/2023/01/ensemble-learning-methods-bagging-boosting-and-stacking/>, 2023. [Citado nas páginas vii, 14, 15 e 16]
- [22] N. Kieu Anh, W. Chen, B.-S. Lin, and U. Seeboonruang, “Comparison of ensemble machine learning methods for soil erosion pin measurements,” *ISPRS International Journal of Geo-Information*, vol. 10, p. 42, 01 2021. [Citado na página 15]

-
- [23] IBM, “What is a neural network.” <https://www.ibm.com/topics/neural-networks>, Junho 2023. [Citado na página 16]
- [24] C. Sekhar and P. Meghana, “A study on backpropagation in artificial neural networks,” *Asia-Pacific Journal of Neural Networks and Its Applications*, vol. 4, pp. 21–28, 08 2020. [Citado na página 17]
- [25] Z. Lipton, “A critical review of recurrent neural networks for sequence learning,” 05 2015. [Citado na página 17]
- [26] Q. Su, C. Kloukinas, and A. d’Avila Garcez, “Modular neural networks for time series forecasting: Interpretability and feature selection using attention,” 2023. [Citado na página 18]
- [27] S. Naeem, A. Ali, S. Anam, and M. Ahmed, “An unsupervised machine learning algorithms: Comprehensive review,” *IJCDS Journal*, vol. 13, pp. 911–921, 04 2023. [Citado nas páginas 18 e 19]
- [28] MathWorks, “Types of machine learning models.” <https://www.mathworks.com/discovery/machine-learning-models.html>, Junho 2023. [Citado na página 18]
- [29] Y. Khalifa, D. Mandic, and E. Sejdić, “A review of hidden markov models and recurrent neural networks for event detection and localization in biomedical signals,” *Information Fusion*, vol. 69, pp. 52–72, 2021. [Citado na página 20]
- [30] MathWorks, “What is deep learning? 3 things you need to know.” <https://www.mathworks.com/discovery/deep-learning.html>, Junho 2023. [Citado na página 21]
- [31] Z. Hao, “Deep learning review and discussion of its future development,” *MATEC Web of Conferences*, vol. 277, p. 02035, 01 2019. [Citado na página 21]
- [32] J. li, A. Sun, R. Han, and C. Li, “A survey on deep learning for named entity recognition,” 12 2018. [Citado na página 21]
- [33] P. Ballester and R. Araujo, “On the performance of googlenet and alexnet applied to sketches,” 02 2016. [Citado na página 22]
- [34] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems* (F. Pereira, C. Burges, L. Bottou, and K. Weinberger, eds.), vol. 25, Curran Associates, Inc., 2012. [Citado na página 22]
- [35] M. Pietroni and D. Žurek, “When deep learning models on gpu can be accelerated by taking advantage of unstructured sparsity,” 2021. [Citado na página 22]

-
- [36] NVIDIA, “Nvidia ai platform for developers.” <https://developer.nvidia.com/deep-learning>, Agosto 2023. [Citado na página 22]
- [37] NVIDIA, “Develop, optimize and deploy gpu-accelerated apps.” <https://developer.nvidia.com/cuda-toolkit>, Agosto 2023. [Citado nas páginas 23 e 64]
- [38] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” 2016. [Citado na página 23]
- [39] G. Boesch, “Yolov7: The most powerful object detection algorithm.” <https://viso.ai/deep-learning/yolov7-guide/>, Agosto 2023. [Citado nas páginas 23 e 65]
- [40] R. Kundu, “Yolo: Algorithm for object detection.” <https://www.v7labs.com/blog/yolo-object-detection>, Agosto 2023. [Citado na página 23]
- [41] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, “Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors,” 2022. [Citado na página 23]
- [42] R. Project, “Circuit recognition dataset.” <https://universe.roboflow.com/rp-project/circuit-recognition>, may 2023. visited on 2023-09-27. [Citado nas páginas vii, 23 e 36]
- [43] RoboflowUniverse, “Open source computer vision community.” <https://universe.roboflow.com/>. [Citado na página 24]
- [44] F. Ciaglia, F. S. Zuppichini, P. Guerrie, M. McQuade, and J. Solawetz, “Roboflow 100: A rich, multi-domain object detection benchmark,” 2022. [Citado na página 25]
- [45] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, “Microsoft coco: Common objects in context,” 2015. [Citado na página 26]
- [46] L. Wood and F. Chollet, “Efficient graph-friendly coco metric computation for train-time model evaluation,” 2022. [Citado na página 26]
- [47] NVidia, “What is edge ai and how does it work.” <https://blogs.nvidia.com/blog/2022/02/17/what-is-edge-ai/>, Agosto 2023. [Citado na página 27]
- [48] “IEC 60617:2012 DB | IEC Webstore — webstore.iec.ch.” <https://webstore.iec.ch/publication/2723>. [Accessed 03-12-2023]. [Citado na página 30]

- [49] M. Moetesum, S. Younus, M. Warsi, and I. Siddiqi, “Segmentation and recognition of electronic components in hand-drawn circuit diagrams,” *ICST Transactions on Scalable Information Systems*, vol. 5, p. 154478, 04 2018. [Citado nas páginas 31 e 32]
- [50] Y. Liu and Y. Xiao, “Circuit sketch recognition.” https://stacks.stanford.edu/file/druid:bf950qp8995/Liu_Xiao.pdf. Accessed: 2023-11-8. [Citado na página 32]
- [51] P. Sala, “A recognition system for symbols of electronic components in hand-written circuit diagrams,” *ResearchGate*, 10 2023. [Citado na página 32]
- [52] M. Gunay and M. Koseoglu, “Detection of circuit components on hand-drawn circuit images by using faster r-cnn method,” *International Advanced Researches and Engineering Journal*, vol. 5, pp. 372–378, 12 2021. [Citado na página 33]
- [53] F. Thoma, J. Bayer, and Y. Li, “A public ground-truth dataset for handwritten circuit diagram images,” 2021. [Citado na página 34]
- [54] M. Dey, S. Mia, N. Sarkar, A. Bhattacharya, S. Roy, S. Malakar, and R. Sarkar, “A two-stage cnn-based hand-drawn electrical and electronic circuit component recognition system,” *Neural Computing and Applications*, vol. 33, 10 2021. [Citado na página 34]
- [55] M. Gunay, M. Koseoglu, and Yildirim, “Classification of hand-drawn basic circuit components using convolutional neural networks,” 06 2020. [Citado na página 35]
- [56] A. Kumar and R. Merchant, “Detection of hand-drawn circuit schematic symbols,” *Stanford CS229 Project Report*, 2021. [Citado na página 35]
- [57] M. Gody, “Hand-drawn electric circuit schematic components,” Sep 2022. [Citado na página 36]
- [58] M. McBride, “Key advantages of numpy.” <https://www.pythoninformer.com/python-libraries/numpy/advantages/>, Agosto 2023. [Citado na página 39]
- [59] J. A. Clark and contributors, “Python imaging library.” <https://pillow.readthedocs.io/en/stable/>. [Citado na página 40]
- [60] G. Boesch, “What is opencv? the complete guide (2023).” <https://viso.ai/computer-vision/opencv/>, Agosto 2023. [Citado na página 40]
- [61] OpenCV, “Industrial inspection.” <https://www.opencvhelp.org/tutorials/applications/industrial-inspection/>. [Citado na página 40]

-
- [62] Tensorflow, “Solve real-world problems with ml.” <https://www.tensorflow.org/>. [Citado na página 41]
- [63] Google, “Introdução ao tensorflow.” <https://developers.google.com/machine-learning/crash-course/first-steps-with-tensorflow/toolkit?hl=pt-br>, Agosto 2023. [Citado nas páginas vii e 42]
- [64] Keras, “Deep learning for humans.” <https://keras.io/>. [Citado na página 41]
- [65] Pytorch, “The pytorch foundation is a neutral home for the deep learning community.” <https://pytorch.org/>. [Citado na página 43]
- [66] K. Yasar, “What is pytorch.” <https://www.techtarget.com/searchenterpriseai/definition/PyTorch>, Agosto 2023. [Citado na página 43]
- [67] Pytorch, “End-to-end machine learning framework.” <https://pytorch.org/features/>, Agosto 2023. [Citado nas páginas vii e 43]
- [68] NVidia, “What is pytorch.” <https://www.nvidia.com/en-us/glossary/data-science/pytorch/>, Agosto 2023. [Citado na página 43]
- [69] S. Naim, B. Caffo, H. I. Sair, and C. K. Jones, “Applications of sequential learning for medical image classification,” 2023. [Citado na página 43]
- [70] L. Polson, R. Fedrigo, C. Li, M. Sabouri, O. Dzikunu, S. Ahamed, A. Rahmim, and C. Uribe, “Pytomography: A python library for quantitative medical image reconstruction,” 2023. [Citado na página 43]
- [71] Y. Wang, C. Zhou, S. ShuChun, and H. Li, “Unsupervised cnn-based dic for 2d displacement measurement,” 2023. [Citado na página 44]
- [72] H. Touvron, A. Vedaldi, M. Douze, and H. Jégou, “Fixing the train-test resolution discrepancy,” 2022. [Citado nas páginas 47 e 60]
- [73] A. Bakhtiarnia, Q. Zhang, and A. Iosifidis, “Efficient high-resolution deep learning: A survey,” 2022. [Citado nas páginas 47 e 60]
- [74] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2017. [Citado nas páginas 59 e 63]
- [75] Kaggle, “Level up with the largest ai & ml community.” <https://www.kaggle.com/>. [Citado na página 64]
- [76] Y. LeCun, “The mnist database of handwritten digits.” <http://yann.lecun.com/exdb/mnist/>. [Citado na página 64]

-
- [77] V. N. Alex Krizhevsky and G. Hinton, “Cifar-10 there are 50000 training images and 10000 test images.” <https://www.cs.toronto.edu/~kriz/cifar.html>. [Citado na página 64]
- [78] P. U. Stanford Vision Lab, Stanford University, “Imagenet is an image database organized according to the wordnet hierarchy.” <https://www.image-net.org/>. [Citado na página 64]
- [79] A. Spark, “Databricks is the data and ai company.” <https://www.image-net.org/>. [Citado na página 64]
- [80] Google, “Google colaboratory.” <https://colab.google/>. [Citado na página 64]

Anexo A

Tabelas com classificações em todas as classes

[illegible]

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
11	0	0	0	0	0	0	0	0	0	0	0	99.54	0	0	0	0.12	0	0.03	0	0	0	0
12	0	1.13	0	0	2.28	0.06	0	0	0	0.02	0.3	99.97	20.34	0	0	0.03	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	1.39	0	0	0.01	0.07	0	0.01	0.01	0	0.01	0	0	94.2	0.95	0.01	0	0	0.03	0	0	0.02	4.15
15	0	0	0	0	0	0	0	0	0	0	0	0	99.17	0	0	0	0	0	0	0	0	0
16	0.13	0.06	0.02	0	0.55	0.02	0	9.97	0	0.01	0	0	15.02	1.33	0.15	4.63	0.03	0.15	0.03	50.49	17.41	0
17	0	0	0	0	0	0	0	0	0	0	0	0	99.34	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	99.56	100	0	0	0	0	0	0	0	0
19	0.09	0	0	0.09	0	0.2	0.01	0.02	0	0	0	0	99.35	99.35	0	0	0	0	0.03	0	0	0
20	0.65	0	0	0	0	0	0	0	0	0	0	0	99.35	99.35	0	0	0	0	0	0	0	0
21	0.33	0.01	0.03	0	0.02	0.04	0.02	0.02	0.05	0	0.03	0.09	0.01	99.08	0.12	0	0.02	0.06	0.07	0.06	0	0
22	0.02	0.03	0.01	0.02	0.02	0.01	0.02	0.02	0.02	0.03	0.01	0.03	0.02	99.12	0.02	0.02	0.02	0.14	0.02	0	0.04	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
35	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
37	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
38	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
39	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
41	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
42	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
43	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
44	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
45	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
46	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
47	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
48	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
49	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
50	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
51	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
52	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
53	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
54	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
55	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
56	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
57	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
58	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
59	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
60	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
61	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
62	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
63	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
64	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
65	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
66	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
67	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
68	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
69	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
70	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
71	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
72	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
73	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
74	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
75	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
76	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
77	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
78	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
79	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
80	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
81	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
82	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
83	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
84	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
85	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
86	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
87	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
88	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
89	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
90	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
91	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
92	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
93	0	0	0	0																		

		0	1	2	3	4	5	6	7	8
2	1	96,97	0	0,21	0,82	0	0	0	0	0
	2	0	0	0	0	100	0	0	0	0
	3	23,84	0	20,16	0	0,31	0	55,69	0	0
	4	100	0	0	0	0	0	0	0	0
	5	100	0	0	0	0	0	0	0	0
4	1	0	100	0	0	0	0	0	0	0
	2	0	100	0	0	0	0	0	0	0
	3	0	100	0	0	0	0	0	0	0
	4	0	100	0	0	0	0	0	0	0
	5	0	100	0	0	0	0	0	0	0
6	1	0	0	100	0	0	0	0	0	0
	2	0	99,88	0,12	0	0	0	0	0	0
	3	0	0	100	0	0	0	0	0	0
	4	0	0	100	0	0	0	0	0	0
	5	0	0,12	0	0	0	0	0	99,88	0
10	1	0	0	0	100	0	0	0	0	0
	2	0	0	0	100	0	0	0	0	0
	3	0	0	0	100	0	0	0	0	0
	4	0	0	0	100	0	0	0	0	0
	5	100	0	0	0	0	0	0	0	0
11	1	0	0	0	0	100	0	0	0	0
	2	0	0	0	0	100	0	0	0	0
	3	2,29	0	0	0	35,89	0,43	61,39	0	0
	4	0	0	0	0	100	0	0	0	0
	5	0	0	0	68,7	30,23	0	0	1,07	0
14	1	0	0	0	0	0	100	0	0	0
	2	0	0	0	0	0	100	0	0	0
	3	0	0	0	0	0	100	0	0	0
	4	0	0	0	0	0	100	0	0	0
	5	0	0	0	0	0	100	0	0	0
15	1	0	0	0	0	0	0	100	0	0
	2	0	0	0	0	0	0	100	0	0
	3	0	0	0	0	0	0	100	0	0
	4	0	0	0	0	0	99,99	0	0	0,01
	5	0	0	0	0	0	0,12	99,88	0	0
18	1	0	0	0	0	0	0	0	100	0
	2	0	0	0	0	0	0	0	100	0
	3	0	0	0	0	0	0	0	100	0
	4	0	0	0	0	0	0	0	100	0
	5	0	0	0	0	0	0	0	100	0
19	1	0	0	0	0	0	0	0	0	100
	2	0	0	0	0	0	0,04	1,61	0	98,35
	3	0	95,66	0	0	0	0	0	0	4,34
	4	0	0	0	0	0	0	0	0	100
	5	0	0	0	0	0	0	0	0	100

Figura A.3: Demonstração resultados com redução classes IP:124;
BS:64; Epochs:45

		0	1	2	3	4	5	6	7	8
2	1	100								
	2		100							
	3	100								
	4	100								
	5	100								
4	1		100							
	2		100							
	3		100							
	4						100			
	5		100							
6	1			100						
	2	100								
	3			100						
	4			100						
	5									100
10	1				100					
	2				100					
	3				100					
	4		100							
	5	100								
11	1					100				
	2					100				
	3	100								
	4					100				
	5		100							
14	1					100				
	2					100				
	3					100				
	4					100				
	5					100				
15	1						100			
	2	0.99					99.01			
	3						100			
	4									100
	5						100			
18	1									100
	2		100							
	3							100		
	4									100
	5							100		
19	1									100
	2			97.75						2.25
	3									100
	4									100
	5									100

Figura A.4: Demonstração resultados com redução classes IP:196;
BS:20; Epochs:45

		0	1	2	3	4	5	6	7	8
1	1	100								
	2	98,16	0,07			0,85		0,8		0,11
	3	87,85	0,01	0,03	0	1,05	0	10,89	0	0,16
	4	99,99	0,01							
	5	100								
2	1		99,99						0,01	
	2	0,02	99,98							
	3		99,99							
	4		95,74					0,39		3,87
	5		100							
4	1			100						
	2	0,02	6,09	94,47	0,53		0,1		0,63	0,17
	3			99,99				0,01		
	4			100						
	5	0,01	2,97	11,18	8,34	0,89	5,92	2,71	50,68	17,29
6	1				100					
	2				100					
	3				100					
	4				99,99	0,01				
	5	99,98				0,01				
10	1					100				
	2					100				
	3	59,4	0,01			37,49	0,04	3,06		
	4					100				
	5		16,76	0,01	82,72	0,5				
11	1						100			
	2						100			
	3						100			
	4						100			
	5						100			
14	1	0,01						99,99		
	2	0,55						99,41		
	3	0,01	0,01					99,98		
	4	93,94		0,01			0,01			6,04
	5	0,33	0,23			0,08	0,75	98,17		0,43
15	1								0,02	99,98
	2								100	
	3								100	
	4								100	
	5								100	
18	1									100
	2	5,73	0,14	0,01	0,27	9,84	1,23	43,82	0,06	38,89
	3		43,58		2,67	0,07		0,02	0,01	53,64
	4							0,01		99,99
	5									100
19	1									
	2									
	3									
	4									
	5									

Figura A.5: Demonstração resultados com redução classes IP:64;
BS:128; Epochs:200

	0	1	2	3	4	5	6	7	8	TP	FP	FN	TN	Accuracy	Precision	Recall	F1-Score
1	0	99.98	0	0	0	0.02	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0
5	0.01	99.97	0	0.02	0	0	0	0	0	4	6	6	64	0.85	0.4	0.4	0.4
6	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	99.97	0.03	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	99.26	0	0	0	0	1.74	0	0	0	0	0	0	0	0	0	0	0
9	99.29	0.01	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	93.21	0	0	0	6.79	0	0	0	0	0	0	0	0	0	0	0
11	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	99.88	0.12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	98.56	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	1.44	0	0	0	0	0	0	0	0	0	0	0
17	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	45.44	15	0	0	0	53.06	0	0	0	0	0	0	0	0	0	0	0
19	6.96	93.04	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0.01	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25	0	0	0.27	0	0	0	99.72	0	0	0	0	0	0	0	0	0	0
26	0	0	5.12	0	0.95	0	0	0.1	93.92	2	8	8	62	0.8	0.2	0.2	0.2
27	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0
28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
30	0	0	99.42	0	0	0	0	0	0	0	0	0	0	0	0	0	0
31	0	0	0	0	0	0	0	0.58	0	0	0	0	0	0	0	0	0
32	99.99	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0
33	100	0	0	0	0	0.01	0	0	0	0	0	0	0	0	0	0	0
34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
35	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
36	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
37	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0
38	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0
39	77.64	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0
40	100	0	0	0	0	22.36	0	0	0	0	0	0	0	0	0	0	0
41	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
42	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0
43	0	0	0	0	0.13	95.79	0	0	4.21	0	0	0	0	0	0	0	0
44	0	0	0	100	0	0	0	0	99.87	0	0	0	0	0	0	0	0
45	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
46	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0
47	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
48	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0
49	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
50	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0
51	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
52	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
53	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
54	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
55	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
56	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
57	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
58	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
59	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
60	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figura A.6: Demonstração resultados Teste - parte 1 IP:64; BS:32;
Epochs:200

14	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
----	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

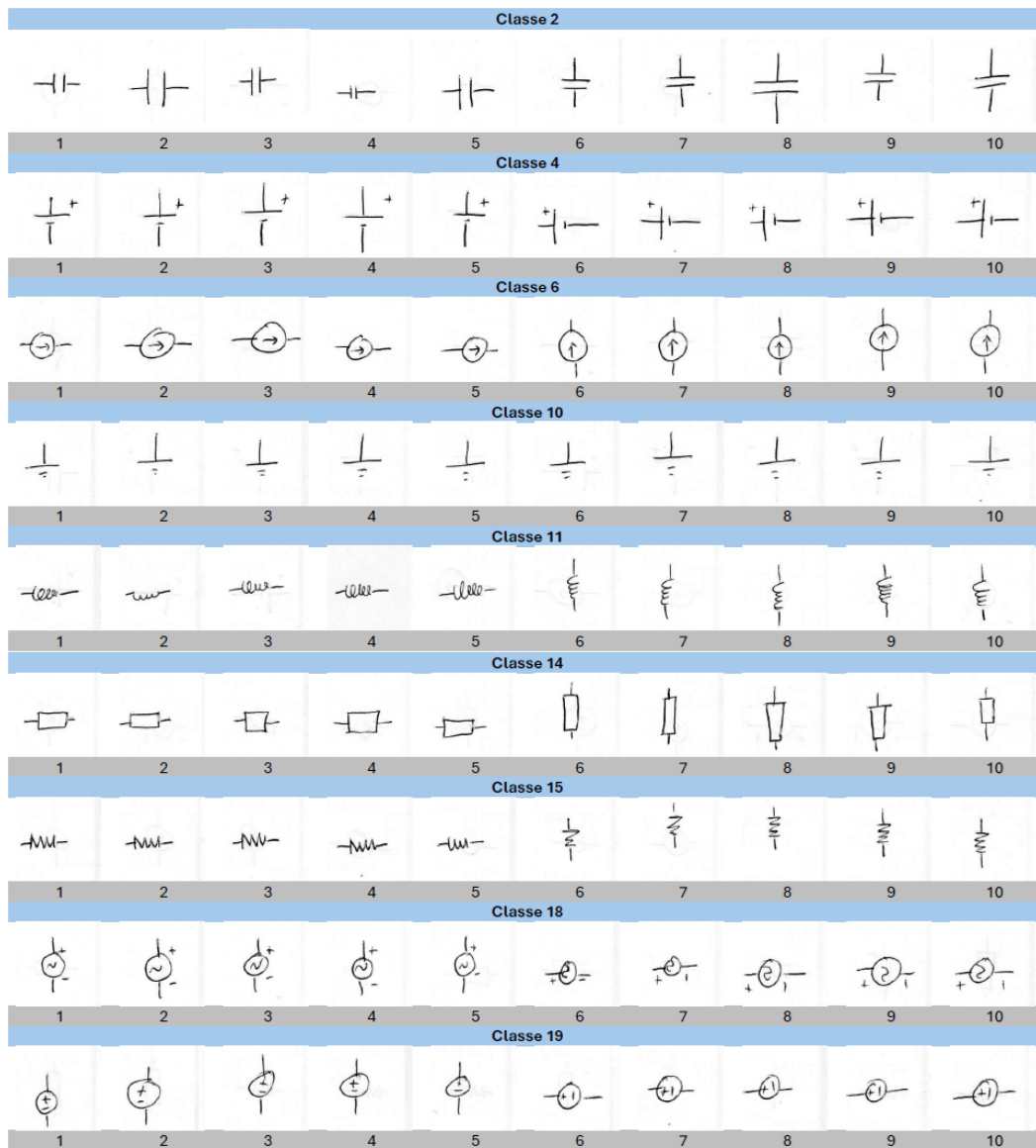


Figura A.8: Conjunto de imagens-teste

Anexo B

Modelo de Dados JSON

Apresenta-se neste anexo a estrutura de dados em JSON, para um exemplo de segmentação de um circuito elétrico.

```
1  {
2    "components": [
3      {
4        "id": "5c426217-3511-4363-96e4-e4be6620c5da",
5        "type": "",
6        "fullName": "",
7        "label": {
8          "id": "93d5fd5c-39f7-4211-841d-5ee1e1e998bf",
9          "owner": "5c426217-3511-4363-96e4-e4be6620c5da",
10         "name": "",
11         "value": "",
12         "unit": "",
13         "position": {
14           "x": 0,
15           "y": 0,
16           "angle": 0
17         },
18         "isNameHidden": true,
19         "isValueHidden": true
```



```

20         },
21         "ports": [
22             {
23                 "id": "e63e7320-5565-4182-bfbb-
24                     c43478e5d1b7",
25                 "owner": "5c426217-3511-4363-96e4-
26                     e4be6620c5da",
27                 "type": "hybrid",
28                 "position": {
29                     "x": 0.5,
30                     "y": 0.0,
31                     "angle": 0
32                 },
33                 "connection": "74a67485-e333-4b2e-be1e-
34                     aa4f50787bc5"
35             }
36         ],
37         "position": {
38             "x": 155,
39             "y": 494,
40             "angle": 0
41         }
42     },
43     {
44         "id": "2a5e46a1-2304-4cd4-869a-7566d8ad2d2f",
45         "type": "",
46         "fullName": "",
47         "label": {
48             "id": "80833ca3-e30a-44af-8382-a64fda2052f7
49             ",
50             "owner": "2a5e46a1-2304-4cd4-869a-7566
51                 d8ad2d2f",
52             "name": "",
53             "value": "",
54             "unit": "",
55             "position": {
56                 "x": 0,
57                 "y": 0,
58                 "angle": 0
59             },
60             "isNameHidden": true,
61             "isValueHidden": true
62         },
63         "ports": [
64             {

```

```
60         "id": "f992587f-64e4-4b54-a0ee-639
        d4c099ff6",
61         "owner": "2a5e46a1-2304-4cd4-869a-7566
        d8ad2d2f",
62         "type": "hybrid",
63         "position": {
64             "x": 0.5,
65             "y": 0.0,
66             "angle": 0
67         },
68         "connection": "40495e8a-b84e-4c8a-8b18
        -410810c1a745"
69     }
70 ],
71     "position": {
72         "x": 559,
73         "y": 478,
74         "angle": 0
75     }
76 },
77 {
78     "id": "1419961f-7912-4bc0-80b6-53fe87627206",
79     "type": "",
80     "fullName": "",
81     "label": {
82         "id": "1e61a5a4-45d6-4ac7-9332-bf4d838ab907
            ",
83         "owner": "1419961f-7912-4bc0-80b6-53
            fe87627206",
84         "name": "",
85         "value": "",
86         "unit": "",
87         "position": {
88             "x": 667,
89             "y": 294,
90             "angle": 0
91         },
92         "isNameHidden": true,
93         "isValueHidden": true
94     },
95     "ports": [
96         {
97             "id": "5f06b471-b3a5-4f58-8c5b-7
                e67b520cb1e",
```

```

98         "owner": "1419961f-7912-4bc0-80b6-53
          fe87627206",
99         "type": "hybrid",
100        "position": {
101            "x": 0.6,
102            "y": 1.0,
103            "angle": 0
104        },
105        "connection": "40495e8a-b84e-4c8a-8b18
          -410810c1a745"
106    },
107    {
108        "id": "ba5c1010-2f17-4f0c-919d-
          cd4a45caa1b6",
109        "owner": "1419961f-7912-4bc0-80b6-53
          fe87627206",
110        "type": "hybrid",
111        "position": {
112            "x": 0.4,
113            "y": 0.0,
114            "angle": 0
115        },
116        "connection": "997b8201-0966-4063-a8c5-
          b85fabf2b7a6"
117    }
118 ],
119 "position": {
120     "x": 564,
121     "y": 290,
122     "angle": 0
123 }
124 },
125 {
126     "id": "a479d047-a102-494a-9516-2ed4ebe1257c",
127     "type": "",
128     "fullName": "",
129     "label": {
130         "id": "d81aa3c8-8e0a-4028-abe1-a34d2339552e
          ",
131         "owner": "a479d047-a102-494a-9516-2
          ed4ebe1257c",
132         "name": "",
133         "value": "",
134         "unit": "",
135         "position": {

```

```
136         "x": 42,
137         "y": 307,
138         "angle": 0
139     },
140     "isNameHidden": true,
141     "isValueHidden": true
142 },
143 "ports": [
144     {
145         "id": "ef2ee55c-af56-4824-ae27-513
146             dec71bf6d",
147         "owner": "a479d047-a102-494a-9516-2
148             ed4ebe1257c",
149         "type": "hybrid",
150         "position": {
151             "x": 0.6,
152             "y": 1.0,
153             "angle": 0
154         },
155         "connection": "74a67485-e333-4b2e-be1e-
156             aa4f50787bc5"
157     },
158     {
159         "id": "729374f2-36bd-4cde-8de0-86
160             ee163239b9",
161         "owner": "a479d047-a102-494a-9516-2
162             ed4ebe1257c",
163         "type": "hybrid",
164         "position": {
165             "x": 0.5,
166             "y": 0.0,
167             "angle": 0
168         },
169         "connection": "0c17d0fb-7b14-4e29
170             -9596-43d59f1a2cde"
171     }
172 ],
173 "position": {
174     "x": 155,
175     "y": 285,
176     "angle": 0
177 }
```

```
175     "type": "",
176     "fullName": "",
177     "label": {
178         "id": "2a8c7418-3534-4981-9cde-783db4f8fcd7",
179         "owner": "97a1a07d-9fdd-4839-8a0c-d987b2ff2347",
180         "name": "",
181         "value": "",
182         "unit": "",
183         "position": {
184             "x": 351,
185             "y": 31,
186             "angle": 0
187         },
188         "isNameHidden": true,
189         "isValueHidden": true
190     },
191     "ports": [
192         {
193             "id": "c1458b10-3693-4177-b3e1-a4a1b9768ea3",
194             "owner": "97a1a07d-9fdd-4839-8a0c-d987b2ff2347",
195             "type": "hybrid",
196             "position": {
197                 "x": 1.0,
198                 "y": 0.5,
199                 "angle": 0
200             },
201             "connection": "997b8201-0966-4063-a8c5-b85fabf2b7a6"
202         },
203         {
204             "id": "16cc89a4-d34a-4e8d-bf0e-e0e85490d7b6",
205             "owner": "97a1a07d-9fdd-4839-8a0c-d987b2ff2347",
206             "type": "hybrid",
207             "position": {
208                 "x": 0.0,
209                 "y": 0.4,
210                 "angle": 0
211             },
212         },
```

```
212         "connection": "0c17d0fb-7b14-4e29
213             -9596-43d59f1a2cde"
214     },
215     "position": {
216         "x": 316,
217         "y": 159,
218         "angle": 0
219     }
220 },
221 ],
222 "nodes": [],
223 "connections": [
224     {
225         "id": "40495e8a-b84e-4c8a-8b18-410810c1a745",
226         "start": "f992587f-64e4-4b54-a0ee-639d4c099ff6"
227         ,
228         "end": "5f06b471-b3a5-4f58-8c5b-7e67b520cb1e",
229         "label": {
230             "id": "0c3fa7d1-9745-44eb-9429-1390db028cc7",
231             "owner": "40495e8a-b84e-4c8a-8b18-410810c1a745",
232             "name": "",
233             "value": "",
234             "unit": "",
235             "position": {
236                 "x": 0,
237                 "y": 0,
238                 "angle": 0
239             },
240             "isNameHidden": true,
241             "isValueHidden": true
242         }
243     },
244     {
245         "id": "74a67485-e333-4b2e-be1e-aa4f50787bc5",
246         "start": "e63e7320-5565-4182-bfbb-c43478e5d1b7"
247         ,
248         "end": "ef2ee55c-af56-4824-ae27-513dec71bf6d",
249         "label": {
250             "id": "e7b1d402-ea77-442b-88d2-d54c8d685135",
251             "owner": "74a67485-e333-4b2e-be1e-aa4f50787bc5",
```

```

250         "name": "",
251         "value": "",
252         "unit": "",
253         "position": {
254             "x": 0,
255             "y": 0,
256             "angle": 0
257         },
258         "isNameHidden": true,
259         "isValueHidden": true
260     }
261 },
262 {
263     "id": "997b8201-0966-4063-a8c5-b85fabf2b7a6",
264     "start": "ba5c1010-2f17-4f0c-919d-cd4a45caa1b6"
265     ,
266     "end": "c1458b10-3693-4177-b3e1-a4a1b9768ea3",
267     "label": {
268         "id": "4fefb2f1-452b-419b-8882-082af736d3d5",
269         "owner": "997b8201-0966-4063-a8c5-b85fabf2b7a6",
270         "name": "",
271         "value": "",
272         "unit": "",
273         "position": {
274             "x": 0,
275             "y": 0,
276             "angle": 0
277         },
278         "isNameHidden": true,
279         "isValueHidden": true
280     }
281 },
282 {
283     "id": "0c17d0fb-7b14-4e29-9596-43d59f1a2cde",
284     "start": "729374f2-36bd-4cde-8de0-86ee163239b9"
285     ,
286     "end": "16cc89a4-d34a-4e8d-bf0e-e0e85490d7b6",
287     "label": {
288         "id": "a7039a59-45da-4aa8-b66c-ad80ec60335e",
289         "owner": "0c17d0fb-7b14-4e29-9596-43d59f1a2cde",
290         "name": "",

```

```
289         "value": "",
290         "unit": "",
291         "position": {
292             "x": 0,
293             "y": 0,
294             "angle": 0
295         },
296         "isNameHidden": true,
297         "isValueHidden": true
298     }
299 }
300 ]
301 }
```

Código B.1: Exemplo Estrutura de dados do Modelo