UNIVERSIDADE
BEIRA INTERIOR

Aplicação de Modelos de Machine Learning para Classificação de Mecanismos de Falha e Deteção de Fissuras em Ligas Metálicas Aeronáuticas

(Versão final após defesa)

Tomás Carvalho Figueiredo

Dissertação para obtenção do Grau de Mestre em
Engenharia Aeronáutica
(Mestrado Integrado)

Orientador: Prof. Doutor Pedro Vieira Gamboa
Coorientadores: Capitão Tomás Barros
Tenente Bruno Santos

março de 2025

Folha em branco

Declaração de Integridade

Eu, Tomás Carvalho Figueiredo, que abaixo assino, estudante com o número de inscrição 43825 de Engenharia Aeronáutica da Faculdade de engenharia, declaro ter desenvolvido o presente trabalho e elaborado o presente texto em total consonância com o **Código de Integridades da Universidade da Beira Interior**.

Mais concretamente afirmo não ter incorrido em qualquer das variedades de Fraude Académica, e que aqui declaro conhecer, que em particular atendi à exigida referenciação de frases, extratos, imagens e outras formas de trabalho intelectual, e assumindo assim na íntegra as responsabilidades da autoria.

Universidade da Beira Interior, Covilhã 01 /04 /2025

Assinado por: **Tomás Carvalho Figueiredo**
Num. de Identificação: 14398862
Data: 2025.04.01 14:23:17+01'00'

Folha em branco

Dedicatória

À minha família.

Folha em branco

Agradecimentos

Em primeiro lugar, gostaria de agradecer ao Major-General José Lourenço da Saúde que possibilitou o desenvolvimento deste projeto com a Força Aérea Portuguesa.

Gostaria também de agradecer ao Professor Doutor Pedro Gamboa, que aceitou prontamente o tema, demonstrando constante interesse e disponibilidade, além de oferecer conselhos valiosos que contribuíram para o rigor do trabalho.

Ao Capitão Tomás Barros e ao Tenente Bruno Santos, gostaria de expressar o meu sincero agradecimento pela constante disponibilidade e prontidão em auxiliar nos desafios encontrados, pelas várias reuniões de orientação e pelos valiosos conselhos e novas perspetivas que tanto contribuíram para alcançar os objetivos propostos.

Folha em branco

Resumo

O presente trabalho explora a aplicação de dois modelos baseadas em *machine learning*, nomeadamente redes neuronais convolucionais, para avaliação de falhas em ligas metálicas de componentes aeronáuticos da Força Aérea Portuguesa. O primeiro foi projetado para a classificação de mecanismos de falha utilizando imagens obtidas através do microscópio eletrónico de varrimento, enquanto o segundo se foca na deteção de fissuras de fadiga.

Para a tarefa de classificação foi utilizada uma arquitetura VGG16, ajustada com uma *classification head* e treinada utilizando técnicas de *transfer learning*. Para a deteção foi implementado um modelo baseado em *Faster* R-CNN com ResNet 101 e *Feature Pyramid Network*, pré-treinado e configurado para identificar e localizar fissuras de fadiga de vários tamanhos. Ambos os modelos foram treinados e validados em conjuntos de dados anotados manualmente e com recurso a técnicas de *data augmentation*.

O modelo de classificação atingiu uma exatidão de aproximadamente 93% e precisão de 92%, enquanto o modelo de deteção apresentou uma exatidão de cerca de 89% e precisão de 90%. Estes resultados destacam a capacidade das redes neuronais convolucionais em automatizar tarefas complexas, reduzindo o tempo de análise de grandes quantidades de imagens de uma só vez.

Palavras-chave

Fractografia; mecanismos de falha; fissuras de fadiga; machine learning; redes neuronais convolucionais; classificação; deteção; transfer learning.

Folha em branco

Abstract

The present work explores the application of two machine learning based models, namely convolutional neural networks, for the evaluation of failures in metallic alloys of aeronautical components of the Portuguese Air Force. The first model was designed for the classification of failure mechanisms using images obtained through a scanning electron microscope, while the second focuses on the detection of fatigue cracks.

For the classification task, a VGG16 architecture was used, adjusted with a customized classification head and trained using transfer learning techniques. For the detection task, a model based on Faster R-CNN with ResNet 101 and Feature Pyramid Network was implemented, pre-trained and configured to identify and locate fatigue cracks of various sizes. Both models were trained and validated on manually annotated datasets and data augmentation techniques were applied.

The classification model achieved an accuracy of approximately 93% and a precision of 92%, while the detection model presented an accuracy of around 89% and a precision of 90%. These results highlight the ability of convolutional neural networks to automate complex tasks, reducing the time required to analyze large amounts of imagens at once.

Keywords

Fractograpy; failure mechanisms; fatigue cracks; machine learning; convolutional neural networks; classification; detection; transfer learning.

Folha em branco

Índice

Folha em branco

Lista de Figuras

Folha em branco

Lista de Tabelas

Folha em branco

Lista de Acrónimos

ANN	*Artificial Neural Network*
BCE	*Binary Cross-Entropy*
BGD	*Batch Gradient Descent*
CCE	*Categorical Cross-Entropy*
CNN	*Convolutional Neural Network*
DAWN	*Deep Adaptive Wavelet Network*
FAP	Força Aérea Portuguesa
FCIS	*Fatigue Crack Initiation Sites*
FCL	*Fully Connected Layer*
FN	*False Negative*
FP	*False Positive*
FPN	*Feature Pyramid Network*
IA	Inteligência Artificial
IoU	*Intersection over Union*
LR	*Learning Rate*
MEV	Microscópio Eletrónico de Varrimento
ML	*Machine Learning*
R-CNN	*Region-Based Convolutional Neural Network*
ReLU	*Rectified Linear Unit*
ResNet	*Residual Network*
ROI	*Region of Interest*
RPN	*Region Proposal Network*
SA	Sistemas de Armas
SCC	*Stress Corrosion Cracking*
SGD	*Stochastic Gradient Descent*
TN	*True Negative*
TP	*True Positive*
YOLO	*You Only Look Once*

Folha em branco

Capítulo 1

1 Introdução

1.1 Motivação

Durante a operação de aeronaves, certos componentes estão sujeitos a condições de carga e ambientais extremas. Estas condições podem levar à falha desses componentes, comprometendo a segurança estrutural da aeronave. Quando ocorrem tais situações é essencial realizar uma análise detalhada das superfícies de fratura resultantes para compreender os mecanismos de falha envolvidos (Duarte et al., 2016).

A fractografia faz parte do processo que se tem de colocar em prática na análise de falhas de componentes em serviço. É uma técnica que permite correlacionar as características topográficas das superfícies de fratura com o mecanismo de falha que levou à falha do material (Pantazopoulos, 2011). Tais caraterísticas são, na maioria das vezes, reveladas apenas quando as superfícies são observadas com ampliações de aproximadamente 1000x ou superior (Janssen et al., 2002). Assim sendo, recorre-se ao microscópio eletrónico de varrimento (MEV) para obtenção de imagens que revelam essas características graças à sua capacidade de alta resolução a elevadas ampliações (Russo, 1978).

A relevância de realizar estas análises pode ser claramente evidenciada quando aplicada aos Sistemas de Armas (SA) da Força Aérea Portuguesa (FAP). Por exemplo, análises efetuadas com recurso ao MEV foram cruciais para identificar o modo de falha em incidentes como a aterragem de emergência da aeronave Epsilon TB-30 em Monte Real, o desacoplamento de painéis do rotor de cauda da aeronave EH-101, a falha no trem de nariz de uma aeronave C-130 e o rebentamento de canos do canhão de aeronaves F-16. Estas observações permitiram identificar qual o modo de falha associado, possibilitando a implementação de ações corretivas de modo a evitar situações semelhantes no futuro.

Apesar de tudo, este processo é muito demorado e suscetível a erros humanos, exigindo um alto nível de experiência e conhecimento especializado. Com o aumento exponencial da quantidade de dados disponíveis, torna-se essencial o desenvolvimento de métodos automatizados que possam processar e analisar essas imagens de forma eficiente e precisa.

Neste contexto, o *Machine Learning* (ML) surge como uma abordagem promissora para automatizar os processos de classificação de imagens de superfícies de fratura nas várias classes e de deteção de certas características de cada mecanismo de falha, tal como as fissuras de fadiga. As técnicas de ML têm demonstrado um elevado desempenho em diversas aplicações de análise de imagens, permitindo a identificação precisa de padrões complexos que seriam difíceis ou até mesmo impossíveis de detetar ao olho humano (Endo et al., 2022; S. Y. Wang & Guo, 2021).

1.2 Objetivos

O principal objetivo desta dissertação consiste na aplicação de ferramentas de ML para avaliação de falhas em ligas metálicas aeronáuticas, nomeadamente, a classificação dos três principais mecanismos de falha (sobrecarga, fadiga e corrosão) com base em imagens obtidas através do MEV e a deteção de fissuras de fadiga. De modo a alcançar estes objetivos, foi necessário realizar as seguintes tarefas intermédias:

- Familiarização com os conceitos de falha de ligas metálicas e o histórico de situações relevantes nos SA da FAP.

- Investigação dos métodos de ML mais apropriados para a classificação de imagens e deteção de objetos.

- Obtenção e preparação de um *dataset* de imagens provenientes do MEV de várias ligas metálicas e diferentes modos de falha.

- Implementação das referidas ferramentas com base em algoritmos já existentes, adaptando-os às necessidades específicas do trabalho.

1.3 Estrutura da Dissertação

A dissertação está organizada em cinco capítulos, que abordam tópicos desde a fundamentação teórica até à aplicação prática das ferramentas. Esta secção tem como objetivo apresentar um resumo dos tópicos principais abordados em cada capítulo.

- **Capítulo 1**: Introdução, no qual são apresentados a motivação, os objetivos e a estrutura da dissertação.

- **Capítulo 2**: Revisão da Literatura, que oferece uma visão geral sobre os conceitos de falhas em ligas metálicas e as técnicas de ML para classificação e deteção.

- **Capítulo 3**: Metodologia, no qual são descritos os processos de aquisição de imagens, pré-processamento das mesmas e arquitetura e configurações dos modelos, bem como as técnicas específicas utilizadas para a classificação dos mecanismos de falha e deteção de fissuras.

- **Capítulo 4**: Resultados, no qual são apresentados os resultados obtidos pelos modelos, seguidos de uma análise crítica aos mesmos.

- **Capítulo 5**: Conclusões, no qual são apresentadas as principais conclusões do trabalho, as dificuldades encontradas e as sugestões para trabalhos futuros.

Capítulo 2

2 Revisão da Literatura

Nesta secção é apresentada a revisão da literatura realizada com o objetivo de compreender os principais conceitos e fundamentos teóricos que servem de base às tarefas propostas. Inicialmente são abordados os diferentes mecanismos de falha envolvidos na tarefa de classificação e explorados os princípios da fractografia. De seguida são introduzidas as várias técnicas de *machine learning*, com destaque para as redes neuronais convolucionais, detalhando a sua estrutura e os seus principais componentes, como camadas convolucionais, funções de ativação, funções de perda e otimizadores, e também aspetos como *overfitting*, *underfitting* e *transfer learning*. Por fim, é analisada a evolução das arquiteturas para tarefas de classificação e deteção, bem como o estado da arte, com o objetivo de justificar e contextualizar as escolhas realizadas ao longo da dissertação.

2.1 Mecanismos de Falha

2.1.1 Sobrecarga

A falha por sobrecarga em ligas metálicas é um fenómeno caracterizado pela incapacidade do material em suportar o aumento da tensão estática aplicada, excedendo a sua resistência mecânica, o que resulta na sua rutura. Dependendo das características microestruturais do material e das condições de carregamento, podem surgir diferentes modos de propagação da fissuras, como a fratura dúctil por coalescência de microvazios ou a fratura intergranular (Janssen et al., 2002).

A fratura dúctil é o principal mecanismo de fratura associado à falha por sobrecarga em ligas metálicas aeronáuticas e é caracterizada pela deformação plástica significativa e grande absorção de energia antes de ocorrer a falha (Callister Jr. & Rethwisch, 2013). O modo como se origina a fratura dúctil envolve a nucleação, crescimento e coalescência de microvazios, como está representado na Figura 2.1 (Meyers & Chawla, 2012). Os pontos onde geralmente se iniciam os microvazios estão associados a zonas de descontinuidade de deformação localizada, que podem ser partículas de segunda fase, inclusões ou fronteiras de grão (ASM Handbook Committee, 1987), que são de caráter

frágil e, consequentemente, não são capazes de suportar grandes deformações plásticas (Janssen et al., 2002; Meyers & Chawla, 2012).

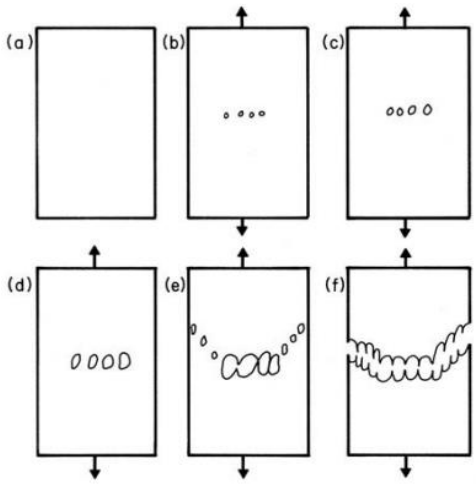

Figura 2.1: Sequência esquemática da formação de uma rutura dúctil através da nucleação, crescimento e coalescência de microvazios (Meyers & Chawla, 2012).

Quando o material deixa de ser capaz de suportar o aumento da deformação e das tensões locais ocorre a rutura. A superfície de fratura dúctil resultante é caracterizada pela presença de *dimples*, que são cavidades formadas pela união dos microvazios durante o processo de fratura e evidenciam a absorção de energia que ocorre antes da falha. Desta forma, os *dimples* são características apenas observadas quando a falha ocorre por sobrecarga (ASM Handbook Committee, 1987; Janssen et al., 2002; Meyers & Chawla, 2012). A Figura 2.2 demonstra como são observadas essas cavidades no MEV.

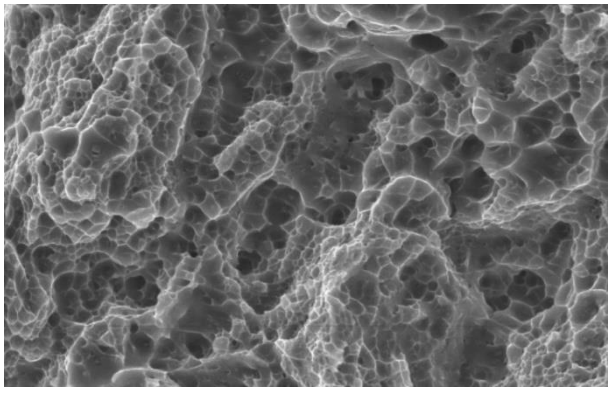

Figura 2.2: Superfície de fratura com *dimples* observada no MEV com ampliação de 2000x.

O tamanho apresentado pelos *dimples* está diretamente relacionado com o número de microvazios que se formam, bem como a forma como estão distribuídos (ASM Handbook Committee, 1987). Quando os pontos de nucleação são poucos e distantes entre si, os

microvazios aumentam consideravelmente de tamanho antes de se unirem, o que resulta numa superfície de fratura composta por *dimples* grandes. Por outro lado, *dimples* de pequena dimensão surgem quando existem muitos pontos de nucleação próximos entre si, de modo que os microvazios coalescem antes de terem a possibilidade de crescer até um tamanho significativo (ASM Handbook Committee, 1987). O seu tamanho é uma característica que pode ser utilizada para avaliar a ductilidade do material (Möser, 1987).

A forma dos *dimples* fornece informações sobre o modo de carregamento a que o material foi submetido (ASM Handbook Committee, 1987; Broek, 1984; Janssen et al., 2002). Existem três modos básicos de carregamento a que um material fissurado pode estar sujeito, tal como é representado na Figura 2.3 (Dowling, 2012).

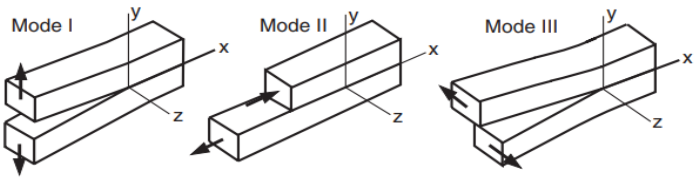

Figura 2.3: Modos de carregamento básicos (Dowling, 2012).

Uma superfície de fratura resultante de um modo de carregamento de tração uniaxial dá origem a *dimples* equiaxiais e com um contorno bem definido. Por outro lado, se a superfície de fratura estiver associada aos modos I, II ou III, os *dimples* formados apresentam uma forma alongada e com uma extremidade aberta (ASM Handbook Committee, 1987; Broek, 1984; Janssen et al., 2002). Em fraturas pelo modo I, os *dimples* alongados estão orientados segundo a mesma direção em ambas as superfícies resultantes da fratura. Em contrapartida, nas fraturas pelo modo II ou III a orientação é oposta em ambas as superfícies de fratura (ASM Handbook Committee, 1987; Janssen et al., 2002). Esta característica pode ser útil para determinar a direção de propagação da fissuras, visto que a extremidade fechada do *dimple* aponta sempre para a origem da mesma (ASM Handbook Committee, 1987). A Figura 2.4 demonstra como o modo de carregamento afeta a forma dos *dimples*.

A fratura intergranular associada à falha por sobrecarga, contrariamente à fratura dúctil caracterizada pelos *dimples*, apresenta uma superfície de rutura com grãos bem definidos, visto que a propagação da fissuras ocorre ao longo das fronteiras de grão sem deformação plástica significativa. De um modo geral podem existir duas formas de originar a fratura intergranular. Na primeira ocorre a rutura do material ao longo das fronteiras de grão através da coalescência de microvazios, semelhante ao que se observa

na fratura dúctil. Neste caso, as partículas de segunda fase e inclusões surgem nas fronteiras de grão, enfraquecendo-as e delineando o trajeto pelo qual se irá propagar a fissura (Janssen et al., 2002). A Figura 2.5 mostra como é observada este tipo de fratura no MEV, com uma superfície relativamente rugosa ao longo das fronteiras de grão.

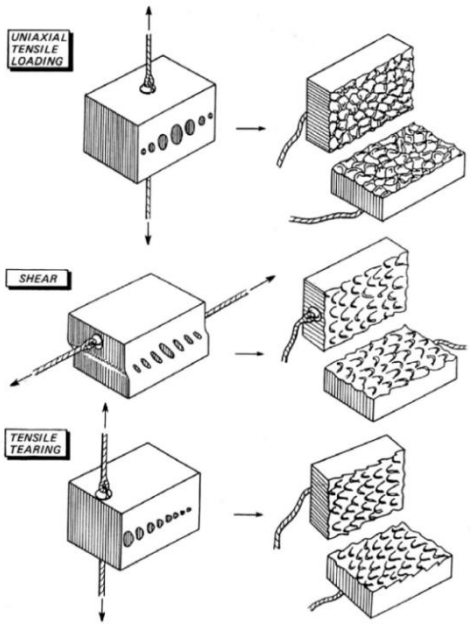

Figura 2.4: Formação de dimples de acordo com o tipo de carregamento aplicado (Janssen et al., 2002).

A segunda forma segundo a qual se pode originar a fratura intergranular é sem a coalescência de microvazios, através de uma rutura rápida sem grande absorção de energia e reduzida deformação plástica. Observa-se principalmente em aços fragilizados por tratamento térmico, tornando-os suscetíveis à falha catastrófica mesmo sob cargas moderadas. A superfície deste tipo de fratura é mostrada na Figura 2.6, com facetas lisas.

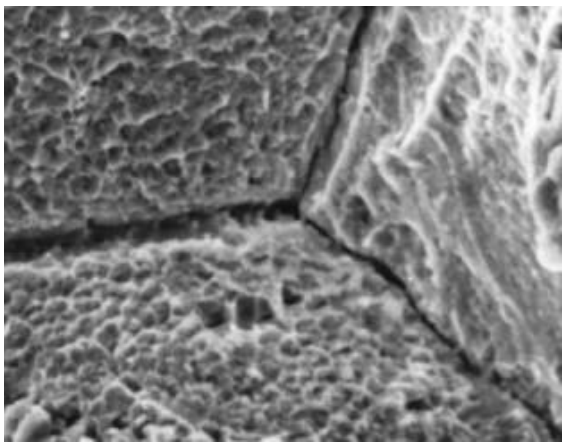

Figura 2.5: Superfície de fratura intergranular por coalescência de microvazios observada no MEV com ampliação 1400x (Janssen et al., 2002).

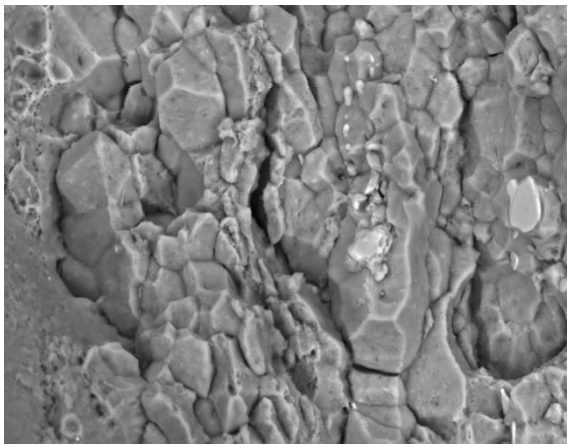

Figura 2.6: Superfície de fratura intergranular sem coalescência de microvazios observada no MEV com ampliação 2000x.

2.1.2 Fadiga

A falha por fadiga ocorre sob carregamentos cíclicos e corresponde à principal causa de falhas em ligas metálicas de componentes aeronáuticos. O processo de falha por fadiga é caracterizado por três estágios: nucleação da microfissura, propagação da fissura e falha final do material (ASM Handbook Committee, 1987; Callister Jr. & Rethwisch, 2013).

A iniciação de fissuras é normalmente desencadeada pela concentração de tensões em descontinuidades do material como intrusões ou entalhes, associadas muitas vezes a defeitos resultantes do seu processo de fabrico. Apesar de na maioria dos casos as tensões nominais a que o material está sujeito estarem abaixo do limite elástico, localmente podem ultrapassar esse valor e, como resultado, ocorre deformação plástica local a uma escala microscópica (Broek, 1984). Esta fase é significativamente afetada pela microestrutura do material e pode representar até 90% da sua vida útil (ASM Handbook Committee, 1987).

Existem vários modelos que tentam explicar como se iniciam as fissuras devido à deformação plástica local. O modelo de Wood, representado na Figura 2.7, descreve que durante um ciclo de carregamento, o deslizamento do material ocorre alternadamente em planos paralelos, podendo formar saliências ou intrusões na superfície do mesmo. A origem destas intrusões está associada ao movimento de deslocações, que são defeitos lineares na estrutura cristalina do material, originando pontos de concentração de tensões. Com a deformação plástica contínua, uma intrusão pode, eventualmente, tornar-se um ponto de nucleação de fissuras (Broek, 1984).

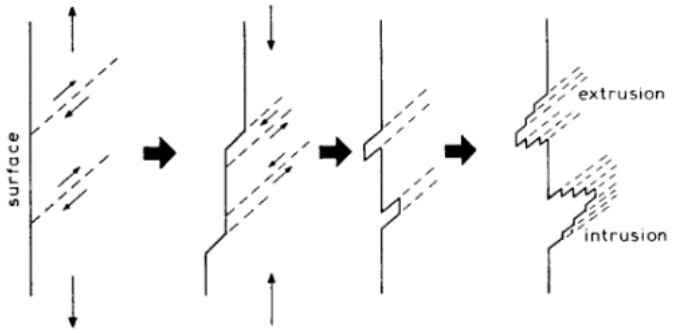

Figura 2.7: Formação de inclusões e extrusões de acordo com o modelo de Wood (Broek, 1984).

Uma vez iniciada, a fissuras causa uma grande concentração de tensões na sua frente, onde o deslizamento pode ocorrer com bastante facilidade. Na Figura 2.8 são mostradas as várias etapas do crescimento de uma fissuras por fadiga. Inicialmente o material

adjacente à frente da fissura pode deslizar ao longo de planos cristalográficos favoráveis em relação à tensão de corte máxima, como se observa nos estágios 1 e 2. Com isto, a fissura não só aumenta a sua área de abertura como também se propaga em comprimento (estágio 3), permitindo que o deslizamento possa ocorrer noutros planos, isto é, se possa propagar em várias direções possíveis. À medida que a fissura avança ocorre deformação plástica localizada na região ao redor da sua frente a cada ciclo de carregamento. Essa deformação provoca o encruamento local do material que, juntamente com o aumento da tensão, levam ao arredondamento da frente da fissura, favorecendo o seu avanço (Δa) durante a fase de aumento de carga (estágio 4). A deformação plástica que ocorre em volta da fissura não se ajusta completamente ao material com comportamento elástico circundante e isso leva ao desenvolvimento de tensões de compressão na frente da fissura durante a parte do ciclo em que há alívio de carga. Estas tensões de compressão são suficientes para causar uma deformação plástica inversa na frente da fissura, ou seja, levar ao fecho da mesma (estágio 5), visto que a nível local ultrapassam a tensão de cedência. Ao longo dos vários ciclos, a sucessiva abertura e fecho da fissura cria um padrão característico de ondulações na superfície de fratura, conhecido como estrias. Cada estria representa um único ciclo de carregamento (Broek, 1984).

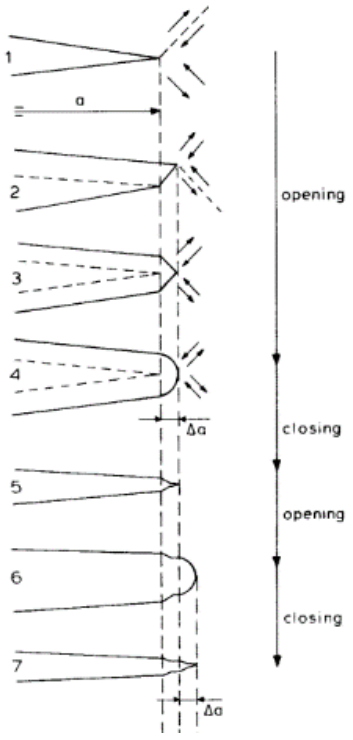

Figura 2.8: Modelo explicativo para a propagação da fissura por fadiga (Broek, 1984).

As estrias são características exclusivas da falha por fadiga e que podem ser observadas no MEV, permitindo assim a sua distinção entre outros mecanismos de falha. As estrias indicam também as sucessivas posições da frente da fissura ao longo da sua propagação e são perpendiculares à direção de propagação da mesma, o que é útil para determinar o local exato da sua nucleação (ASM Handbook Committee, 1987; Broek, 1984). Como cada estria é resultado de um único ciclo de carga, a sua análise permite estimar a velocidade de propagação da fissura, sendo um parâmetro relevante para a definição de intervalos de inspeção e previsão da vida útil do componente (Russo, 1978).

Através da observação de uma superfície de fratura por fadiga no MEV, especificamente uma região do estágio de propagação da fissura, é possível obter informações adicionais a partir do espaçamento e da definição das estrias, como se pode ver na Figura 2.9. Quando as cargas cíclicas têm amplitude constante, o espaçamento entre as estrias é uniforme (A). Por outro lado, uma condição de carga cíclica aleatória produz estrias com espaçamentos variados (B). No caso de o espaçamento entre estrias ser muito reduzido, pode indicar uma condição de alto número de ciclos e baixa carga (C). Por outro lado, um espaçamento grande indica uma condição de baixo número de ciclos e alta carga (D). A clareza que as estrias apresentam também fornece informações importantes, isto é, estrias bem definidas são produzidas pela aplicação de tensões de tração pura (E), e estrias com aparência desgastada e com marcas de pequenas fissuras sofreram cargas de compressão (F) (Russo, 1978).

No último estágio, a fissura atinge um tamanho crítico, a partir do qual se dá um crescimento repentino e progressivamente deixam de se formar estrias até que o material não consiga mais suportar a carga e ocorra a sua falha final. A área de fratura final oferece uma indicação da magnitude das cargas aplicadas, ou seja, uma grande área de fratura final indica que a tenacidade à fratura foi excedida para um comprimento de fissura relativamente curto, o que significa que a carga máxima foi alta, a tenacidade à fratura é baixa, ou ambos. O conhecimento da tenacidade à fratura de um determinado material permite obter uma estimativa razoável da tensão máxima na falha (Janssen et al., 2002).

Desta forma, a falha por fadiga pode ser identificada através da análise da superfície de fratura resultante, caracterizada pela presença de estrias, que apenas se formam quando o componente está sujeito a cargas cíclicas.

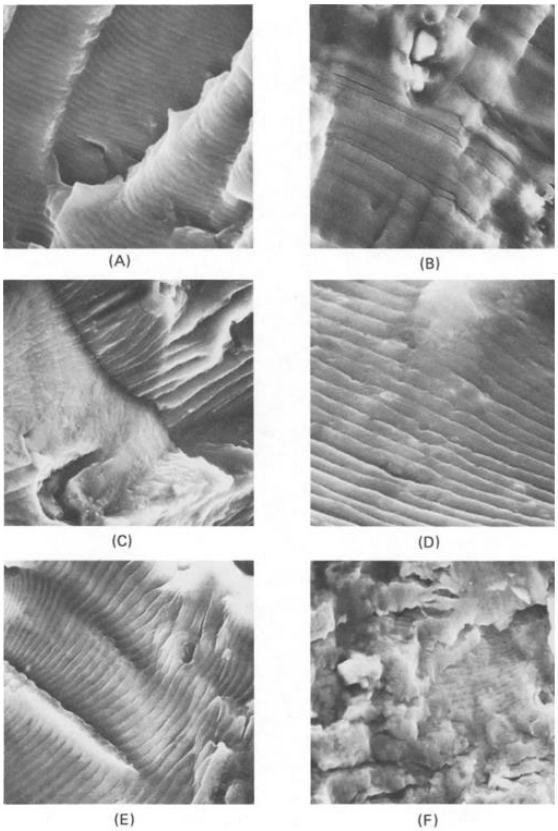

Figura 2.9: Estrias de fadiga resultantes de várias condições de carregamento observadas no MEV (Russo, 1978).

2.1.3 Corrosão

A falha por corrosão é normalmente designada por *stress corrosion cracking* (SCC) e resulta da combinação de três fatores principais: material sensibilizado, tensões de tração aplicadas (normalmente abaixo do limite elástico) e um ambiente corrosivo. Esta interação resulta na iniciação e propagação de fissuras de forma lenta, mas potencialmente catastrófica (Janssen et al., 2002; Khalifeh, 2020).

Relativamente à superfície de fratura resultante, que permite compreender as condições que levaram à falha do material, esta apresenta, na maioria dos casos, aspeto frágil com aparência intergranular ou transgranular, dependendo das condições específicas de carregamento e da microestrutura (ASM Handbook Committee, 1987). A aparência intergranular ocorre quando agentes corrosivos, como cloretos ou sulfetos, por exemplo, promovem a segregação de impurezas nas fronteiras de grão, enfraquecendo-as. Posteriormente, sob tensão de tração, os contornos de grão tornam-se caminhos preferenciais para a propagação de fissuras (Janssen et al., 2002; Khalifeh, 2020). Por outro lado, a aparência transgranular está frequentemente associada à rutura de filmes protetores na superfície do material, provocada por tensões mecânicas ou reações

químicas com o ambiente. Esta exposição permite que ocorra dissolução anódica localizada, facilitando em alguns casos a entrada de hidrogénio na microestrutura do material. O hidrogénio pode também ser absorvido durante processos associados à sua fabricação como soldadura, decapagem ou como foi referido, pela interação com ambientes corrosivos. Uma vez difundido no material, desloca-se para zonas de maior concentração de tensão, como as pontas de fissuras. Aí, o hidrogénio interage com imperfeições que possam existir, como microvazios, enfraquecendo a estrutura cristalina, o que contribui para o avanço da fissura que, sob tensões de tração elevadas, facilita a propagação das mesmas ao longo de planos dentro dos próprios grãos (Janssen et al., 2002; Möser, 1987).

A SCC é também descrita como uma forma de *delayed failure*, conceito que se refere ao comportamento progressivo deste tipo de falha, em que as microfissuras formadas (Figura 2.10) evoluem lentamente ao longo do tempo, até que, num dado momento, culminam numa rutura abrupta do material (ASM Handbook Committee, 1987; Russo, 1978). Quando observadas ao MEV, as superfícies associadas a este mecanismo de falha, que contêm indícios de produtos corrosivos, podem apresentar maior brilho (Figura 2.11). Este fenómeno ocorre porque os agentes corrosivos possuem um número atómico superior ao do material, resultando em regiões mais claras na imagem, quando observada ao MEV. Desta forma, a presença destes depósitos de agentes corrosivos em superfícies com ramificações de fissuras possibilita a distinção deste mecanismo de falha, evidenciando a ação conjunta do meio corrosivo e as tensões aplicadas.

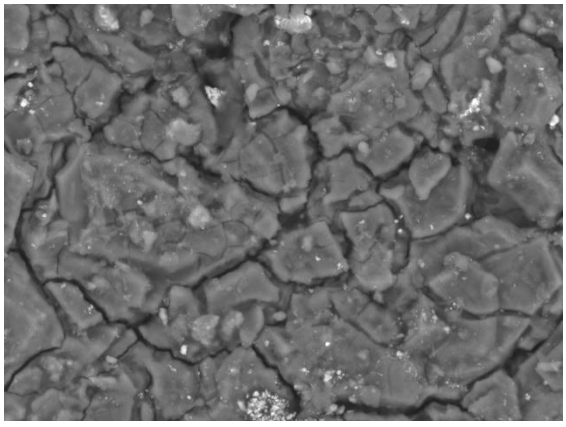

Figura 2.11: Imagem de superfície corroída evidenciando ramificações de fissuras, observadas no MEV com ampliação 2000x.

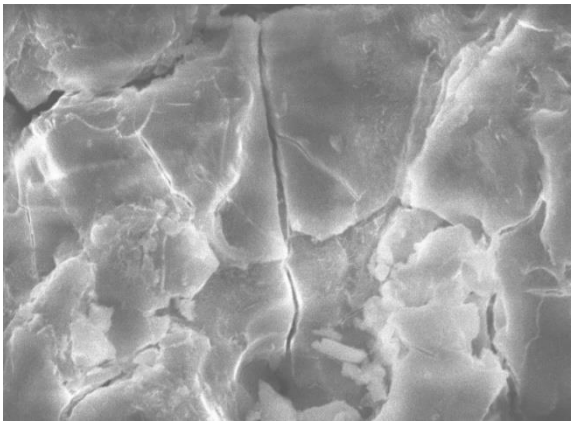

Figura 2.10: Imagem de superfície corroída com indícios de agentes corrosivos, zonas mais claras, observadas no MEV com ampliação 2000x.

2.2 Fractografia

Introduzida por Carl A. Zapffe em 1944, a fractografia consiste no estudo das características topográficas das superfícies de fratura, com o objetivo de correlacioná-las com os respetivos mecanismos de falha (ASM Handbook Committee, 1987). Zapffe destacou-se na aplicação do microscópio ótico à fractografia, conseguindo realizar análises com ampliações de 1500x a 2000x, tendo sido o pioneiro na observação de estrias em superfícies de fratura por fadiga, descritas num estudo de 1951 (ASM Handbook Committee, 1987).

O microscópio eletrónico de varrimento, desenvolvido a partir da década de 1930, com avanços significativos na década de 1960, tornou-se uma ferramenta indispensável na análise de superfícies de fratura. Em relação ao microscópio ótico, o MEV apresenta duas principais melhorias: aumenta os limites de resolução e a profundidade de campo. Enquanto o microscópio ótico é limitado a ampliações de até 2000x e uma resolução de 100 a 200 nm, o MEV pode atingir ampliações de 10000x a 60000x, com uma resolução muito superior, na ordem dos 4 a 5 nm (ASM Handbook Committee, 1987). De entre as principais contribuições do MEV para a fractografia, destacam-se aquelas relativas aos seguintes mecanismos de fratura:

- **Fraturas dúcteis por *overload***: Confirmação dos micromecanismos de fratura dúctil, nomeadamente a iniciação, crescimento e coalescência de microvazios. Foram também desenvolvidas correlações entre o tamanho e formato dos *dimples* com o estado de tensão e pureza do material (ASM Handbook Committee, 1987).

- **Fraturas por fadiga**: Foi possível desenvolver novos modelos que explicam os mecanismos de fratura por fadiga, além de estabelecer correlações entre as estrias de fadiga, o número de ciclos e as condições de carregamento (ASM Handbook Committee, 1987).

Atualmente a fractografia com recurso ao MEV é uma das técnicas mais comuns e cruciais para o estudo detalhado das superfícies de falha. No entanto, esta abordagem requer um conhecimento técnico profundo e experiência prévia, já que as mesmas características podem aparecer de formas distintas dependendo do material e das condições de observação. Sendo um processo realizado manualmente por especialistas altamente qualificados, o tempo despendido é bastante grande e existe forte dependência

do fator humano, o que torna o processo suscetível a erros (Endo et al., 2022; Tsopanidis et al., 2020).

Assim, a automação deste processo com o uso de algoritmos de ML é vista como uma solução promissora para aumentar a confiabilidade e eficiência da análise fractográfica. Desta forma, não só se reduz a carga de trabalho manual, como também permite que se obtenham novas informações sobre os vários mecanismos de falha (Tsopanidis et al., 2020).

2.3 Técnicas de Machine Learning

A Inteligência Artificial (IA) é a ciência que procura desenvolver máquinas e programas capazes de adquirir e aplicar conhecimento para resolver problemas complexos. Dentro da IA surgiram subcampos como o *Machine Learning* e *Deep Learning*, que são fundamentais para alcançar esses objetivos. A IA foca-se em criar sistemas inteligentes que imitam a capacidade de tomada de decisão humana, enquanto o principal foco do ML é treinar esses sistemas para melhorar a sua precisão com base nos dados de entrada, sem a necessidade de programação explícita para cada tarefa (Suguna et al., 2021).

O *Deep Learning*, uma subárea do ML, utiliza redes neuronais com várias camadas para processar grandes volumes de dados e realizar tarefas como deteção de objetos, segmentação de imagens e extração de características. Desta forma tornou-se possível a aplicação de IA em áreas que envolvem dados visuais, como a visão computacional, permitindo que as máquinas analisem e interpretem imagens digitais, replicando a maneira como o cérebro humano processa essas informações. O seu objetivo é extrair informações úteis dos pixéis das imagens (Suguna et al., 2021).

2.3.1 Tipos de Modelos

Uma forma de classificar diferentes modelos de ML é com base nos diferentes tipos de supervisão humana que necessitam durante o treino. De um modo geral existem duas categorias: aprendizagem supervisionada e não supervisionada.

A aprendizagem supervisionada é o caso mais comum em ML, onde o objetivo é ensinar o modelo a mapear os dados de entrada para saídas conhecidas, usando um conjunto de exemplos que já foram previamente anotados (Chollet, 2017). Alguns exemplos dos

algoritmos mais importantes que utilizam aprendizagem supervisionada são: *Logistic Regression, Support Vector Machines* e *Artificial Neural Networks* (Géron, 2019).

A aprendizagem não supervisionada, por outro lado, procura descobrir padrões ou estruturas nos dados de entrada sem usar rótulos pré-definidos. Duas categorias de aprendizagem não supervisionada são *clustering* e *dimensionality reduction* (Chollet, 2017). Alguns exemplos de algoritmos que utilizam aprendizagem não supervisionada são: *K-Means, Hierarchical Cluster Analysis* e *Principal Component Analysis* (Géron, 2019).

2.3.2 Redes Neuronais Artificiais

Uma rede neuronal artificial (*artificial neural network*, ANN) é um tipo de algoritmo de ML supervisionado inspirado no funcionamento do cérebro humano, projetado para reconhecer padrões e estabelecer relações entre diferentes conjuntos de dados. Esta analogia deve-se ao facto de uma ANN ser constituída por neurónios artificiais, também conhecidos como nós, que são comparados aos neurónios existentes no cérebro humano e estão organizados em várias camadas. Numa ANN há três tipos de camadas: a camada de entrada (*input layer*), que recebe os dados iniciais; as camadas ocultas (*hidden layers*), responsáveis por processar e transformar esses dados, e a camada de saída (*output layer*), que gera a resposta final (Narciso, 2022; Santos, 2020). A Figura 2.12 apresenta a estrutura básica de uma ANN.

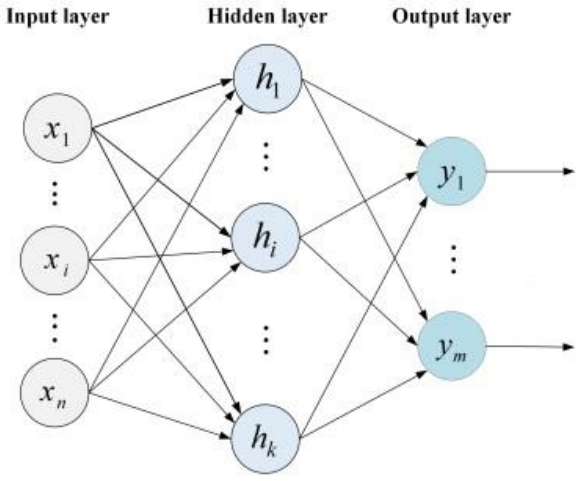

Figura 2.12: Representação da estrutura básica de uma ANN. Adaptado de (Chen et al., 2021).

Cada neurónio recebe dados de entrada que são ponderados por valores chamados pesos (*weights*). Esses pesos determinam a influência de cada entrada na saída do neurónio e, para além disso, um valor adicional chamado *bias* é somado ao produto da entrada e do peso, permitindo que o neurónio seja ativado ou não, mesmo quando as entradas são nulas. Esse mecanismo é essencial para evitar que a rede se torne inativa ou incapaz de gerar saídas úteis (Narciso, 2022; Santos, 2020). A transformação matemática que descreve a relação entre o sinal de entrada e o valor de saída é dada pela equação 2.1:

$$y = f(b + \sum_{i=1}^{n} (x_i w_i)) \tag{2.1}$$

onde f é a função de ativação, x_i corresponde aos dados de entrada, w_i é o peso do respetivo dado de entrada, b a *bias* e y o valor de saída. Esta operação é ilustrada na Figura 2.13, que apresenta a estrutura de um neurónio artificial e como os dados de entrada são processados para gerar uma saída:

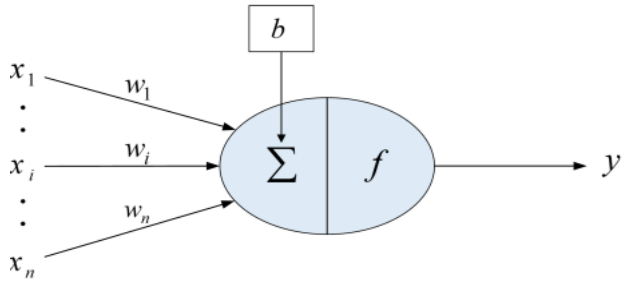

Figura 2.13: Representação de um neurónio artificial (Chen et al., 2021).

O processo de aprendizagem de uma ANN, ilustrado na Figura 2.14, envolve o ajuste contínuo dos parâmetros da rede (pesos e *biases*) durante o treino, realizado ao longo de múltiplas iterações, agrupadas em épocas (*epochs*). Cada época corresponde a uma passagem completa pelo conjunto de treino, enquanto uma iteração se refere ao processamento de um lote de dados (*batch*). Durante o treino, os parâmetros são ajustados automaticamente para minimizar o erro entre as previsões da rede e os resultados esperados (*groundtruth*), através de algoritmos de otimização, que calculam o gradiente da função de perda e atualizam os parâmetros no sentido de diminuir o seu valor (Santos, 2020). Para além disso, antes de iniciar o treino são definidos os hiperparâmetros, que são configurados manualmente e têm influência no desempenho do processo de otimização dos parâmetros da rede. A taxa de aprendizagem e o número de épocas são exemplos de hiperparâmetros (Chollet, 2017).

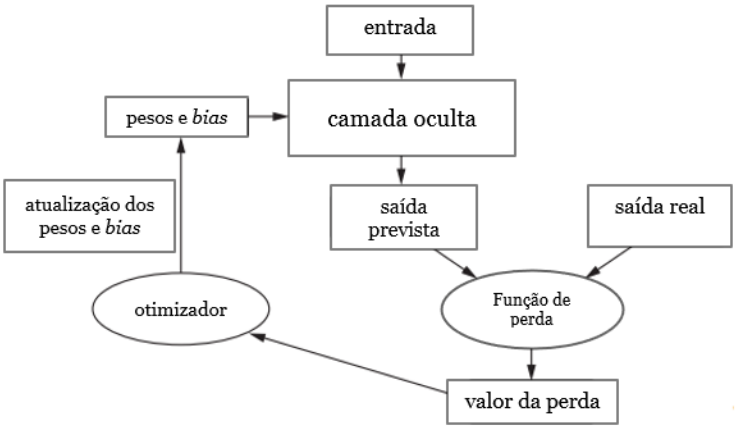

Figura 2.14: Representação simplificada do processo de treino de uma ANN. Adaptado de (Chollet, 2017).

2.3.3 Redes Neuronais Convolucionais

Uma rede neuronal convolucional (*convolutional neural network*, CNN) é um tipo de ANN profunda que é principalmente utilizada quando os dados de entrada são imagens (Gonçalves, 2020). As CNNs são uma escolha ideal para tarefas de visão computacional, como reconhecimento e deteção de objetos e classificação de imagens.

A estrutura básica de uma CNN, apresentada na Figura 2.15, é composta por várias camadas, incluindo camadas convolucionais, camadas de *pooling*, funções de ativação não-lineares e camadas totalmente conectadas. Primeiramente, a imagem de entrada é pré-processada seguindo para a rede pela camada de entrada. De seguida a imagem passa por várias camadas convolucionais alternadas com camadas de *pooling*, onde são extraídas características específicas (*features*) das imagens através das camadas convolucionais e reduzida a dimensionalidade dos mapas de características gerados nas camadas de *pooling*, mantendo as informações mais relevantes e minimizando a carga computacional. No final da rede, as camadas totalmente conectadas atuam como um classificador que usa as características extraídas para determinar a probabilidade de a imagem pertencer a uma determinada classe. Já em tarefas de deteção, as características extraídas são utilizadas não só para classificar a classe de cada objeto como para indicar a sua posição na imagem (Chen et al., 2021).

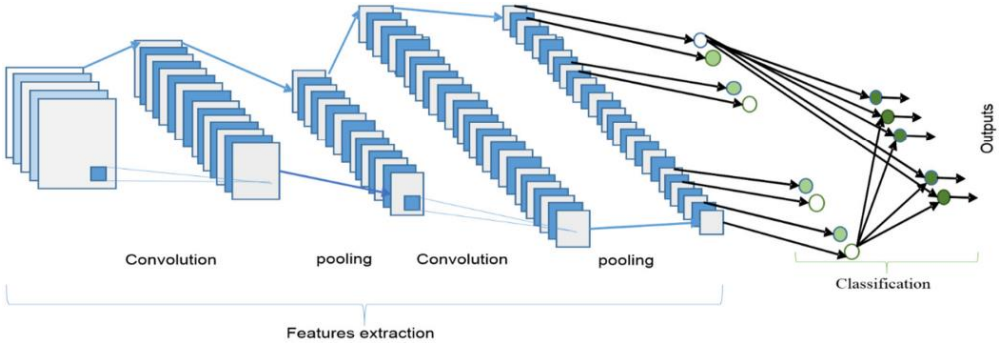

Figura 2.15: Arquitetura de uma CNN, incluindo a entrada, camadas de convolução e *pooling* intercaladas, uma camada totalmente conectada e as saídas. Adaptado de (Alom et al., 2019).

2.3.3.1 Camada Convolucional

A camada convolucional atua sobre os dados de entrada, aplicando filtros (*kernels*), com o objetivo de extrair características representativas das imagens. Normalmente, em CNNs com alguma profundidade, as camadas convolucionais mais superficiais extraem características básicas como textura, linhas e cantos enquanto as camadas mais profundas são responsáveis por extrair características mais abstratas e complexas como um rosto, olhos ou um carro, por exemplo (Chen et al., 2021).

Cada camada convolucional possui vários filtros, que são geralmente matrizes quadradas de pequena dimensão, como 3x3, 5x5 e 7x7, constituídos por pesos treináveis e que realizam operações de convolução. Estes filtros são sucessivamente aplicados a todos os pixéis da imagem de entrada, realizando multiplicações ponto a ponto entre os valores do *kernel* e os valores correspondentes na imagem que está a ser processada. O resultado dessas multiplicações é somado, produzindo um único valor de saída. Este processo é repetido para cada posição possível na matriz de entrada, com o filtro deslocando-se por toda a matriz segundo um valor de passo (*stride*) específico. No final, é gerado um mapa de características (*feature map*) que é uma matriz formada por todos os valores de saída somados de cada operação de convolução singular. O *stride* permite que o filtro se mova ao longo da matriz de entrada com um valor de passo que pode ser definido, indicando o número de pixéis que o mesmo deve deslocar-se em altura e largura da imagem. Assim, cada filtro é responsável por extrair uma determinada característica e a utilização de vários filtros em cada camada convolucional permite que sejam obtidas várias características distintas em simultâneo, aumentando a capacidade da rede capturar informações complexas (Chen et al., 2021). A Figura 2.16 mostra como é realizada a operação de convolução.

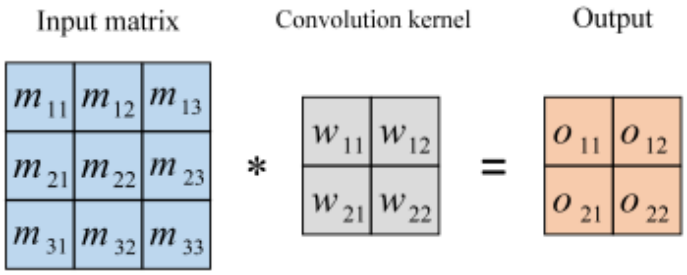

Figura 2.16: Operação convolucional para um *kernel* de tamanho 2x2, stride = 1 e padding = 0 (Chen et al., 2021).

Para se obter o *output* são realizadas as seguintes operações:

$$o_{11} = w_{11}m_{11} + w_{12}m_{12} + w_{21}m_{21} + w_{22}m_{22}$$
$$o_{12} = w_{11}m_{12} + w_{12}m_{13} + w_{21}m_{22} + w_{22}m_{23}$$
$$o_{21} = w_{11}m_{21} + w_{12}m_{22} + w_{21}m_{31} + w_{22}m_{32}$$
$$o_{22} = w_{11}m_{22} + w_{12}m_{23} + w_{21}m_{32} + w_{22}m_{33}$$

Uma das principais vantagens das camadas convolucionais é a capacidade de atualização dos pesos do *kernel* durante o treino, através de um processo chamado *gradient backpropagation*. Este processo permite ajustar os parâmetro treináveis do *kernel* para minimizar o erro entre as previsões do modelo e os valores reais. Para além disso ocorre também *parameter sharing*, o que significa que o mesmo filtro é aplicado em toda a imagem, reduzindo significativamente o número de parâmetros de treino.

2.3.3.2 Camada de *Pooling*

A camada de *pooling* é geralmente utilizada após a camada convolucional para simplificar e otimizar o processamento das informações extraídas. A principal função desta camada é reduzir a dimensionalidade dos mapas de características, o que, por sua vez, diminui o número de parâmetros e, consequentemente, a carga computacional da rede (Chen et al., 2021). Esta redução é alcançada através da aplicação de uma matriz fixa, ou janela de *pooling*, que percorre os valores de *input* de maneira a condensar as informações mais relevantes nessa zona e eliminar redundâncias. O processo de *pooling* pode ser realizado de diferentes maneiras, sendo as mais comuns o *max pooling* e o *average pooling*. No *max pooling* é selecionado o valor máximo de uma região específica do mapa de características, enquanto no *average pooling* é calculada a média dos valores dentro da região selecionada (Narciso, 2022). A Figura 2.17 demonstra como é realizada a operação de *pooling*.

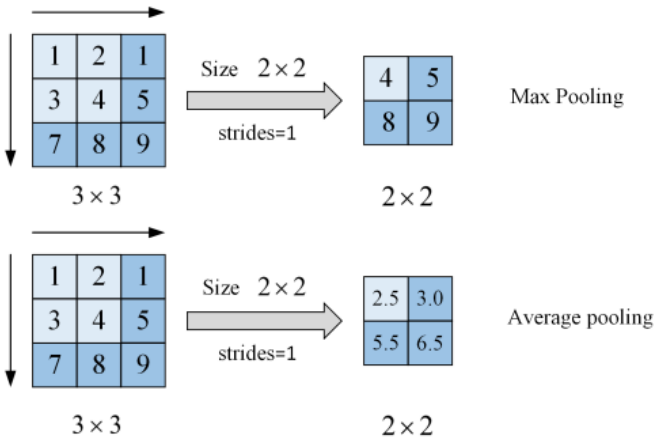

Figura 2.17: Operações de *max pooling* e *average pooling* (Chen et al., 2021).

2.3.3.3 Função de Ativação

As funções de ativação determinam se um neurónio deve ser ativado ou não com base nos seus *inputs*, pesos e *bias*. Estas funções introduzem não linearidade no modelo, permitindo que as redes neuronais aprendam e representem relações complexas entre os dados de entrada e saída, algo que seria impossível com uma simples combinação linear. Existem várias funções de ativação, cada uma com características e finalidades específicas (Chen et al., 2021).

- ***Softmax*:** É amplamente utilizada para problemas de classificação multiclasse, transformando os *outputs* da rede numa distribuição de probabilidades pelas diferentes classes, como mostra a equação 2.2. As probabilidades para cada classe são representadas por valores no intervalo [0, 1] e a soma de todas elas é igual a 1. Esta abordagem facilita a interpretação dos resultados, demonstrando qual classe tem a maior probabilidade de ser correta (Santos, 2020).

$$f(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}} \tag{2.2}$$

- ***Sigmoid*:** Representada pela equação 2.3, transforma os valores de *input* em valores de *output* no intervalo [0, 1], ou seja, valores muito maiores que 1 são transformados em 1 e valores muito menores que 0 são transformados em 0. Os valores intermédios são os mais sensíveis à função tal como se observa no gráfico da Figura 2.18. É frequentemente utilizada em problemas de classificação binária, onde os valores de *output* são interpretados como probabilidades (Gonçalves, 2020).

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.3}$$

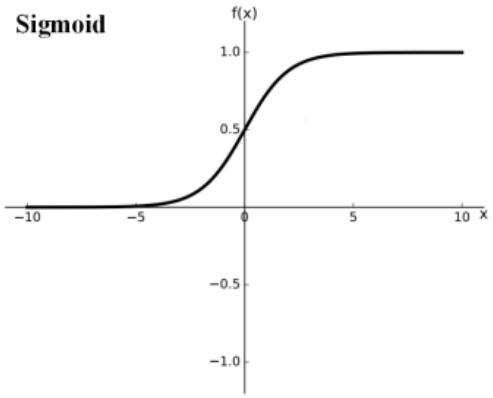

Figura 2.18: Função de ativação *sigmoid* (Chen et al., 2021).

- ***Rectified Linear Unit* (ReLU):** Retorna 0 para todos os valores de *input* negativos e o próprio valor para os positivos, como mostra a equação 2.4. A simplicidade e eficiência da função ReLU tornam-na uma das mais utilizadas em CNNs, pois permite que as redes neuronais convirjam mais rápido durante o treino (Gonçalves, 2020). A Figura 2.19 mostra como se comporta o gráfico da função ReLU.

$$f(x) = max(0, x) \tag{2.4}$$

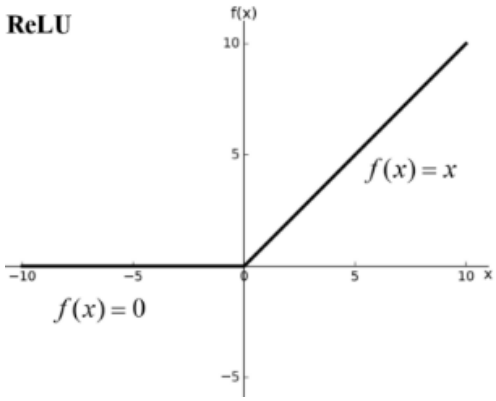

Figura 2.19: Função de ativação ReLU (Chen et al., 2021).

2.3.3.4 Função de Perda

A função de perda, também conhecida como função objetivo, é responsável por avaliar o quão bom é o desempenho do modelo durante o processo de treino. Esta função

quantifica a discrepância entre as previsões feitas pelo modelo e os valores reais dos dados de treino. O objetivo da rede é minimizar, ou nalguns casos maximizar, o valor dessa função, ajustando os seus parâmetros de modo a melhorar a qualidade das previsões. Existem várias funções de perda, cada uma adequada a diferentes tipos de problemas. Torna-se assim importante escolher aquela que melhor representa uma medida de sucesso para a tarefa que se tem em mão (Chollet, 2017). Para problemas de classificação, a seguinte função de perda é bastante utilizada:

- ***Categorical Cross-Entropy* (CCE):** Utilizada para problemas de classificação multiclasse. Só pode ser utilizada quando o *output* do modelo corresponde a uma distribuição de probabilidades (Narciso, 2022). Matematicamente é definida por:

$$CCE = -\sum_{i=1}^{m} y_i \log(p_i) \tag{2.5}$$

onde m representa o número total de dados de treino, y é o rótulo verdadeiro e p é a probabilidade prevista pelo modelo para o *input* pertencer à classe correta. É de destacar que as previsões incorretas têm maior impacto na função de perda do que as previsões corretas, devido à utilização da função logarítmica (Narciso, 2022).

Para tarefas de deteção de objetos, a função de perda tem como objetivo medir a diferença de posição entre a caixa delimitadora (*bounding box)* prevista e a *bounding box* verdadeira. Algumas das funções de perda mais utilizadas para este problema são:

- ***Smooth L1*:** Esta função combina características que proporcionam robustez em relação a valores atípicos (*outliers*) e suavidade no ajuste do erro, oferecendo uma minimização contínua mais eficiente durante o treino do modelo (Q. Wang et al., 2022). A função pode ser representada pela seguinte expressão:

$$Smooth\ L1 = \begin{cases} 0{,}5x^2 & se\ |x| < 1 \\ |x| - 0{,}5 & caso\ contrário \end{cases} \tag{2.6}$$

onde x é a diferença entre a coordenada prevista e a coordenada real da *bounding box*.

- ***Intersection over Union* (IoU):** Ao contrário de todas as anteriores, a função de perda IoU visa maximizar a sobreposição entre a caixa delimitadora prevista pelo modelo e a caixa delimitadora real. Este é um caso em que a função objetivo deve ser maximizada em vez de minimizada. Ao calcular a interseção e a união das áreas das duas caixas, a função utiliza todas as variáveis que definem cada caixa, tratando-as como um todo (Q. Wang et al., 2022). Esta abordagem resulta em previsões mais precisas. É dada pela seguinte expressão:

$$IoU\ loss\ = -\ln(\frac{I}{U}) \tag{2.7}$$

onde I representa a área de interseção entre a caixa prevista e a caixa real, isto é, a área em que ambas se sobrepõem e U é a área de união das duas caixas, ou seja, a área total abrangida pelas duas.

2.3.3.5 Camada Totalmente Conectada

A camada totalmente conectada (*fully connected layer*, FCL) é uma camada caracterizada pelo facto de cada neurónio estar conectado a todos os neurónios da camada anterior. Após as camadas convolucionais e de *pooling*, que são responsáveis por extrair e processar características locais da imagem, a FCL atua como um classificador. A sua função é receber as informações de alto nível extraídas nas camadas anteriores e produzir uma previsão final (Chen et al., 2021).

Durante o treino da rede, os parâmetros da FCL são ajustados através do processo de *backpropagation*. O *output* desta camada geralmente passa por uma função de ativação para normalizar as probabilidades antes de calcular a sua função de perda, o que facilita a comparação com as categorias reais da imagem (Chen et al., 2021).

2.3.3.6 Otimizadores

Os otimizadores são responsáveis por ajustar os pesos das camadas da rede, de modo a minimizar a função de perda durante o treino. Com isto, os otimizadores permitem levar o modelo à convergência, ou seja, para um ponto onde as previsões sejam o mais próximas possíveis dos valores reais.

O processo de treino de uma rede neuronal envolve calcular o gradiente da função de perda em relação aos parâmetros do modelo através de uma derivada de primeira ordem. Este gradiente indica a direção e a magnitude que os pesos devem seguir para reduzir o erro de previsão (Gonçalves, 2020). A taxa de aprendizagem (*learning rate,* LR) é o hiperparâmetro que define o tamanho dos passos que o otimizador deve dar ao ajustar os pesos com base no gradiente. Um *learning rate* muito alto diminui o tempo de treino, mas a convergência pode não acontecer. Por outro lado, um *learning rate* muito pequeno permite que a convergência seja atingida apesar de o tempo de treino ser demasiado elevado (Costa, 2020). Assim, a escolha da taxa de aprendizagem é crucial para o sucesso do treino do modelo. Existem vários otimizadores que podem ser utilizados, no entanto é importante escolher aquele que melhor se adequa ao problema em questão:

- ***Batch Gradient Descent* (BGD):** Atualiza os pesos de toda a rede em direção ao mínimo da função de perda. Para um conjunto de dados pequeno, o modelo converge rapidamente, no entanto, pode ser computacionalmente pesado e lento para conjuntos de dados maiores, visto que é necessário calcular o gradiente para todo o conjunto de treino em cada iteração (Chen et al., 2021).

- ***Stochastic Gradient Descent* (SGD):** Os pesos são atualizados após o cálculo do gradiente para apenas uma amostra arbitrária do conjunto de dados, tornando o processo mais rápido do que o BGD. Por outro lado, a convergência torna-se mais instável devido às atualizações frequentes, que introduzem aleatoriedade e ruído (Chen et al., 2021).

- ***Momentum*:** Acelera as atualizações de gradiente ao considerar também o histórico de gradientes passados, ajudando a superar mínimos locais e facilitando a convergência para o mínimo global. Isso é particularmente útil em funções de perda com topologias complexas, onde os mínimos locais podem dificultar o progresso do modelo (Chen et al., 2021).

2.3.3.7 Métricas

As métricas podem ser utilizadas tanto para avaliação e interpretação dos resultados obtidos nos dados que ainda não foram "vistos" pelo modelo, como para comparação direta entre modelos distintos (Howard & Gugger, 2020). A diferença entre as métricas

e as funções de perda é que as primeiras são destinadas ao utilizador e as últimas são para o modelo perceber como está a comportar-se durante o processo de treino.

Para perceber como a maioria das métricas são calculadas é necessário entender conceitos como verdadeiro positivo (*true positive*, TP), verdadeiro negativo (*true negative*, TN), falso positivo (*false positive*, FP) e falso negativo (*false negative*, FN). Supondo, por exemplo, um problema de classificação binária, o TP refere-se a uma previsão correta do modelo para a classe positiva enquanto o TN diz respeito a uma previsão correta para a classe negativa. Já o FP e o FN correspondem a uma previsão incorreta do modelo para a classe positiva e negativa, respetivamente (Rainio et al., 2024).

Para problemas onde existe mais do que uma classe estes conceitos também podem ser utilizados, basta serem aplicados a cada classe individualmente de modo a obter uma métrica que é representativa da própria classe. Para problemas de classificação normalmente são utilizadas as seguintes métricas:

- **Exatidão (*accuracy*):** É a métrica mais utilizada em algoritmos de ML e expressa a razão entre o número total de previsões corretas e o total de previsões efetudas pelo modelo (Narciso, 2022; Rainio et al., 2024). É definida pela seguinte expressão:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \in [0,1] \tag{2.8}$$

  Apesar da sua simplicidade, a exatidão pode ser mal interpretada em casos onde os dados não estão balanceados, o que pode gerar uma falsa impressão de bom desempenho. Por exemplo, num conjunto de dados onde uma classe é significativamente mais representada que a(s) outra(s), o modelo pode apresentar alta exatidão pelo facto de acertar a maioria das previsões da classe maioritária, enquanto falha em prever corretamente as previsões da(s) classe(s) minoritária(s) (Narciso, 2022; Rainio et al., 2024).

- **Precisão:** É o quociente entre as previsões TP e todas as previsões feitas pelo modelo para a classe considerada positiva, como indica a expressão 2.9. Um valor de precisão próximo de 1 indica que o modelo está a realizar previsões positivas com elevada confiança (Narciso, 2022; Rainio et al., 2024).

$$Precisão = \frac{TP}{TP + FP} \in [0,1] \quad (2.9)$$

- ***Recall*:** Mede a proporção de exemplos positivos que foram corretamente identificados em relação ao total de exemplos positivos no conjunto de dados. Pode ser tratada como a sensibilidade do modelo em prever corretamente os exemplos positivos. É crucial em situações onde é necessário garantir que a maioria dos exemplos positivos seja corretamente classificada (Narciso, 2022; Rainio et al., 2024). O *recall* é definido matematicamente como:

$$Recall = \frac{TP}{TP + FN} \in [0,1] \quad (2.10)$$

- ***F1-score***: É a média harmónica entre a precisão e *recall*, como evidenciado na expressão 2.11. É particularmente útil em problemas onde o conjunto de dados não está equilibrado, fornecendo uma única métrica que demonstra tanto a capacidade de o modelo prever corretamente exemplos positivos como a confiança dessas previsões. Um *F1-score* próximo de 1 indica que o modelo possui tanto alta precisão quanto alto *recall* (Rainio et al., 2024).

$$F1 = \frac{2 \times Precisão \times Recall}{Precisão + Recall} \in [0,1] \quad (2.11)$$

Para problemas de deteção, as métricas avaliam não só a classe a que pertence cada objeto como o quão próximas as caixas delimitadores previstas estão das suas posições corretas. A principal métrica utilizada para avaliar a posição das caixas delimitadoras previstas é a seguinte:

- **IoU:** Semelhante à função de perda IoU, avalia a sobreposição entre duas caixas delimitadoras: a caixa prevista pelo modelo e a caixa correta que foi anotada. Na prática, considera-se que uma caixa prevista corresponde a uma caixa correta se o valor de IoU for superior a um determinado limite (*threshold*) definido. Por exemplo, se consideramos um IoU limite de 0,5, isso significa que a caixa prevista tem de se sobrepor à caixa anotada em pelo menos 50% para ser considerada um TP. É de notar que, se várias caixas previstas se sobrepuserem à mesma caixa anotada, apenas aquela com o maior valor de IoU é considerada um TP, enquanto as outras são consideradas FP. Por outro lado, as caixas anotadas sem qualquer

correspondência são consideradas FN (Rainio et al., 2024). De forma simples, pode ser definida da seguinte forma:

$$IoU = \frac{I}{U} \tag{2.12}$$

2.3.3.8 *Underfitting* e *Overfitting*

O *underfitting* e *overfitting* são dois problemas comuns em algoritmos de ML que afetam a capacidade de generalização dos modelos. Essa generalização refere-se à capacidade de o modelo ser capaz de prever corretamente nos dados treino, bem como nos dados de teste.

O problema de *bias* e variância refere-se ao desafio de encontrar o equilíbrio entre a precisão do modelo nos dados de treino e a sua capacidade de generalizar para novos dados. Um alto *bias* reflete simplificações excessivas no modelo, levando ao *underfitting*, onde a perda nos dados de treino (*training loss*) e validação (*validation loss*) permanece alta devido à incapacidade de capturar a complexidade dos dados. Por outro lado, a variância elevada indica que o modelo é demasiado sensível aos dados de treino, resultando em *overfitting*, que pode ser observado quando a *training loss* diminui, mas a *validation loss* começa a aumentar (Jabbar & Khan, 2015; Santos, 2020). A Figura 2.20 apresenta uma comparação entre as diferentes situações que podem ocorrer.

Figura 2.20: Representação gráfica de *underfitting* (esquerda), generalização (centro) e *overfitting* (direita) (Weng, 2020).

2.3.3.9 Soluções para Evitar *Underfitting*

- **Aumento da complexidade do modelo:** Aumentar a profundidade da rede, adicionando mais camadas ou neurónios. Isso permite que o modelo tenha mais parâmetros para ajustar e, portanto, uma maior capacidade de aprender (Santos, 2020).

- **Aumento do número de épocas:** Treinar o modelo durante mais tempo pode ajudar a que sejam encontrados os parâmetros ótimos, minimizando a função de perda (Santos, 2020).

2.3.3.10 Soluções para Evitar *Overfitting*

- **Redução da complexidade do modelo:** Reduzir o número de camadas ou de neurónios na rede pode ajudar a evitar que o modelo memorize demasiado os dados de treino, melhorando a sua capacidade de generalização (Pothuganti, 2018).

- ***Dropout*:** Consiste em remover aleatoriamente neurónios de uma determinada camada durante o treino, forçando o modelo a aprender de forma mais robusta (Pothuganti, 2018).

- ***Early stopping*:** Interrompe o processo de treino quando o desempenho nos dados de validação começa a estagnar durante um determinado número de épocas (Pothuganti, 2018).

- ***Data augmentation*:** É um método que permite aumentar o conjunto de dados de treino sem a necessidade de recolher novos. Envolve a aplicação de várias transformações aos dados existentes, como rotação, translação, espelhamento e redimensionamento (Gu et al., 2018). Existem duas formas de aplicar *data augmentation* ao conjunto de dados: *data augmentation offline* e *data augmentation online*. A primeira consiste em gerar novas imagens antes do processo de treino, armazenando-as num diretório para posteriormente serem utilizadas no treino do modelo. Apesar de esta abordagem ser muito útil para equilibrar conjuntos de dados que possuem, originalmente, uma distribuição desequilibrada pelas várias classes, pode ser limitador ao nível do armazenamento disponível. A segunda diz respeito à geração de novas imagens em cada iteração durante o treino do modelo. As imagens geradas são sempre diferentes em cada época, o que significa que o modelo recebe sempre uma variação diferente da imagem original. Apesar de o armazenamento extra não constituir um problema, visto que as imagens são geradas no momento e descartadas imediatamente após a sua utilização, é difícil controlar quais as imagens que são geradas e passadas ao modelo (Santos, 2020).

2.3.3.11 Transfer Learning

Transfer learning é uma técnica que permite reutilizar o conhecimento adquirido por um modelo pré-treinado para resolver uma tarefa diferente, mas que de certa forma está relacionada. Este método reduz significativamente o tempo de treino e a necessidade de grandes quantidades de dados, aproveitando os parâmetros treináveis de um modelo previamente treinado num grande conjunto de dados, como o *ImageNet* (Deng et al., 2009) ou *COCO dataset* (Lin et al., 2014). Após carregar esses parâmetros, o modelo é ajustado ao novo problema durante o treino (Leonard, 2019).

Tal como já foi referido, as camadas mais próximas à entrada de uma CNN capturam características genéricas enquanto as camadas mais profundas aprendem características mais abstratas e específicas. Para adaptar um modelo pré-treinado a um novo domínio, é possível "congelar" parte das camadas e treinar apenas as camadas finais, ou, em casos onde o novo domínio seja muito semelhante ao original, treinar apenas o classificador final (Santos, 2020). Assim sendo, o *transfer learning* pode ser aplicado de três formas diferentes:

- **Ajustar todo o modelo**: Treinar todas as camadas do modelo, adaptando os pesos tanto das camadas convolucionais como do classificador. Este método é útil quando o poder computacional não é um problema (Santos, 2020).

- **Congelar parte do modelo**: Manter algumas camadas convolucionais inalteradas, treinando apenas as restantes e o classificador. Esta abordagem é vantajosa quando o novo conjunto de dados é pequeno e diferente do conjunto original, podendo variar o número de camadas treinadas consoante o nível de diferença entre ambos os problemas (Santos, 2020).

- **Treinar apenas o classificador**: Usado quando o novo problema é muito semelhante ao original. Apenas o classificador é treinado, enquanto a base convolucional é mantida intacta. Muito utilizado quando o poder computacional disponível é reduzido (Santos, 2020).

A Figura 2.21 ilustra as diferentes formas de aplicar esta técnica.

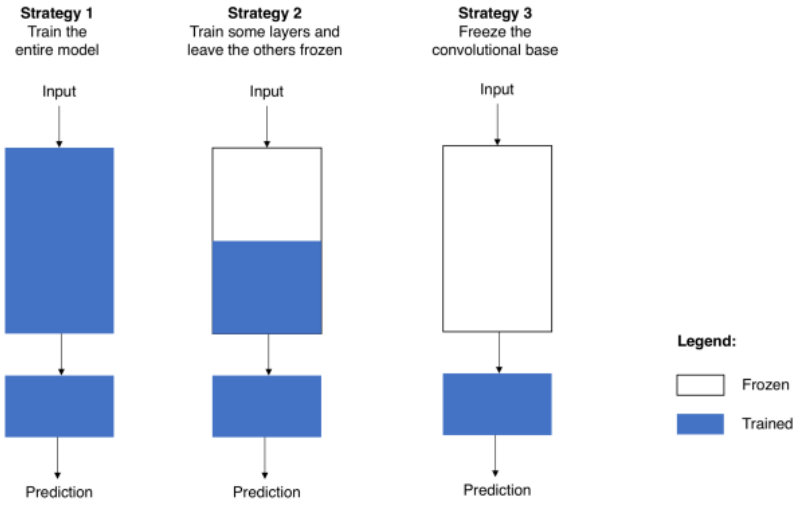

Figura 2.21: Diferentes técnicas de transfer learning (Santos, 2020).

2.4 Classificação de Imagens

Nesta secção são apresentadas várias arquiteturas de CNNs e a sua evolução, que tem levado a avanços significativos na área da visão computacional, principalmente em tarefas de classificação.

2.4.1 Evolução das CNNs para Tarefas de Classificação

LeCun et al. propôs, em 1998, a LeNet-5, uma das primeiras arquiteturas CNN (LeCun et al., 1998). A LeNet-5 é composta por duas camadas convolucionais, seguidas de duas camadas de *pooling*, duas camadas totalmente conectadas e uma camada de saída. Apesar da sua utilização em tarefas relacionadas com reconhecimento de documentos manuscritos apresentar um bom desempenho para a época, o mesmo não aconteceu para *datasets* maiores e outros tipos de problemas (Alom et al., 2019).

A grande reviravolta ocorreu em 2012 com a introdução da AlexNet por Krizhevsky et al. que revolucionou a área do ML e visão computacional ao vencer a competição *ImageNet Large Scale Visual Recognition Challenge*. A *AlexNet* trouxe inovações como o uso da função de ativação ReLU, camadas convolucionais mais profundas e *dropout* como técnica de regularização. A sua arquitetura consiste em cinco camadas convolucionais, algumas seguidas por camadas de *max-pooling*, e três camadas totalmente conectadas (Krizhevsky et al., 2017; Alom et al., 2019).

Em 2014, a arquitetura VGGNet (Simonyan & Zisserman, 2015) demonstrou que o aumento da profundidade da rede melhora significativamente o seu desempenho em tarefas de reconhecimento e classificação. Para além de maior profundidade, ou seja, maior número de camadas, a arquitetura VGG utiliza filtros nas camadas convolucionais de tamanho 3x3, muito inferior ao que se verificava na AlexNet. As variações mais populares, como a VGG16 e VGG19, têm 16 e 19 camadas, respetivamente (Alom et al., 2019).

A arquitetura VGG16, representada na Figura 2.22, é composta por cinco blocos conectados em série. Os primeiros dois blocos são formados por duas camadas convolucionais idênticas seguidas por uma camada de *max-pooling* e os restantes três blocos são formados por três camadas convolucionais idênticas seguidas por uma camada de *max pooling*. As camadas convolucionais mantêm as dimensões da entrada inalteradas, enquanto as camadas de *pooling* reduzem essas dimensões para metade, facilitando a extração progressiva de características cada vez mais complexas. Após os cinco blocos, a rede possui três camadas totalmente conectadas, das quais uma funciona como camada de saída. Por fim, é aplicada a função *softmax* que transforma os valores da camada de saída em probabilidades, permitindo que seja atribuída uma classe à entrada (Chen et al., 2021).

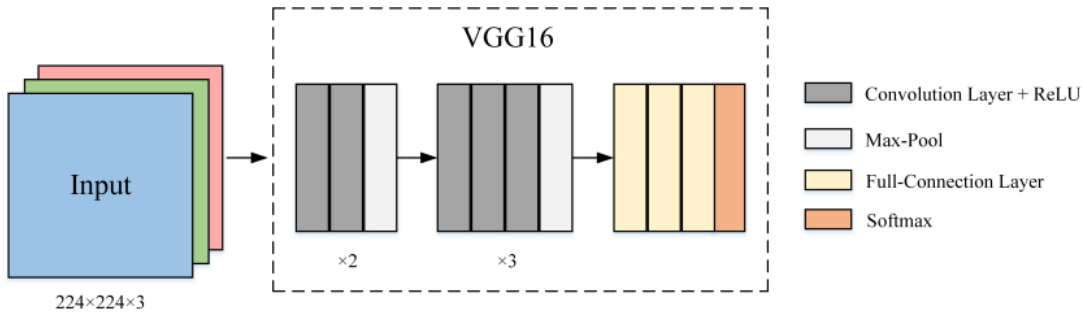

Figura 2.22: Arquitetura da rede VGG16 (Chen et al., 2021).

Apesar de até então o aumento da profundidade da rede parecer demonstrar um aumento no seu desempenho, verificou-se que existe um determinado número de camadas a partir do qual esse desempenho se começa a degradar. Esta situação deve-se ao problema dos *vanishing gradients*, que ocorre em redes profundas quando os gradientes utilizados para ajustar os pesos durante o treino se tornam extremamente pequenos à medida que são propagados para trás ao longo das várias camadas (Gonçalves, 2020).

Em 2015, para resolver esse problema, uma equipa de investigadores da *Microsoft* apresentou o conceito de *skip connections* através da arquitetura *Residual Network* (ResNet) (He et al., 2016), conquistando o primeiro lugar na competição ILSVRC. As *skip connections*, que integram os *residual blocks*, permitem que o fluxo do gradiente "salte" uma ou mais camadas, reutilizando as ativações de camadas anteriores. Esta técnica garante que o gradiente continua a fluir de forma eficaz da última camada até à primeira, mesmo em redes muito profundas. (Alom et al., 2019; Gonçalves, 2020).

O objetivo dos *residual blocks* não é aprender diretamente o resultado esperado $H(x)$, mas sim a diferença entre esse resultado e a entrada x, denominada de função residual ($F(x) = H(x) - x$). Após o cálculo dessa diferença, a função residual é somada de volta à entrada original x, obtendo o resultado desejado: $H(x) = F(x) + x$. Uma das vantagens que esta abordagem oferece é a facilidade de treino, visto que o modelo precisa apenas de aprender a função residual, que geralmente é uma pequena variação do mapeamento identidade ($H(x) \approx x$), permitindo ajustes mais rápidos e eficientes. Além disso, as *skip connections* asseguram que a informação e os gradientes fluam de forma eficiente ao longo da rede, prevenindo problemas como os *vanishing gradients* (Chen et al., 2021). A Figura 2.23 ilustra esse processo, comparando-o ao que se verifica numa CNN comum.

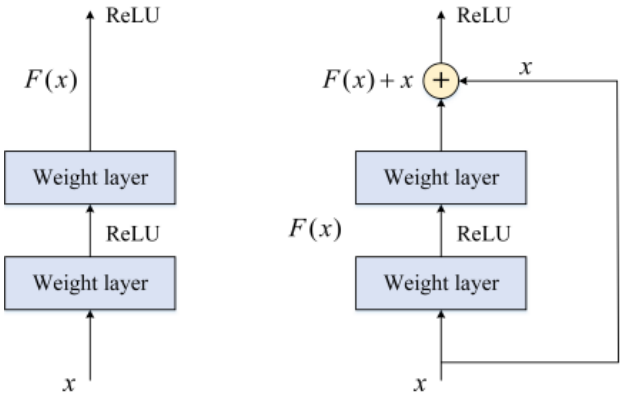

Figura 2.23: Comparação entre a aprendizagem de uma CNN comum (esquerda) e aprendizagem residual (direita) (Chen et al., 2021).

2.5 Deteção de Objetos

A tarefa de deteção envolve identificar não só o que está presente na imagem (classificação), mas também onde está posicionado (localização). Para isso, são utilizados algoritmos que preveem a categoria do objeto e desenham uma caixa delimitadora à volta da região do mesmo, indicando a sua posição exata na imagem. Estes modelos têm

evoluído significativamente ao longo dos anos, especialmente com o desenvolvimento das CNNs (Cheng et al., 2018).

2.5.1 Evolução das Arquiteturas para Tarefas de Deteção

A *Region-based* CNN (R-CNN) (Girshick et al., 2014) marcou um dos primeiros grandes avanços na deteção de objetos utilizando CNNs. A R-CNN utiliza o método de *selective search* para gerar várias propostas de regiões de interesse (*region of interest*, RoI) na imagem, nas quais os objetos podem estar presentes. De seguida, uma CNN pré-treinada é utilizada para extrair características dessas regiões de forma e realizar a classificação na respetiva classe e a previsão da localização da caixa delimitadora. Dado que cada região é processada individualmente, este processo torna-se extremamente demorado (Suguna et al., 2021). A Figura 2.24 representa esquematicamente o funcionamento de uma arquitetura R-CNN.

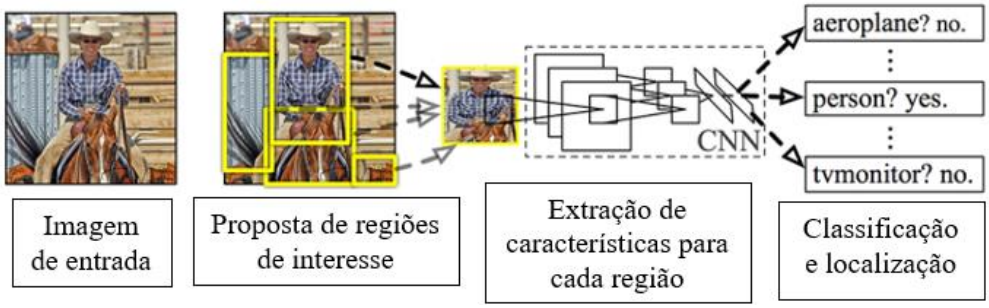

Figura 2.24: Representação do funcionamento de uma arquitetura R-CNN. Adaptado de (Rosebrock, 2017).

Para resolver essa limitação, surgiu a *Fast* R-CNN (Girshick, 2015), onde inicialmente toda a imagem é passada por uma CNN que gera um mapa de características, a partir do qual as regiões de interesse são posteriormente propostas. Por fim, a rede faz a classificação e o ajuste das caixas delimitadoras. Esta abordagem reduz significativamente o tempo de computação em até 10 vezes em relação à R-CNN (Suguna et al., 2021).

A *Faster* R-CNN (Ren et al., 2016), uma evolução da *Fast* R-CNN, incorpora uma *Region Proposal Network* (RPN), eliminando a necessidade de métodos externos de geração de propostas como a *selective search*. A RPN é integrada diretamente na arquitetura CNN, gerando propostas de regiões de forma eficiente, visto que é treinada juntamente com a rede convolucional. A RPN faz deslizar uma "janela" sobre o mapa de características gerado pela última camada convolucional da CNN compartilhada e utiliza caixas pré-

definidas em diferentes escalas e proporções (*anchor boxes*) para sugerir regiões na imagem onde um objeto pode estar presente. Cada uma das propostas é avaliada em dois aspetos: se contém ou não um objeto (classificação) e como ajustar a caixa para melhor enquadrar o objeto (regressão), através da *Intersection over Union loss,* que determina a qualidade da proposta. A *Faster* R-CNN é um dos modelos mais utilizados para tarefas de deteção de objetos, uma vez que oferece um bom equilíbrio entre precisão e velocidade (Joiya, 2022; Suguna et al., 2021).

O conceito de *transfer learning* também é amplamente utilizado na deteção de objetos. Modelos como o *Faster* R-CNN utilizam redes pré-treinadas, como as VGG ou ResNet, para extrair características da imagem, acelerando o processo de treino e melhorando o desempenho em conjuntos de dados menores (S. Y. Wang & Guo, 2021). A Figura 2.25 mostra o seu funcionamento.

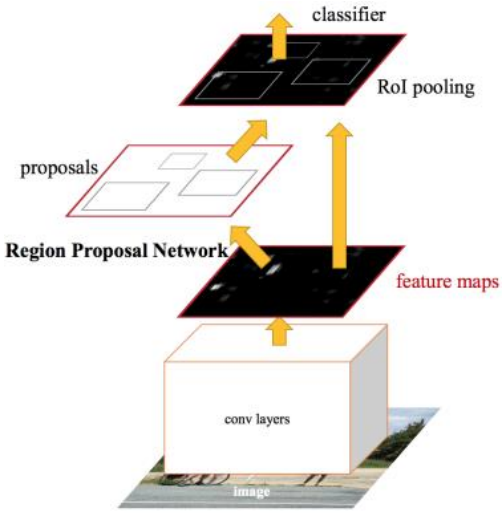

Figura 2.25: Representação do funcionamento de uma arquitetura *Faster* R-CNN (Rosebrock, 2017).

Por fim, a introdução da *Feature Pyramid Network* (FPN) como complemento ao *Faster* R-CNN trouxe ainda mais avanços na tarefa de deteção. A FPN é responsável por construir uma pirâmide de características em diferentes escalas, permitindo a deteção eficaz de objetos com várias dimensões (S. Y. Wang & Guo, 2021).

2.6 Técnicas de ML Aplicadas à Fractografia

São diversos os trabalhos que têm demonstrado a utilização de CNNs para tarefas como a classificação de mecanismos de falha ou superfícies de fratura, para além dos modelos de deteção para localizar regiões de interesse em imagens fractográficas, como locais de iniciação de fissuras . A seguir, são apresentados alguns exemplos do estado-da-arte de modelos de ML aplicados na análise fractográfica.

2.6.1 Tarefas de Classificação

Em 2020, Bastidas-Rodriguez et al. apresentaram um estudo sobre a classificação de modos de fratura em materiais metálicos em três classes principais: fratura dúctil, por fadiga e frágil. Foram avaliados dois conjuntos de dados: um em escala real e outro com imagens obtidas através do MEV. Foram também testadas diversas arquiteturas de CNNs, como a AlexNet, VGG16 e ResNet50, além de ter sido proposta uma arquitetura adaptada chamada *Deep Adaptive Wavelet Network* (DAWN), que incorpora métodos avançados para extração de características de textura. Os resultados demonstraram que a abordagem DAWN superou as CNNs tradicionais no conjunto de dados obtidos com o MEV, alcançando uma exatidão de 87,1% e *F1-score* de 86,4%. (Bastidas-Rodriguez et al., 2020)

No mesmo ano, Yamagiwa et al. realizaram a classificação de imagens de superfícies de fratura obtidas com o MEV em seis classes: *dimples*, facetas, contornos de grão, fadiga com baixa ampliação, fadiga com ampliação média e estrias. O estudo utilizou a arquitetura CNN *Inception V3* para processar e classificar as imagens e o conjunto de dados beneficiou de técnicas de *data augmentation* para aumentar a sua dimensão e variedade. O modelo alcançou uma exatidão de 92% nos dados de teste. (Yamagiwa, 2020)

Endo et al. propuseram em 2022 um método para classificação hierárquica de três modos de fratura utilizando imagens de superfícies de fratura obtidas com o MEV com baixa e alta ampliação. O método consistiu na realização de classificação binária em duas etapas, onde inicialmente é verificada a presença de *dimples* nas imagens de baixa ampliação para distinguir entre fratura dúctil ou por fadiga. De seguida, para fraturas por fadiga, a presença de estrias é examinada em imagens de alta ampliação. A observação de estrias confirma a fratura por fadiga, enquanto a sua ausência indica uma

fratura do tipo *microstructure-dependent*. A classificação binária é feita para cada segmento da imagem (*patch*) através de um modelo de *logistic regression* e com base na extração de características de textura e a votação final é feita com base na classificação de cada *patch* individualmente. Na classificação dos três modos de fratura foi obtida uma exatidão de aproximadamente 90%, tornando esta abordagem particularmente útil para conjuntos de dados pequenos e com uma lógica bastante fácil de ser interpretada (Endo et al., 2022).

2.6.2 Tarefas de Deteção

Em 2020, Wang et al. desenvolveram técnicas de ML para deteção de locais de iniciação de fissuras por fadiga (*fatigue crack initiation sites*, FCIS) em imagens fractográficas de materiais metálicos. A arquitetura utilizada foi a *Deeply Supervised Object Detector*, reconhecida pelo seu elevado desempenho em conjuntos de dados limitados e capacidade de ser treinada a partir do zero. Apesar de esta arquitetura apresentar uma precisão acima da média na deteção de objetos noutras áreas, o mesmo não se verificou para este caso particular, onde o modelo mostrou dificuldades em generalizar para diferentes tipos de FCIS e resoluções de imagens (S. Y. Wang et al., 2020).

Com base nos desafios encontrados, os mesmos autores sugeriram o aumento do conjunto de dados e a utilização de técnicas de *transfer learning* para melhorar o desempenho do modelo. Assim, em 2021, realizaram um estudo comparativo entre diferentes modelos de ML com utilização de *transfer learning* para a deteção de FCIS. Foram selecionados três modelos utilizando o *Faster* R-CNN como detetor e três arquiteturas CNN, nomeadamente a VGG-16, ResNet-101, e FPN, como extratores de características. O estudo demonstrou que a combinação de técnicas de ML com *transfer learning* melhora significativamente o desempenho na deteção de FCIS, atingindo até 95,9% de precisão com a ResNet-101 (S. Y. Wang & Guo, 2021).

No mesmo ano, Zhao et al. identificaram microfissuras em materiais metálicos em imagens obtidas pelo MEV. O modelo desenvolvido destaca-se pela aplicação de *bounding boxes* rotativas, otimizadas para capturar as características locais das fissuras. Esta técnica permite melhorar a precisão na deteção ao reduzir as diferenças entre as características capturadas em *bounding boxes* consecutivas. A abordagem permitiu obter uma precisão de deteção de 71,12%, com um mIoU máximo de 64,13%, demonstrando também eficácia na deteção de microfissuras em imagens de diferentes ampliações e *backgrounds* (Zhao et al., 2021).

Capítulo 3

3 Metodologia

Neste capítulo é apresentada a metodologia adotada, na qual são detalhados os processos de aquisição e preparação das imagens utilizadas, bem como a escolha de cada um dos modelos individualmente. Inicialmente é descrita a abordagem de aquisição das várias imagens, o pré-processamento realizado para garantir a qualidade das mesmas e o modo como foram organizadas para posterior utilização. De seguida é abordado o processo de seleção das arquiteturas, as configurações e hiperparâmetros de treino e o processo de otimização levado a cabo para obter os melhores modelos possíveis. Por fim, é explicada a metodologia adotada para validar os modelos, destacando a importância da *cross validation* na obtenção de resultados mais robustos.

3.1 Ambiente de Desenvolvimento

Os modelos foram treinados no computador pessoal equipado com uma GPU NVIDIA GTX 1050 Ti com suporte à API CUDA 11.8 e à biblioteca cuDNN. Todo o código foi implementado em *Python* 3.11, num ambiente interativo *Jupyter Notebook*.

Para a tarefa de classificação utilizou-se o *TensorFlow* 2.0 com *Keras*, facilitando a criação e treino do modelo. Além disso, foram importadas bibliotecas como *Numpy* para manipulação de dados numéricos, *Matplotlib* para geração de gráficos, e *Scikit-learn* para o cálculo das métricas.

Para a tarefa de deteção foi utilizado o *Detectron 2*, uma biblioteca desenvolvida pela *Facebook AI Research Team*, especializada em tarefas de visão computacional. As principais bibliotecas utilizadas para este problema foram *PyTorch*, para manipulação de tensores e treino com GPU, *Matplotlib* para geração de gráficos e *OpenCV* para manipulação de imagens. Também foram utilizadas funções específicas do *Detectron 2*, como o *DefaultPredictor*, para realizar previsões em imagens de teste.

3.2 Metodologia para Classificação

3.2.1 Aquisição das Imagens

A aquisição das imagens dos mecanismos de falha para o modelo de classificação foi realizada em quatro recolhas distintas, em colaboração com várias instituições. Este processo garantiu a criação de um *dataset* bastante diversificado e representativo das três classes principais de mecanismos de falha em ligas metálicas utilizadas em aeronaves da FAP: sobrecarga, fadiga e corrosão. As imagens obtidas representam superfícies de fratura reais de componentes aeronáuticos como o olhal do trem de nariz de uma aeronave C-130, uma jante, uma turbina, canos do canhão da aeronave F-16, entre outros.

A primeira e segunda fases de recolha foram realizadas no Instituto Superior Técnico, em Lisboa, utilizando o MEV disponível, Hitachi S2400, sem qualquer preparação prévia das amostras observadas. A terceira e quarta fases foram realizadas no Laboratório Nacional do Medicamento, pertencente à Unidade Militar Laboratorial de Defesa Biológica e Química do Exército Português, onde foi utilizado o MEV disponível, Hitachi TM3030Plus. Durante as duas últimas recolhas foi implementada uma metodologia de preparação rigorosa das amostras, a fim de garantir a qualidade e a clareza das superfícies observadas. O procedimento de preparação seguiu os seguintes passos:

- **Lavagem e desengorduramento:** As amostras foram colocadas em tubos com acetona e submetidas a agitação com Vortex durante 1 minuto à velocidade máxima (Figura 3.1). Desta forma, qualquer gordura ou contaminação superficial que pudesse interferir na qualidade das imagens foi removida.

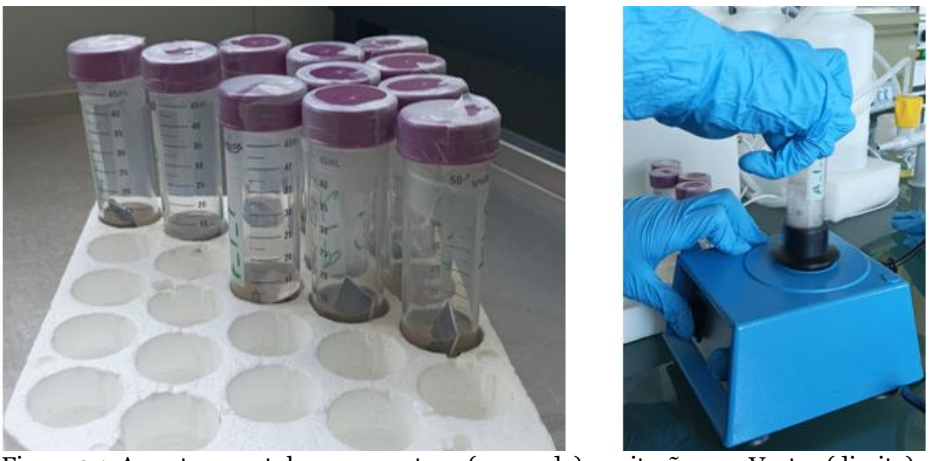

Figura 3.1. Amostras em tubos com acetona (esquerda) e agitação com Vortex (direita).

- **Lavagem com éter dietílico**: Após a acetona, foi utilizado éter dietílico para um segundo processo de desengorduramento, seguido novamente pela agitação com o Vortex.

- **Lavagem com água destilada e etanol**: Para garantir que nenhum resíduo de solventes permanecesse nas amostras, foi feita uma lavagem com água destilada e, de seguida, com etanol.

- **Secagem em estufa**: As amostras foram colocadas numa estufa para garantir que estavam completamente secas antes da análise.

- **Configurações do MEV**: O MEV foi configurado com *backscattering* a 15 kV e uma ampliação de 2000x, o que permitiu a visualização detalhada das superfícies.

Este processo de preparação cuidadosa, realizado nas duas últimas recolhas de imagens, foi fundamental para garantir a qualidade das imagens utilizadas no treino do modelo. Por outro lado, a diversidade de superfícies de fratura e os diferentes contextos de aquisição garantiram a criação de um *dataset* robusto. A Figura 3.2 demonstra uma imagem recolhida para cada classe, evidenciando a qualidade das mesmas.

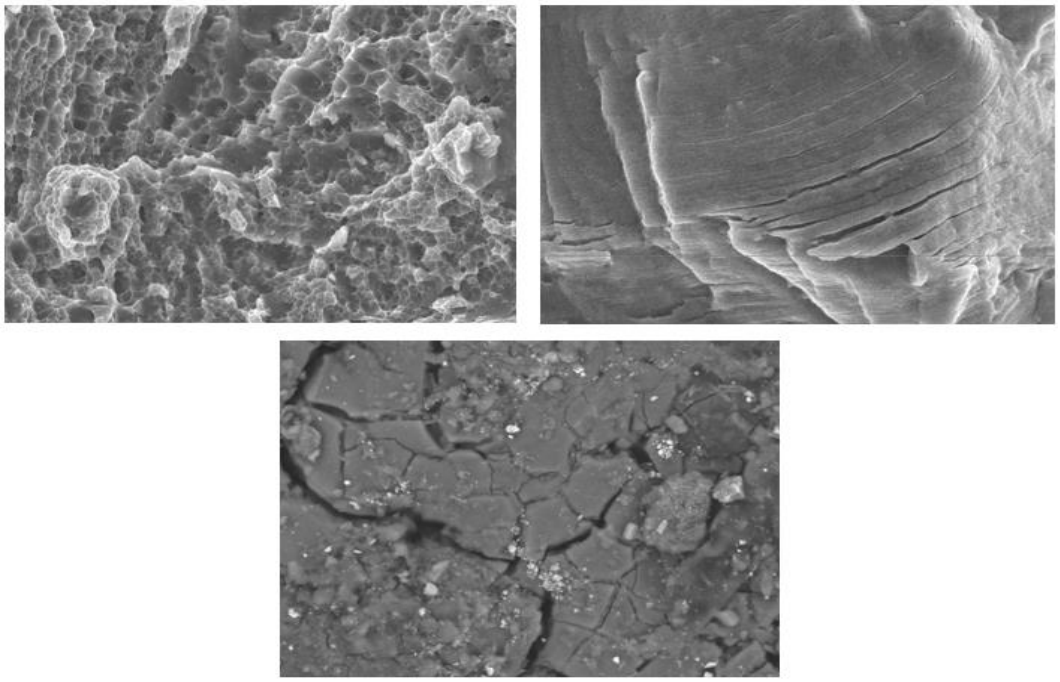

Figura 3.2. Exemplos de imagens recolhidas com o MEV para a classe de sobrecarga (em cima à esquerda), fadiga (em cima à direita) e corrosão (em baixo).

Ao longo das quatro fases de recolha foram capturadas 309 imagens. A Figura 3.3 mostra a distribuição das imagens por classe, permitindo uma análise clara da composição total do *dataset*.

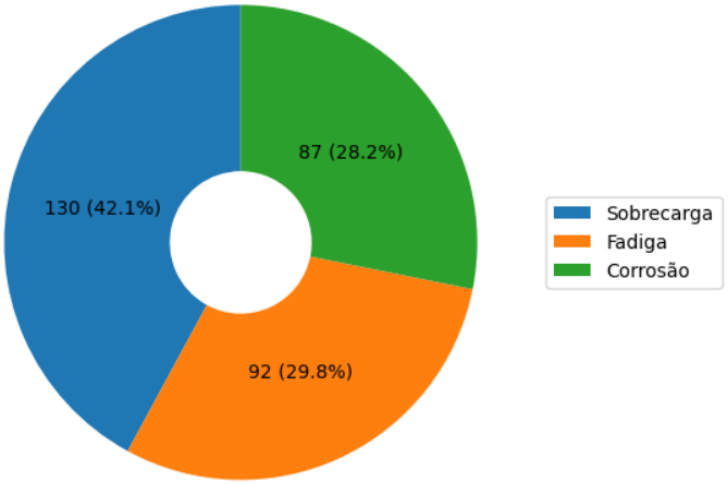

Figura 3.3. Distribuição das imagens por classe.

3.2.2 Pré-processamento das Imagens

Após a aquisição das imagens, segue-se o pré-processamento das mesmas. Esta etapa é crucial na preparação dos dados para o treino do modelo de classificação e inclui os seguintes passos:

- **Redimensionamento:** As imagens originais possuíam tamanhos diferentes entre si e relativamente grandes, como por exemplo 2900 x 2175 ou 1280 x 1100 píxeis, o que compromete a consistência dos dados de entrada e aumenta a carga computacional. A arquitetura utilizada para a tarefa de classificação, tal como irá ser explicado mais à frente (secção 3.2.4), espera uma entrada de tamanho fixo padrão de 224 x 224 píxeis. Assim, todas as imagens foram redimensionadas para esse tamanho antes de serem fornecidas ao modelo.

- **Conversão de formatos:** A arquitetura escolhida foi projetada para receber, por padrão, imagens com três canais de cor no formato RGB. Apesar de as imagens obtidas pelo MEV se encontrarem na escala de cinza, as mesmas apresentam três canais. Estes três canais são idênticos, ou seja, o mesmo canal monocromático é repetido três vezes para cada imagem. Desta forma, foi possível manter a compatibilidade com a arquitetura utilizada sem necessidade de qualquer conversão adicional de formatos.

- **Normalização:** Cada pixel foi dividido por 255, de tal forma que passaram para valores entre 0 e 1. Este procedimento de normalização é essencial, pois padroniza a escala dos dados, o que permite melhorar a estabilidade do treino e acelerar o processo de aprendizagem.

- ***Data augmentation*:** De modo a melhorar a generalização do modelo e evitar *overfitting*, foram aplicadas técnicas de *data augmentation*, com recurso à biblioteca *ImageDataGenerator*. As transformações aplicadas consistiram em rotações com um ângulo de até 15º, deslocamento horizontal e vertical de até 10%, *zoom* de até 10%, seja de aumento ou redução, ajuste no brilho variando entre 80% e 120% do original para simular diferentes condições de iluminação, *flip* horizontal e vertical e, por fim, foi aplicado o *fill mode reflect* para preencher áreas vazias das imagens geradas por transformações de rotação, por exemplo. Estas transformações referem-se à *data augmentation offline*, realizada antes de as imagens serem passadas ao modelo, com exemplos apresentados na Figura 3.4. A *data augmentation online*, realizada a cada época, consistiu apenas em transformações simples, como o *flip* horizontal e vertical. Estas transformações permitem que as imagens passadas ao modelo a cada época sejam sempre ligeiramente diferentes entre si, aumentando a sua capacidade de generalização.

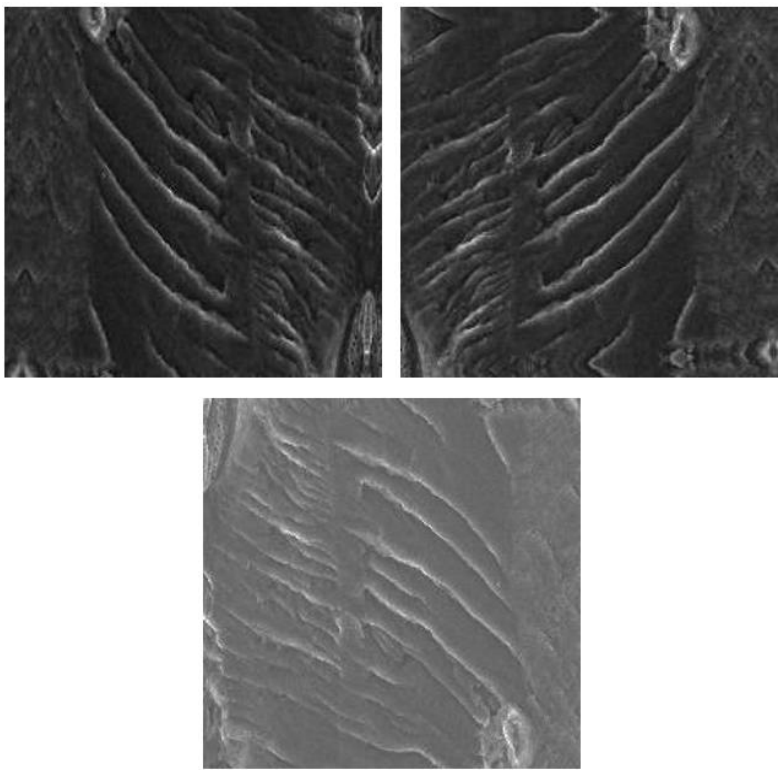

Figura 3.4. Exemplo de utilização de *data augmentation* para criar novas imagens (cima) a partir da original (baixo).

No final deste processo o número de imagens do *dataset* foi expandido para 797, seguindo a distribuição que se apresenta na Figura 3.5.

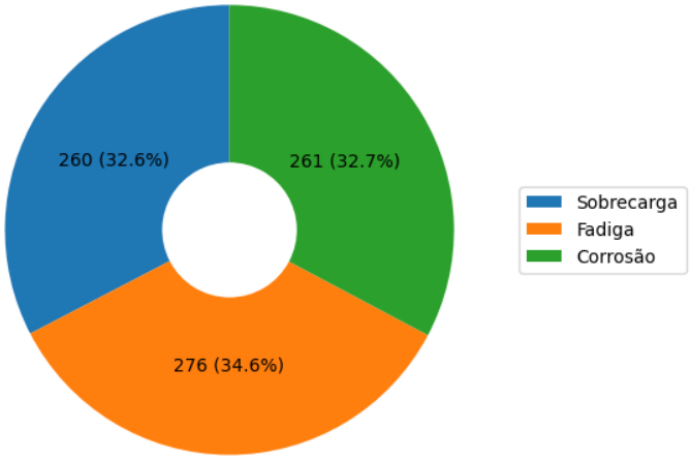

Figura 3.5. Distribuição das imagens pelas três classes após *data augmentation*.

Divisão do *dataset*: As imagens foram divididas em conjuntos de treino (70%), validação (20%) e teste (10%), de forma aleatória e mantendo a proporção das classes, com recurso à função *train_test_split* da biblioteca *scikit-learn*.

3.2.3 Organização das Imagens

As imagens destinadas ao modelo de classificação foram separadas em pastas de acordo com a classe a que pertenciam, isto é, três pastas, cada uma correspondente a um dos mecanismos de falha. A Figura 3.6 representa a estrutura de organização adotada.

```
.../dataset
    │
    ├── sobrecarga
    │       │
    │       ├── sobrecarga1.jpg
    │       ├── sobrecarga2.jpg
    │       └── ...
    │
    ├── fadiga
    │       │
    │       ├── fadiga1.jpg
    │       ├── fadiga2.jpg
    │       └── ...
    │
    └── corrosão
            │
            ├── corrosão1.jpg
            ├── corrosão2.jpg
            └── ...
```

Figura 3.6. Organização das imagens por classe.

3.2.4 Arquitetura do Modelo

O modelo de classificação foi estruturado com base na arquitetura VGG16, reconhecida principalmente pela sua simplicidade estrutural e alta capacidade de extração de detalhes complexos, como é o caso das imagens dos mecanismos de falha, através de uma hierarquia de camadas convolucionais profundas. A escolha desta arquitetura é, portanto, fundamentada tanto na literatura existente como nas exigências específicas da tarefa em questão.

Foram avaliadas diversas arquiteturas, incluindo aquelas que apresentaram melhores resultados no estudo de Bastidas-Rodriguez et al. (secção 2.6.1), mas a VGG16 destacou-se por alcançar um desempenho inicial mais promissor, enquanto outras redes estagnavam em valores baixos de exatidão ou apresentavam sinais de *overfitting* precoce. Assim, esta arquitetura demonstrou ser uma base sólida para a tarefa de classificação quando devidamente ajustada (Bastidas-Rodriguez et al., 2020).

Para além disso, o *dataset* reduzido e a complexidade associada às características das imagens tornam o uso de *transfer learning* essencial, permitindo que o modelo aproveite o conhecimento adquirido previamente num conjunto de dados amplo e diversificado como a *ImageNet* e se adapte mais facilmente ao problema em causa.

3.2.5 Configurações de Treino e Hiperparâmetros

As camadas totalmente conectadas originais da VGG16, responsáveis pela classificação das 1000 classes na *ImageNet*, foram removidas para adaptar a arquitetura às necessidades da tarefa em questão. Desta forma, foi adicionado um conjunto personalizado de camadas, denominado ao longo desta dissertação por *classification head*, projetado para transformar os mapas de características extraídos pelas camadas convolucionais em previsões para as três classes definidas no *dataset* utilizado. Este novo conjunto é composto por uma camada *flatten*, que converte a saída tridimensional dos mapas de características num vetor unidimensional, seguida por duas camadas totalmente conectadas, que seguem a lógica de redução da dimensionalidade progressiva da arquitetura original, com função de ativação *ReLU* e *dropout* de 20% para evitar *overfitting*. Por fim, uma camada de saída com ativação *Softmax* realiza a classificação das imagens segundo cada classe. A diferença entre a arquitetura original e modificada da VGG16 está representada esquematicamente na Figura 3.7.

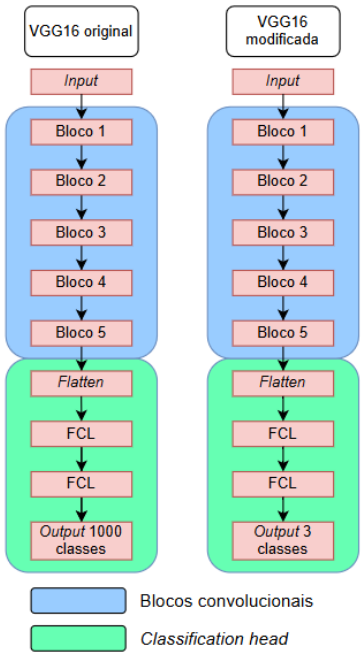

Figura 3.7: Representação da arquitetura VGG16 original e modificada.

Durante a fase inicial, todas as camadas da VGG16 foram congeladas garantindo que o foco fosse apenas o treino das camadas pertencentes à *classification head*. Com esta abordagem o modelo beneficiou das características genéricas previamente aprendidas na *ImageNet* para facilitar a sua adaptação às especificidades do problema. Nesta etapa foi utilizado o otimizador SGD com uma taxa de aprendizagem inicial de 3e-4 e *momentum* de 0,9. A função de perda escolhida foi a *categorical cross-entropy*, adequada para problemas de classificação multiclasse. Com o objetivo de monitorizar o treino e evitar problemas de *overfitting* foram definidos alguns *callbacks*, nomeadamente *early stopping*, redução dinâmica da taxa de aprendizagem e *model checkpoint*. O *early stopping* foi configurado para interromper o treino em caso de não se verificarem melhorias na *validation loss* superiores a 0,01 até 5 épocas consecutivas. A redução dinâmica da taxa de aprendizagem foi ativada sempre que a *validation loss* estabilizasse por 3 épocas consecutivas, reduzindo o *learning rate* em 20%, começando com um valor inicial de 3e-4. Por fim, o *model checkpoint* guardou automaticamente o melhor modelo com base no menor valor da *validation loss*. Após o ajuste inicial, foi realizado o *fine-tuning*, descongelando os blocos superiores da arquitetura, nomeadamente os blocos 4 e 5, responsáveis pela aprendizagem de características mais abstratas e complexas. Por outro lado, as camadas inferiores, responsáveis pela extração de características mais básicas, não foram descongeladas mantendo os seus pesos pré-treinados e a robustez das representações previamente aprendidas. Durante esta etapa, a taxa de aprendizagem inicial foi reduzida para 5e-6 de modo a garantir atualizações

mais suaves dos pesos. O *early stopping* foi configurado para interromper o treino em caso de não se verificarem melhorias na *validation loss* superiores a 5e-3 até 10 épocas consecutivas, permitindo melhorias menores e maior capacidade para ajustar as camadas superiores. As configurações da redução dinâmica da taxa de aprendizagem e do *model checkpoint* mantiveram-se inalteradas. A Figura 3.8 ilustra a diferença entre as duas fases de treino, enquanto a Tabela 3.1 apresenta todos os hiperparâmetros definidos para cada fase.

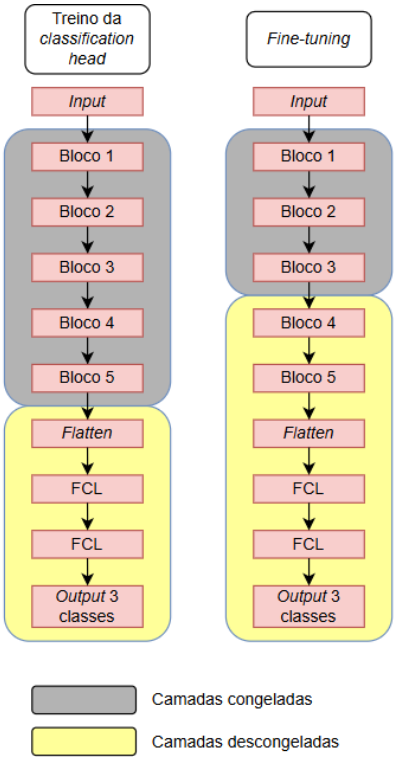

Figura 3.8: Ilustração das duas fases de treino.

Tabela 3.1: Definição de hiperparâmetros no treino da *classification head* e *fine-tuning*.

Hiperparâmetros	*Classification head*	*Fine-tuning*
Momentum (SGD)	0,9	0,9
Função de perda	CCE	CCE
LR inicial	3e-4	5e-6
Redução do LR	20%	20%
Early stopping	*patience* = 5	*patience* = 10
Critério de paragem	melhoria mínima < 0,01	melhoria mínima < 5e-3
Dropout	0,2	0,2
Camadas treináveis	*classification head*	*classification head* + blocos 4 e 5
Model checkpoint	sim	sim

3.2.6 Ajuste de Hiperparâmetros

Durante o processo de escolha do modelo de classificação, foram testadas diversas configurações e hiperparâmetros com o objetivo de melhorar o desempenho do modelo. Este processo seguiu uma abordagem iterativa baseada na tentativa e erro, onde cada alteração foi avaliada a partir dos gráficos da *training loss* e *validation loss*. A sua análise permitiu identificar a melhor configuração, ainda que não seja garantido que o modelo final represente a solução ideal. Algumas das configurações e hiperparâmetros testados incluem:

- **Arquitetura do modelo:** Foram testadas diferentes arquiteturas, nomeadamente a VGG16 e a ResNet-50, com e sem *fine-tuning*. A VGG16 apresentou resultados superiores e demonstrou uma boa capacidade de generalização mesmo antes de se ter realizado o *fine-tuning*.

- ***Classification head*:** Uma camada de *global average pooling* em vez da camada *flatten* foi uma alternativa considerada, no entanto a última apresentou um desempenho superior. O aumento e redução do número de neurónios das camadas totalmente conectadas também foram testados, mas manter esses números o mais próximo possível dos que integram a VGG16 original revelou ser a melhor abordagem, permitindo capturar relações complexas nos dados sem introduzir *overfitting*.

- ***Dropout*:** O valor de *dropout* aplicado foi ajustado para reduzir o risco de *overfitting*. Valores como 10%, 20% e 30% foram testados, tendo-se verificado que 20% apresentava o melhor compromisso entre regularização e preservação do desempenho.

- ***Fine-tuning*:** Inicialmente, foi avaliado o impacto de descongelar apenas o bloco 5, responsável pela extração das características mais específicas das imagens. No entanto, os resultados obtidos indicaram que esta abordagem não era suficiente para ajustar completamente o modelo às particularidades do *dataset*. A estratégia que apresentou os melhores resultados consistiu em descongelar os blocos 4 e 5 durante o *fine-tuning*, mantendo os blocos inferiores congelados. Por outro lado, descongelar os blocos inferiores restantes oferece ganhos pouco significativos, pois estas camadas são responsáveis por extrair

características muito genéricas, e para além disso aumenta significativamente o tempo de treino.

- **Taxa de aprendizagem:** A taxa de aprendizagem inicial para o treino da *classification head* foi definida em 3e-4, com redução dinâmica em 20% quando a *validation loss* estabilizava. Outros valores foram testados, como 1e-3 e 1e-4, no entanto o primeiro apresenta maiores oscilações e dificuldade na convergência e o segundo atrasa bastante o progresso inicial. Assim, um valor intermédio foi escolhido para equilibrar a velocidade de convergência e a estabilidade do treino. Para o *fine-tuning*, a taxa de aprendizagem inicial foi definida em 5e-6, também com a redução dinâmica de 20%. Este valor bastante reduzido é fundamental para ajustar os pesos das camadas descongeladas de forma controlada, minimizando o risco de alterar em demasia as características já aprendidas durante o pré-treino. À semelhança do treino da *classification head*, valores maiores, como 1e-5, originam flutuações elevadas na *validation loss*, enquanto valores menores, como 1e-7, tornam o progresso demasiado lento.

- ***Early stopping*:** Para o treino da *classification head* foram testados valores para a melhoria mínima da *validation loss* desde 0,01, 5e-3 e 1e-3, sendo que o valor intermédio foi o que permitiu uma convergência equilibrada antes do *fine-tuning*, com melhorias significativas e sem estabilizar demasiado o progresso. O valor do número de épocas consecutivas sem melhoria para o qual deveria ser interrompido o treino foi mantido em 5 pois é suficiente para evitar uma interrupção precoce do progresso. No *fine-tuning* também foram testados vários valores, no entanto, pelas mesmas razões descritas na primeira etapa de treino, 5e-3 mostrou ser o valor mais adequado. No entanto, para o número de épocas consecutivas sem melhoria, foram testados valores como 5, 10 e 15, chegando à conclusão de que as 10 épocas apresentavam um melhor compromisso entre dar ao modelo tempo suficiente para realizar ajustes leves e evitar o prolongamento desnecessário do treino.

3.2.7 Validação do Modelo

Inicialmente, foram definidos os melhores hiperparâmetros com base num conjunto fixo de treino e validação. Posteriormente, foi realizado um único ciclo de *cross-validation* com cinco divisões de dados diferentes, de forma a obter métricas mais robustas e menos suscetíveis de serem influenciadas por *overfitting*. No processo de *cross validation*, os

dados de treino e validação foram divididos em cinco *folds*, garantindo que cada imagem fosse usada como parte do conjunto de treino em quatro *folds* e como validação no restante.

3.3 Metodologia para Deteção

3.3.1 Aquisição das Imagens

A aquisição das imagens para a tarefa de deteção de fissuras em superfícies metálicas de componentes aeronáuticos envolveu uma abordagem diferente da utilizada para a tarefa de classificação. Neste caso, optou-se por uma metodologia mais prática e rápida, utilizando uma câmara de telemóvel equipada com uma lupa adicional fixada sobre a câmara que permitiu aumentar a ampliação. Este método foi considerado mais adequado, uma vez que as fissuras a serem analisadas não requeriam a mesma ampliação e detalhe que os mecanismos de falha observadas no MEV.

A recolha das imagens foi realizada no Centro de Formação Militar e Técnica da Força Aérea, onde muitos componentes aeronáuticos que apresentavam fissuras foram disponibilizados para análise. Estes componentes pertenciam a várias aeronaves da FAP, variando em tipo e localização estrutural, o que garantiu uma grande diversidade nos exemplos obtidos.

As imagens foram capturadas de forma a garantir a diversidade de cenários de recolha, com o objetivo de maximizar a capacidade de generalização do modelo de deteção. O processo de aquisição seguiu a seguinte metodologia:

- **Recolha de imagens com fissura:** Foram obtidas múltiplas imagens de cada fissura, com e sem frente presente, variando a posição da câmara, o ângulo e as condições de iluminação, mas mantendo a ampliação, para gerar uma diversidade significativa de exemplos.

- **Recolha de imagens sem fissura:** Em seguida, foram capturadas imagens das áreas sem fissuras nos mesmos componentes, garantindo o equilíbrio do *dataset*. Nesta etapa, também se aplicaram as condições descritas no processo de recolha de imagens com fissuras de modo a aumentar a sua variedade, tentando também obter imagens com defeitos que não correspondessem a fissuras.

A Figura 3.9 apresenta exemplos de imagens recolhidas de acordo com a abordagem descrita.

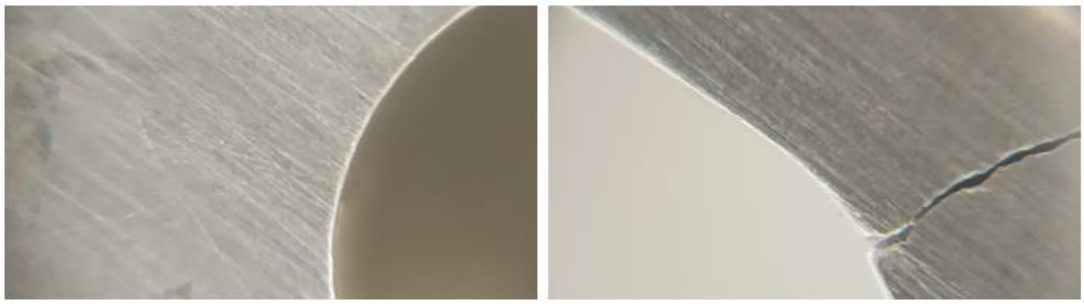

Figura 3.9. Exemplo de uma imagem recolhida sem fissura (esquerda) e com fissura (direita).

Este processo, além de ser mais prático do que a utilização do MEV, foi eficiente na captura de detalhes visuais relevantes sem comprometer a qualidade das imagens. No total foram obtidas 406 imagens, e a sua distribuição está representada na Figura 3.10:

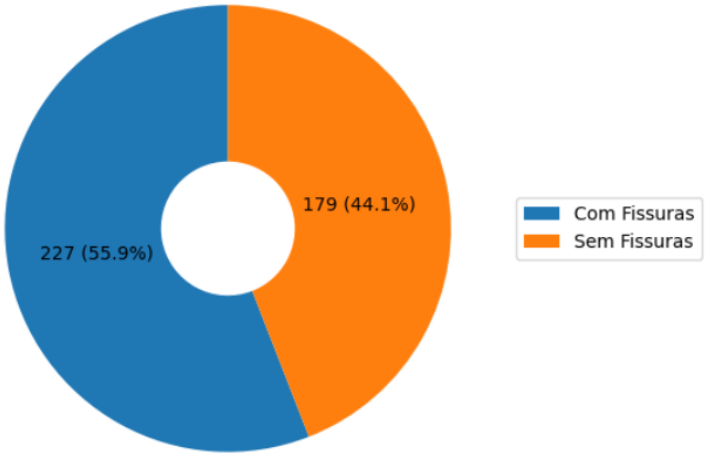

Figura 3.10. Distribuição das imagens por classe.

3.3.2 Anotação

A anotação das fissuras foi realizada utilizando a ferramenta online CVAT (*CVAT: Computer Vision Annotation Tool*, n.d.), uma plataforma amplamente utilizada para anotação de imagens em tarefas de visão computacional. O objetivo desta etapa foi gerar as caixas delimitadoras para as imagens que continham fissuras, bem como garantir que as imagens sem fissuras fossem corretamente identificadas no formato adequado. As etapas deste processo foram as seguintes:

- **Anotação das imagens com fissuras:** Foi utilizada a funcionalidade de anotação manual de *bounding boxes* no CVAT. Cada fissura visível na superfície foi contornada com uma caixa delimitadora, criando assim a área de interesse para o modelo de deteção (Figura 3.11). O formato de anotação escolhido foi o YOLO, amplamente utilizado para tarefas de deteção de objetos. Cada anotação foi posteriormente exportada como um ficheiro *.txt*.

- **Anotação das imagens sem fissuras:** Para as imagens que não apresentavam fissuras também foram criados ficheiros *.txt*, porém, nesses casos, os ficheiros estavam vazios. Este procedimento é necessário para indicar que não há objetos a serem detetados na imagem. Dessa forma, o modelo é treinado tanto para identificar fissuras, como para não detetar fissuras quando elas não estão presentes.

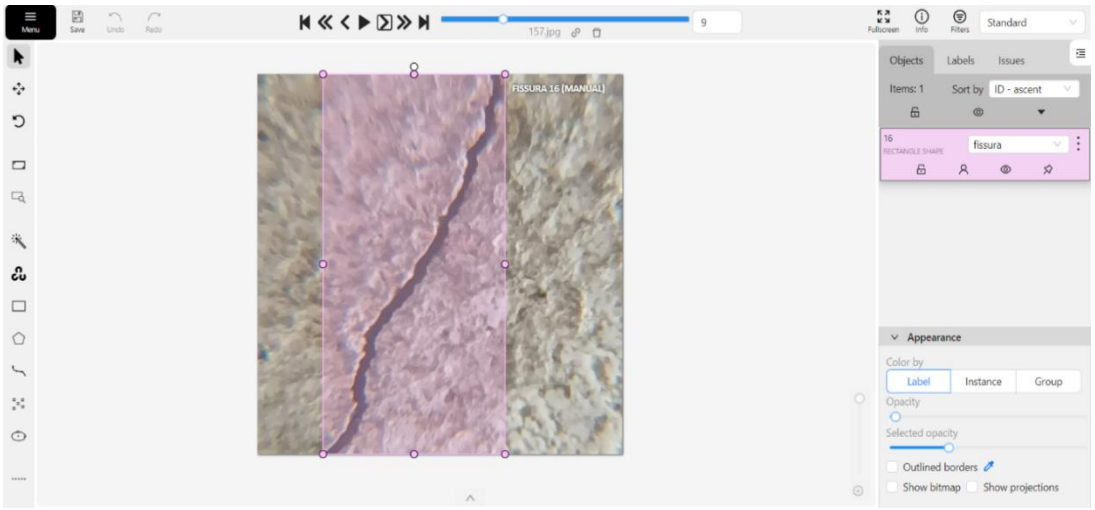

Figura 3.11. Exemplo de anotação de uma fissura em ambiente CVAT.

Cada anotação no formato YOLO é representada por um ficheiro de texto associado a uma imagem, isto é, possui o mesmo nome que a imagem correspondente. Este ficheiro contém as coordenadas de uma ou mais caixas delimitadoras que identificam a/s fissura/s na imagem, bem como a classe correspondente. Desta forma, quando não há nenhum objeto na imagem, o ficheiro encontra-se vazio. Um ficheiro de anotação neste formato contém as seguintes informações:

- ***class_id***: É o número que representa a classe do objeto, neste caso a única classe (fissura) é representado por zero.

- ***x_center***: A coordenada do centro da caixa delimitadora, normalizada em relação à largura da imagem, ou seja, um valor entre zero e um.

- ***y_center***: A coordenada do centro da caixa delimitadora, normalizada em relação à altura da imagem.

- ***width***: A largura da caixa delimitadora, normalizada em relação à largura da imagem.

- ***height***: A altura da caixa delimitadora, normalizada em relação à altura da imagem.

A Figura 3.12 mostra como são os ficheiros de texto utilizados para representar as anotações no formato YOLO.

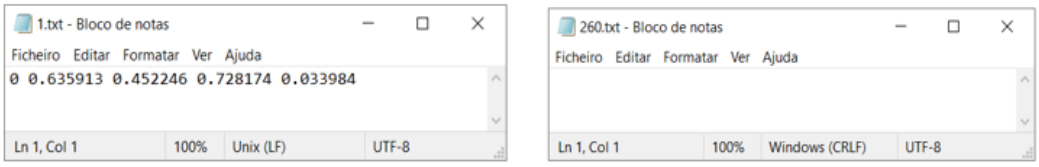

Figura 3.12. Exemplo de anotação em formato YOLO numa imagem com uma fissura (esquerda) e numa imagem sem nenhuma fissura (direita).

3.3.3 Pré-processamento das Imagens

Após a aquisição e anotação das imagens, segue-se o pré-processamento das mesmas. Esta etapa é crucial na preparação dos dados para o treino do modelo de deteção e inclui os seguintes passos:

- **Redimensionamento:** As imagens originais possuíam dimensões relativamente grandes, como 2250x4000 e 4000x2250 píxeis. Para otimizar o processo de treino e reduzir a utilização de memória, foi feito o redimensionamento de todas as imagens para 1024x1024 píxeis. Este tamanho foi escolhido por ser suficientemente grande para capturar os detalhes das várias imagens, sem comprometer o desempenho computacional do modelo. Independentemente deste procedimento, durante o treino, o *Detectron 2* realiza automaticamente um redimensionamento dinâmico adicional, com as imagens sendo ajustadas entre 640 e 800 píxeis para o lado mais pequeno e garantindo que o tamanho do lado maior não ultrapassa os 1333 píxeis.

- **Conversão de formatos:** As imagens originais encontram-se no formato RGB, no entanto foram processadas automaticamente no formato BGR, que é o formato padrão do *OpenCV* para leitura de imagens.

- **Normalização:** O *Detectron 2* realiza a normalização dos valores dos píxeis internamente, garantindo que esses valores se situem numa escala entre 0 e 1, em vez de 0 a 255. Isso permite que o modelo apresente maior estabilidade durante o treino e melhore o seu desempenho, pois as redes neuronais, no geral, são bastante sensíveis a escalas muito grandes.

- ***Data augmentation*:** De modo a aumentar o número de imagens sem necessidade de novas recolhas, foi utilizada a técnica de *data augmentation*. Apesar das múltiplas ações de manipulação de imagens que se podem realizar com esta técnica, o problema de deteção de fissuras apresenta certas características que limitam esta abordagem. Primeiramente, mais de metade do *dataset* é constituído por imagens com fissuras, que é o objeto a ser detetado, e consequentemente não se podem realizar ações de manipulação que introduzam ruído ou distorçam a forma das fissuras. Em segundo lugar, é importante destacar que o processo de aumento do número de imagens foi realizado depois de terem sido efetuadas as devidas anotações, de modo a diminuir o tempo despendido na anotação de todas as imagens do *dataset*. No entanto, ao aplicar manipulações mais complexas, como rotações ou deformações, existe o risco de a *bounding box* se deslocar de forma errada, mesmo com o uso de bibliotecas como o *Albumentations*, que automaticamente ajusta as *bounding boxes* após as transformações. Dados esses desafios, optou-se por realizar ações de *data augmentation offline* simples e seguras como a rotação de 90 graus e o ajuste de brilho e contraste, que não comprometem nem a forma da fissura nem a localização das *bounding boxes* (Figura 3.13).

  Para além das transformações *offline* aplicadas, o *Detectron 2* aplica automaticamente técnicas de aumento de dados *online*. Especificamente, a técnica de *RandomFlip* foi utilizada, realizando uma inversão horizontal aleatória das imagens.

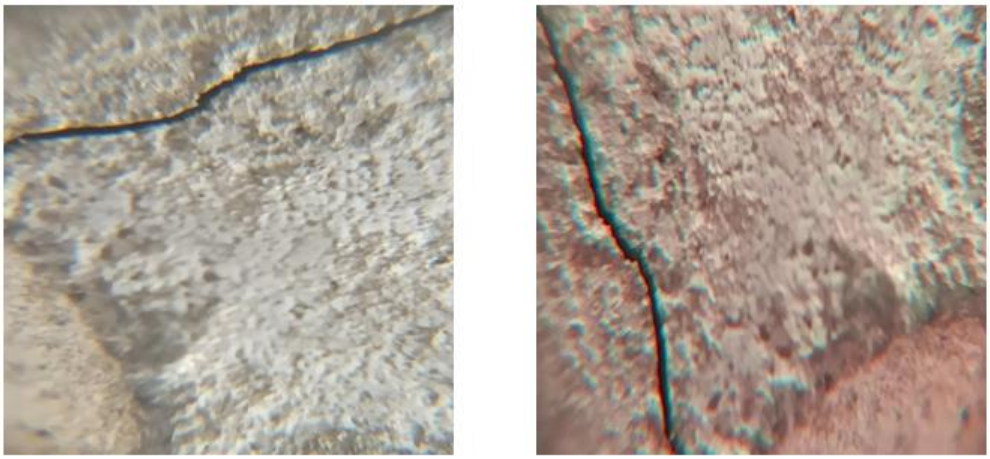

Figura 3.13. Exemplo de utilização de rotação de 90 graus e ajuste de brilho e contraste para formar uma nova imagem (direita) a partir da original (esquerda).

Assim, foi possível aumentar o *dataset* para 784 imagens, distribuídas da seguinte forma:

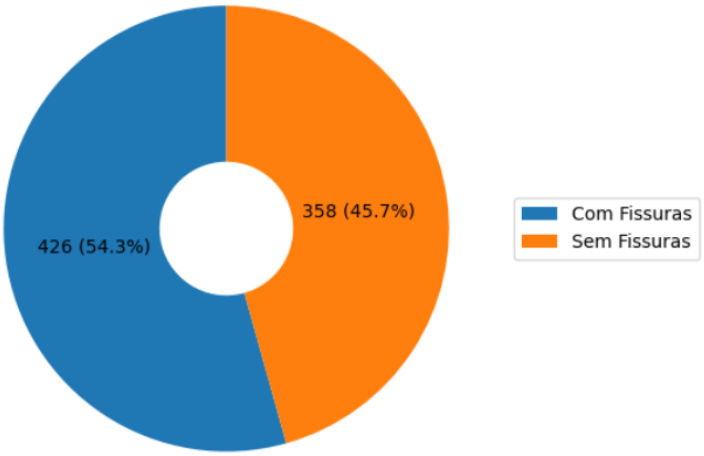

Figura 3.14. Distribuição do *dataset* final, após *data augmentation*.

Divisão do *Dataset*: Após todo este processo, o *dataset* foi dividido em três subconjuntos, treino (70%), validação (20%) e teste (10%), de forma aleatória e mantendo a proporção das classes.

3.3.4 Organização das Imagens

Após a divisão do *dataset* foi necessário organizar o mesmo através de uma estrutura de pastas e subpastas de forma a distinguir claramente os conjuntos de treino, validação e teste, bem como as anotações correspondentes de cada imagem. A Figura 3.15 demonstra esquematicamente como foi feita essa organização. Esta abordagem apresenta vantagens

como a facilidade de visualização da distribuição das imagens, compatibilidade com os *frameworks* utilizados (*PyTorch*) e a reprodutibilidade dos resultados, no entanto pode apresentar um problema ao nível do armazenamento se a quantidade de imagens for relativamente grande.

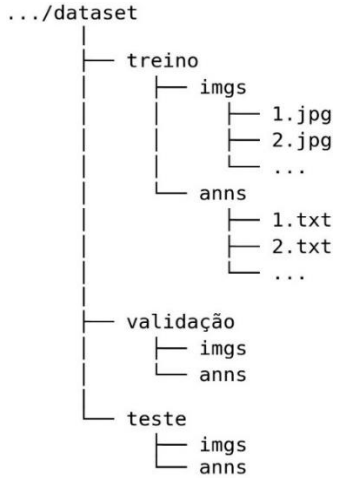

Figura 3.15. Organização das imagens e anotações.

3.3.5 Arquitetura do Modelo

O modelo de deteção proposto foi ajustado com recurso à biblioteca *Detectron 2*, com uma arquitetura baseada no modelo *Faster R-CNN* com uma CNN *ResNet-101* e FPN como *backbone*. Esta escolha é justificada com base em trabalhos semelhantes referidos na literatura, bem como nos parâmetros de desempenho obtidos pela própria arquitetura no modelo *baseline* do *Detectron 2*.

Em relação ao que é referido na literatura, esta escolha baseia-se principalmente num dos trabalhos apresentados na secção 2.6.2, onde o desempenho do *Faster R-CNN* foi avaliado com diferentes combinações, como *ResNet-101* e FPN, numa tarefa de deteção de FCIS (S. Y. Wang & Guo, 2021). O modelo *Faster R-CNN* com *ResNet-101* destacou-se pelo desempenho superior em termos de precisão e *F1-score*, o que o torna uma escolha preferencial para tarefas que exijam elevada certeza nas previsões. Por outro lado, o modelo com FPN demonstrou vantagem em termos de *recall*, sendo mais adequado para problemas onde a deteção de um maior número de objetos é prioritária. Assim, a escolha da arquitetura proposta tem como objetivo combinar o excelente desempenho dos dois modelos, que apresentaram resultados complementares no estudo comparativo. Para além disso, foi consultada a tabela de desempenho dos vários modelos *baseline* do *Detectron 2* (Facebook AI Research, n.d.), que utilizam o modelo *Faster R-*

CNN como detetor, combinado com diferentes CNNs como *backbones*. De entre as diversas configurações, a arquitetura escolhida destaca-se por alcançar uma das melhores pontuações em *Average Precision* (42), enquanto mantém um tempo de inferência por imagem (0,051 s/imagem) e uso de memória (4.1 GB) moderados. Esta combinação de parâmetros é fundamental, uma vez que se pretende manter uma boa precisão na previsão, sem comprometer o desempenho em termos de tempo e custo de memória.

Para finalizar foi também adotada a técnica de *transfer learning* para aproveitar os pesos pré-treinados no *dataset COCO*, amplamente utilizado para problemas de deteção de objetos.

3.3.6 Configurações de Treino e Hiperparâmetros

Na configuração do modelo, foram feitos ajustes para melhorar o seu desempenho no problema em questão. Por defeito, o *Detectron 2* descarta as imagens sem anotações, no entanto, é necessário que o modelo seja capaz de prever fissuras quando estas existem, mas também não fazer qualquer previsão quando estas não estão presentes. Desta forma, o modelo foi configurado para incluir as imagens sem anotações, permitindo que as características de uma zona sem fissuras também fossem aprendidas.

Em relação às *anchor boxes*, estas foram definidas com tamanhos de [128, 256, 512] píxeis e proporções de [0.5, 1.0, 2.0], de modo a detetar fissuras predominantemente grandes, que representam a maioria dos casos no *dataset* (99%). Esta configuração foi ajustada para maximizar a deteção de fissuras de tamanho considerado *large* pelo *Detectron 2*. Por outro lado, a FPN tem um papel crucial em garantir que o modelo também faça a deteção de fissuras menores, embora menos frequentes, oferecendo um equilíbrio entre precisão na deteção de fissuras grandes e sensibilidade para os pequenos detalhes. A taxa de aprendizagem foi definida em 1e-4 com um *scheduler* de modo a reduzir o seu valor em 10% para 60% e 80% do número total de iterações, definidas em 12000, além de um fator de *warmup* de 1e-3 nas primeiras 100 iterações para suavizar o impacto inicial do treino. Para facilitar a monitorização do progresso de treino e permitir a recuperação do mesmo em caso de interrupções, o modelo foi configurado para guardar *checkpoints* a cada 500 iterações.

A escolha de lotes com apenas 2 imagens teve em conta o tamanho relativamente grande das mesmas e os limites de memória computacional impostos. Em termos de

regularização, o *Detectron 2* aplica automaticamente uma penalização L2 (*weight decay*), o que contribui para reduzir o risco de *overfitting*. A função de perda e o otimizador também seguem os valores predefinidos do *Detectron 2*. A função de perda é uma combinação da perda de classificação (CCE) e perda da localização (*Smooth* L1) das caixas delimitadoras e o otimizador utilizado é o SGD, que é bastante eficaz para treinar CNNs profundas. A Tabela 3.2 apresenta todos os hiperparâmetros definidos para o treino.

Tabela 3.2: Definição de hiperparâmetros no treino do modelo de deteção.

Hiperparâmetros	**Valor**
Tamanho das *anchor boxes*	[128, 256, 512] píxeis
Proporções das *anchor boxes*	[0.5, 1.0, 2.0]
LR inicial	1e-4
Número de iterações	12000
Scheduler	Redução de 10% em 60% e 80% das iterações
Fator de *warmup*	1e-3
Número de iterações de *warmup*	100
Frequência de *checkpoints*	500 iterações
Regularização	L2
Função de perda	CCE + *Smooth* L1
Otimizador	SGD

3.3.7 Ajuste de Hiperparâmetros

No desenvolvimento de um modelo de ML, o processo de melhoria e ajuste de hiperparâmetros é essencial para garantir que o modelo final não só seja capaz de apresentar um bom desempenho nos dados de treino, mas que também seja capaz de generalizar para novos dados. Alcançar o melhor modelo requer um processo iterativo, onde os resultados de cada versão do modelo são avaliados, levando a ajustes progressivos nos hiperparâmetros e na estrutura da rede. Ainda assim, nada garante que o modelo final seja o melhor absoluto.

Durante o ajuste do modelo de deteção de fissuras, foi realizada uma série de intervenções de melhoria, através de um processo de tentativa e erro. As principais técnicas de monitorização do desempenho do modelo incluíram a análise dos gráficos da *total loss*, além de métricas de avaliação comuns em tarefas de deteção de objetos, como a precisão, *recall*, *F1-score* e exatidão. A partir da análise destes elementos foi possível

perceber o comportamento do modelo de modo que fossem realizadas as alterações necessárias. Os principais hiperparâmetros ajustados incluem:

- **Arquitetura do modelo:** Foram testadas como *backbone* a *ResNet-50* e *ResNet-101* para perceber qual o impacto da complexidade da arquitetura no desempenho do modelo. A *ResNet-101*, sendo mais complexa, demonstrou uma melhor capacidade de generalização sem introduzir *overfitting*.

- **Número de iterações:** O número total de iterações foi testado com base na observação dos gráficos da *training loss* e *validation loss*, bem como nas métricas de avaliação já referidas.

- **Taxa de aprendizagem:** Foram testadas duas abordagens diferentes na definição da taxa de aprendizagem, a primeira utilizando um valor fixo e a segunda utilizando um valor variável que permita estabilizar a atualização dos pesos na fase final do treino. A taxa de aprendizagem fixa acaba por, inevitavelmente, limitar o progresso do treino e por isso optou-se pela taxa variável.

- **Tamanho das *anchor boxes*:** Vários tamanhos para as *anchor boxes* foram testados, bem como várias razões de aspeto para acomodar diferentes tamanhos e formas de fissuras. A seleção de *anchor boxes* personalizadas aumenta claramente o desempenho do modelo em comparação com aquelas que são definidas por defeito.

- **Congelamento de camadas:** Diferentes técnicas de *transfer learning* foram avaliadas nomeadamente *transfer learning* padrão do *Detectron 2*, onde é carregado o modelo pré-treinado, mas todas as camadas podem ser ajustadas durante o treino, congelamento da primeira camada convolucional e congelamento da segunda camada (residual). Conclui-se que o processo de *transfer learning* padrão era o que permitia obter os melhores resultados, ainda que sem introduzir uma grande vantagem em relação às restantes abordagens.

3.3.8 Validação do Modelo

Inicialmente, foram definidos os melhores hiperparâmetros com base num conjunto fixo de treino e validação. Posteriormente, foi realizado um único ciclo de *cross-validation*

com cinco divisões de dados diferentes, de forma a obter métricas mais robustas e menos suscetíveis de serem influenciadas por *overfitting*. No processo de *cross validation*, todas as imagens foram divididas em cinco *folds*, garantindo que cada imagem fosse usada como parte do conjunto de treino em quatro *folds* e como validação no restante. Para cada *fold*, o modelo foi treinado com 12000 iterações, com *checkpoints* guardados a cada 500 iterações. Esta abordagem foi adotada para assegurar que fosse sempre possível selecionar o modelo que apresentasse o melhor desempenho nos dados de validação do respetivo *fold*, seja ao fim do total de iterações ou num *checkpoint* intermédio.

Capítulo 4

4 Resultados

Neste capítulo são apresentados e analisados os resultados obtidos pelos modelos apicados, tanto para a classificação de mecanismos de falha como para a deteção de fissuras. Inicialmente, são discutidos os resultados obtidos após realização de *cross validation*, incluindo os gráficos da função de perda, tempo total de treino e métricas de desempenho calculadas de forma global. Posteriormente, são apresentados os resultados nos dados de teste, com destaque não só para as métricas globais como para as métricas individuais de cada classe, permitindo uma análise mais detalhada da capacidade de generalização dos modelos. Por fim, alguns exemplos de previsões realizadas nos dados de teste são apresentados com o objetivo de perceber melhor o comportamento do modelo, auxiliando na identificação de possíveis limitações e aspetos a melhorar.

4.1 Resultados do Modelo de Classificação

4.1.1 Desempenho Após Cross Validation

A Tabela 4.1 apresenta os resultados obtidos pelo modelo de classificação nas métricas calculadas, tanto para cada divisão como para a média das cinco.

Tabela 4.1: Resultados do modelo de classificação após *cross validation*.

Métricas (%)	Divisão 1	Divisão 2	Divisão 3	Divisão 4	Divisão 5	Média (%)
Precisão	93,06	91,69	92,39	93,68	90,17	92,2
Recall	93,04	91,55	92,26	93,58	90,21	92,1
F1-score	93,03	91,49	92,21	93,60	90,10	92,1
Exatidão	93,06	91,61	92,31	93,71	90,21	92,2

Pela observação dos resultados obtidos é possível notar a elevada consistência de todas as métricas nas cinco divisões diferentes, com apenas uma variação máxima de 3,51% registada para a precisão. Este comportamento indica que o modelo não é dependente de uma divisão específica dos dados, demostrando a sua capacidade de generalização.

As métricas médias obtidas, com valores bastante elevados, refletem a sua capacidade de cumprir o objetivo proposto, isto é, classificar de forma confiável os diferentes

mecanismos de falha. A precisão média de 92,2% indica que a maioria das classificações realizadas pelo modelo são corretas, enquanto o *recall* médio de 92,1% demonstra que o modelo é capaz de identificar a grande maioria dos casos existentes minimizando o número de falsas previsões. Por fim, a exatidão média de 92,2% sugere que, globalmente, o modelo apresenta uma elevada taxa de acerto, demostrando que este tem capacidade para fornecer classificações confiáveis.

Os gráficos da evolução da *training loss* e *validation loss* para cada divisão dos dados durante a *cross validation* são apresentados na Figura 4.1. A linha tracejada vertical faz a divisão entre as duas fases de treino distintas que já foram explicadas anteriormente. Na fase inicial, correspondente ao treino da *classification head*, verifica-se uma redução acentuada na função de perda, tanto para os dados de treino como para os de validação, o que é um comportamento esperado, uma vez que o modelo está a ajustar os pesos para aprender as características gerais das imagens. O facto de a *validation loss* se manter relativamente próxima da *training loss* indica que o modelo está a generalizar bem nesta etapa inicial.

Com a transição para o *fine-tuning*, observa-se uma redução mais gradual, mas sempre contínua na *training loss* até ao final de todo o processo. Este comportamento é característico desta fase, onde as camadas superiores são descongeladas e ajustadas para melhorar a capacidade do modelo de capturar padrões mais específicos das imagens. A *validation loss* também apresenta uma tendência de descida durante o *fine-tuning*, embora com valores sempre ligeiramente superiores aos registados na *training loss*. Este comportamento é comum e normal, desde que ambas as curvas continuem a descida de forma "paralela", tal como se observa em cada gráfico, indicando que as melhorias obtidas no treino estão a ser refletidas também nos dados de validação, não estando a ocorrer *overfitting*.

O comportamento das curvas para cada divisão é bastante semelhante, à exceção da divisão 2 que apresenta os valores da *training loss* e *validation loss* ligeiramente mais elevados, cerca de 0,2 e 0,3, respetivamente. Apesar deste comportamento o desempenho do modelo para a divisão 2 não foi afetado.

É importante destacar que o *early stopping* foi sempre ativado no final do *fine-tuning* de modo a evitar *overfitting*. Apesar das melhorias aparentes na *validation loss* nas últimas épocas, estas foram inferiores ao valor limite definido, levando à interrupção do

treino. Mais especificamente, pode afirmar-se que nas últimas 10 iterações a melhoria na *validation loss* foi inferior a 5e-3.

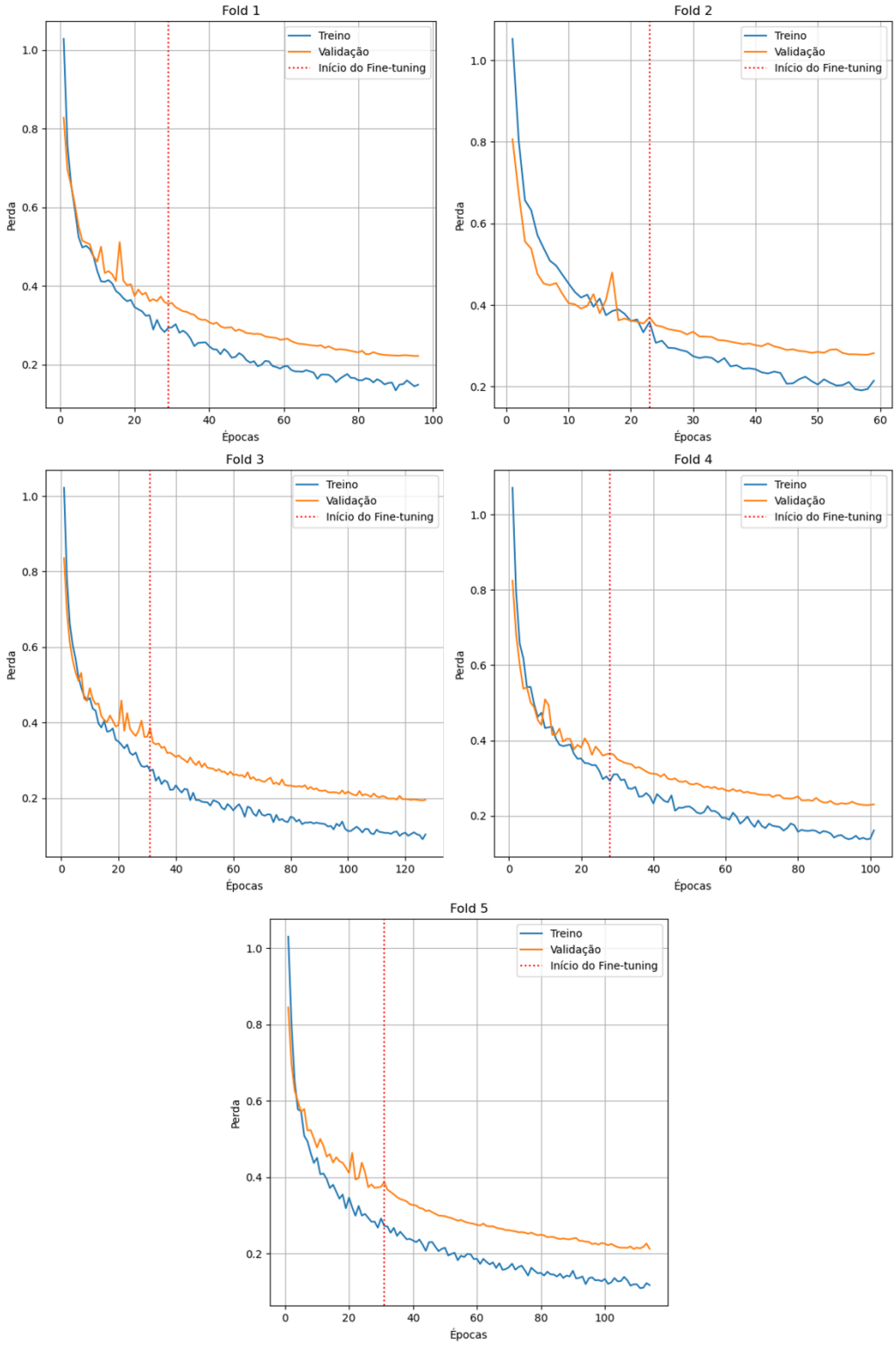

Figura 4.1. Gráficos da *training loss* e *validation loss* para cada uma das divisões de dados, com destaque para a transição entre o treino da *classification head* e o *fine-tuning*.

Por fim, a Tabela 4.2 mostra o número total de épocas e o tempo total de treino para cada divisão, com um tempo médio de execução por época de 161 segundos, evidenciando a elevada complexidade computacional envolvida.

Tabela 4.2: Número total de épocas e tempo total de treino para cada divisão de dados.

Divisão	Total de Épocas	Tempo de Treino (h)
1	96	4,25
2	59	2,49
3	127	5,89
4	101	4,61
5	114	5,16

4.1.2 Desempenho nos Dados de Teste

Para avaliar a capacidade de generalização do modelo de classificação, foi reservado um conjunto de imagens de teste que corresponde a 10% do total de imagens do *dataset*, de forma a garantir que as mesmas não fossem utilizadas durante o treino do modelo. Com um total de 80 imagens e mantendo a proporção das três classes, o conjunto de teste permitiu avaliar o desempenho real do modelo.

Após a *cross validation*, as imagens utilizadas para treino e validação foram reunidas para obter um novo conjunto de treino, correspondente a 90% do *dataset* total. Durante este processo, foi também incluído um subconjunto de validação menor, correspondente a 10% dos 90% utilizados para treino, com o objetivo de monitorizar o desempenho durante o treino final e evitar *overfitting*, mas sempre com o objetivo de maximizar a quantidade de imagens utilizadas no treino. Esta abordagem é fundamentada em práticas descritas na literatura (Berrar, 2018; Wilimitis & Walsh, 2023), que sugerem combinar os dados de treino e validação após *cross validation* para treinar um modelo final mais robusto e representativo para a avaliação no conjunto de teste, garantindo que este permaneça completamente independente para medir a capacidade de generalização do modelo. A Tabela 4.3 apresenta os resultados obtidos pelo modelo final nos dados de teste, individualmente para cada classe e de forma global.

De um modo geral é possível afirmar que a classe em que o modelo apresentou pior desempenho foi a falha por sobrecarga, muito provavelmente associado à elevada complexidade e variabilidade das imagens correspondentes a essa classe, tal como será comprovado na visualização das previsões feitas pelo modelo mais à frente nesta secção. Apesar da precisão elevada de 91,30%, o *recall* de 84,00% indica uma maior dificuldade

em classificar todas as imagens para essa classe em comparação às restantes. O *F1-score* de 87,50% reflete exatamente esse comportamento sugerindo que, apesar da elevada confiabilidade nas previsões desta classe, ainda há margem para melhorias.

Tabela 4.3: Resultados do modelo de classificação nos dados de teste.

Classe	Precisão (%)	*Recall* (%)	*F1-score* (%)	Exatidão (%)
Sobrecarga	91,30	84,00	87,50	**92,5**
Fadiga	93,33	100,00	96,55	
Corrosão	92,59	92,59	92,59	
Média	**92,4**	**92,2**	**92,2**	

Contrariamente à falha por sobrecarga, a falha por fadiga foi a classe onde o modelo apresentou o melhor desempenho em todas as métricas, com a maior precisão de 93,33% e o *recall* de 100% resultando num elevado *F1-score* de 96,55%. Estes valores demonstram a grande confiabilidade das previsões do modelo, bem como a sua capacidade para classificar corretamente todas as imagens desta classe. As estrias de fadiga, padrão característico das imagens desta classe e que apresentam baixa variabilidade nas imagens do *dataset*, poderão ser a causa do excelente desempenho do modelo.

Relativamente à classe de corrosão, os valores foram bastante sólidos, apresentando um desempenho intermédio em relação às restantes classes, com precisão, *recall* e *F1-score* de 92,59%. Estes resultados demonstram um bom equilíbrio entre a capacidade de o modelo prever corretamente esta classe e evitar falsos positivos.

Por fim, a média final de todas as métricas é igualmente elevada com uma exatidão global de 92,5%, superior em 0,3% à observada após *cross validation*. Estes resultados refletem a capacidade do modelo em cumprir o seu objetivo principal, ou seja, realizar previsões confiáveis para novas imagens de diferentes mecanismos de falha.

Para uma análise ainda mais detalhada do desempenho do modelo, foi construída uma *confusion matrix*, como é apresentada na Figura 4.2. A sua análise em conjunto com as restantes métricas permite uma avaliação completa do desempenho do modelo, destacando os seus pontos fortes e aspetos a melhorar.

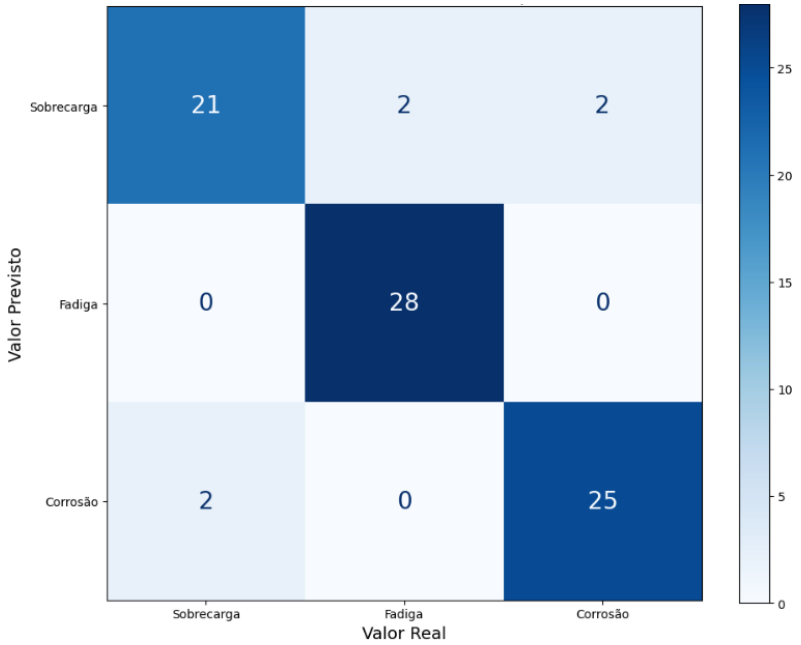

Figura 4.2: *Confusion matrix*.

Para além da análise de todas as métricas, foram também geradas imagens que demonstram visualmente todo o tipo de classificações realizadas pelo modelo. A visualização destas imagens tem como objetivo compreender como se comporta o modelo em diferentes situações, os seus pontos fortes e os motivos para algumas classificações incorretas.

A Figura 4.3 apresenta exemplos de classificações corretas realizadas pelo modelo para a classe de falha por sobrecarga. O facto desta classe apresentar o pior desempenho de entre todas, tal como se constata na Tabela 4.2, deve-se à complexidade e grande variedade dos padrões presentes nas imagens. Os casos a) e b), por exemplo, apresentam *dimples* muito característicos da falha por sobrecarga, mas de tamanhos diferentes. Tendo em conta que todas as imagens foram obtidas com a mesma ampliação, o caso a) mostra uma superfície de fratura constituída por muitos *dimples* e pequenos enquanto o caso b) mostra um superfície com menos *dimples,* mas de maior tamanho. Por outro lado, os casos c) e d) representam outros padrões que podem ser encontrados neste tipo de falhas, como os grãos bem definidos, por exemplo.

Relativamente à classe de falha por fadiga, a Figura 4.4 mostra alguns exemplos de classificações corretas realizadas pelo modelo. Os exemplos apresentados e o *recall* de 100% obtido para esta classe (Tabela 4.2) demonstram a capacidade do modelo em prever corretamente todas as imagens de teste para esta classe. A falha por fadiga não

apresenta uma grande variabilidade nos padrões das suas imagens sendo sempre muito caracterizadas pelas estrias, tal como se observa nos quatro exemplos da Figura 4.4.

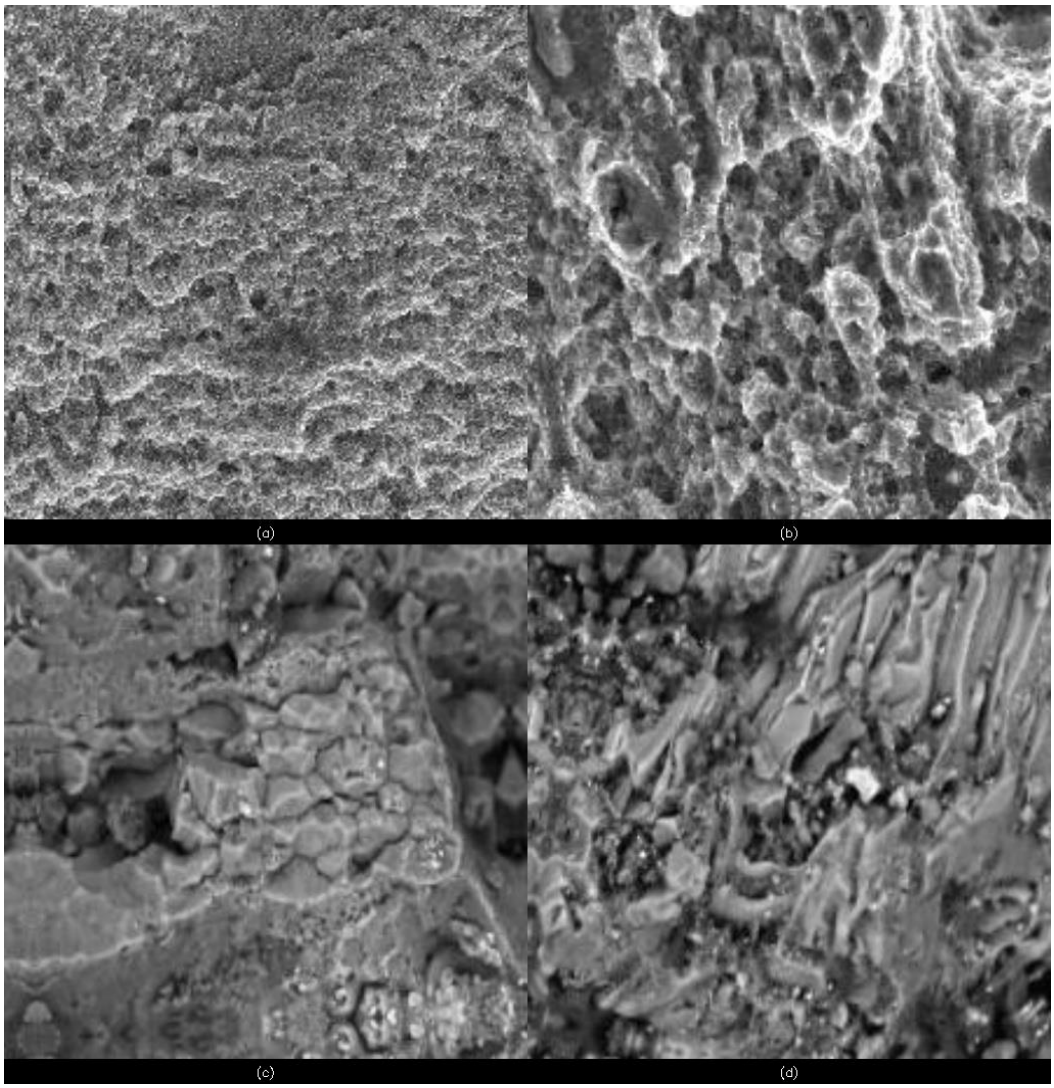

Figura 4.3. Exemplos de previsões corretas para a classe sobrecarga.

Finalmente, a Figura 4.5 apresenta exemplos de classificações corretamente realizadas pelo modelo para a classe corrosão. Esta classe apresenta alguma variabilidade nas suas imagens, como por exemplo os casos a), b) e c) que evidenciam a tridimensionalidade da superfície de fratura e o caso d) que apresenta uma topologia mais bidimensional. Em todo o caso, o modelo foi capaz de prever corretamente a maioria das imagens desta classe, apresentando um desempenho intermédio relativamente às outras duas classes.

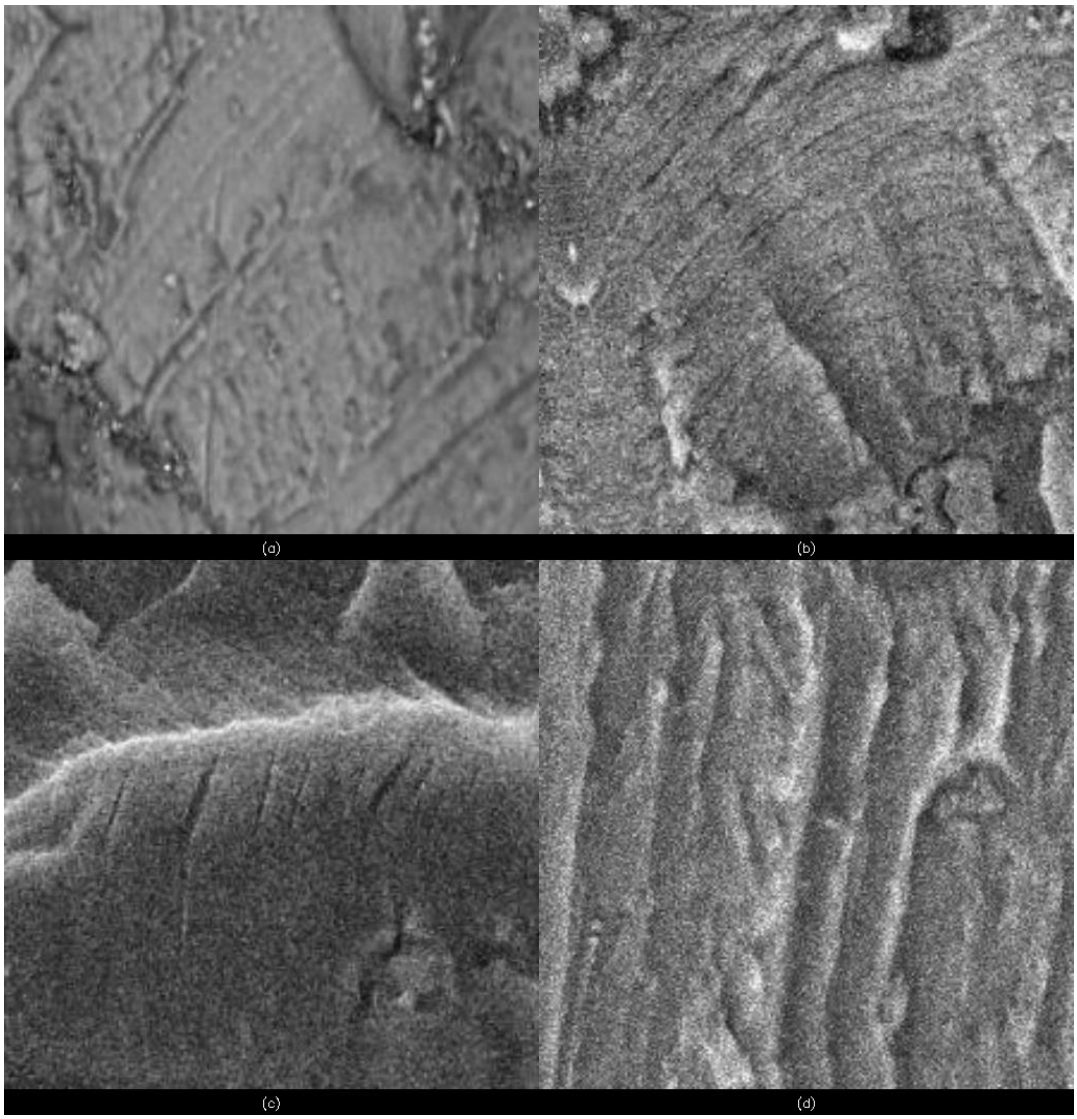

Figura 4.4. Exemplos de previsões corretas para a classe fadiga.

Em relação às previsões realizadas incorretamente pelo modelo (Figura 4.6), apenas são apresentados exemplos em que este não conseguiu classificar corretamente imagens pertencentes à classe corrosão e sobrecarga. Nos casos a) e b) o modelo fez a previsão de sobrecarga onde na realidade se apresentam imagens de corrosão. Estes dois casos foram os únicos falsos negativos registados e ambos indicam que possa existir algum tipo de semelhança entre os padrões presentes na corrosão com os de sobrecarga. Os casos c) e d) representam dois falsos negativos para a classe de sobrecarga, no entanto para o primeiro o modelo classificou a imagem como fadiga e para o segundo como corrosão. Estes dois exemplos juntamente com os valores apresentados na *confusion matrix* (Figura 4.2), confirmam a complexidade associada aos padrões representativos desta classe, de tal forma que foi confundida com as restantes, duas vezes para cada uma. Com base nas observações das várias previsões do modelo é possível perceber a razão do seu

desempenho ter sido melhor para a classe de fadiga, justificar o pior desempenho para a sobrecarga e compreender o desempenho intermédio para a corrosão.

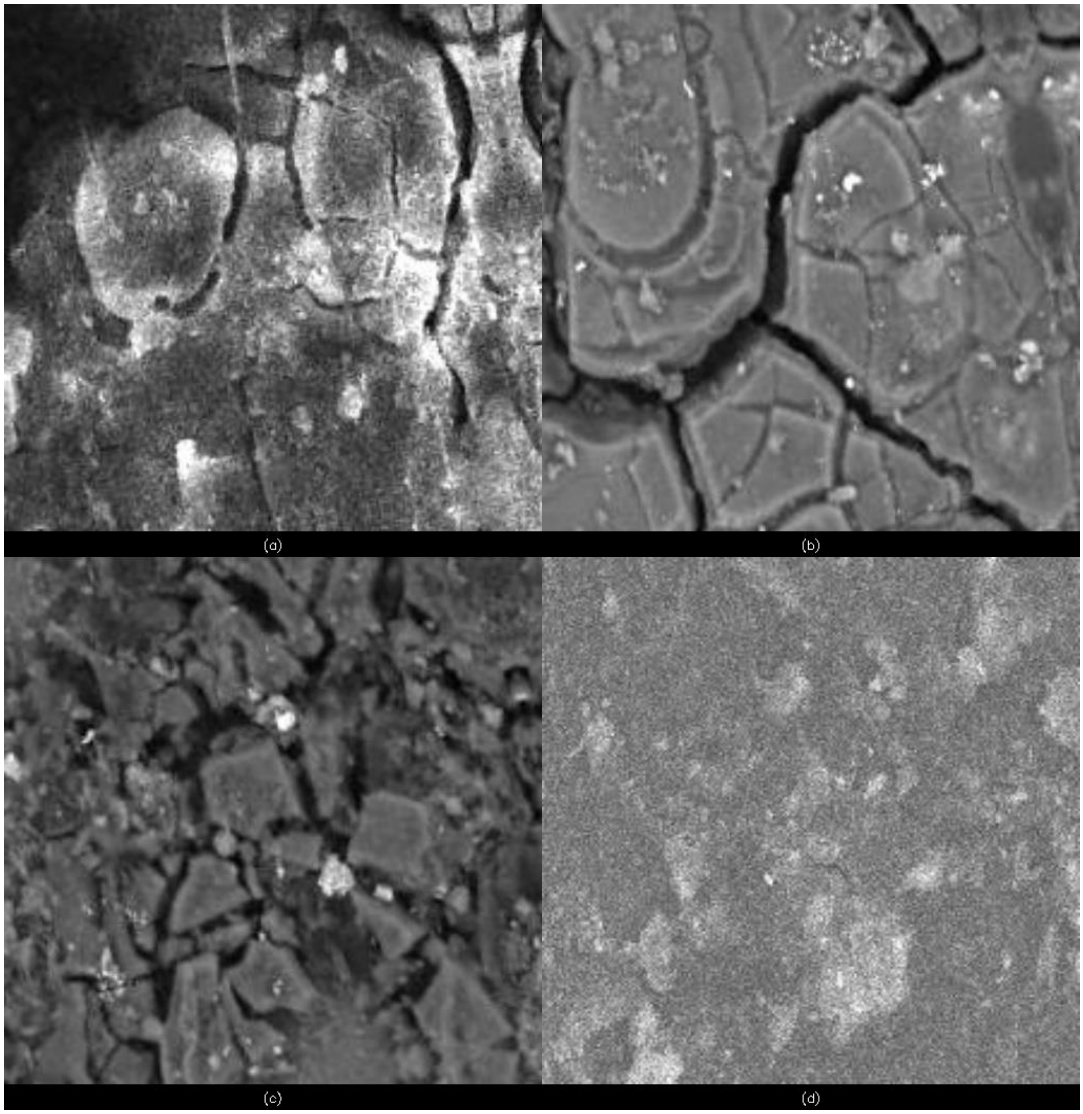

Figura 4.5. Exemplos de previsões corretas para a classe corrosão.

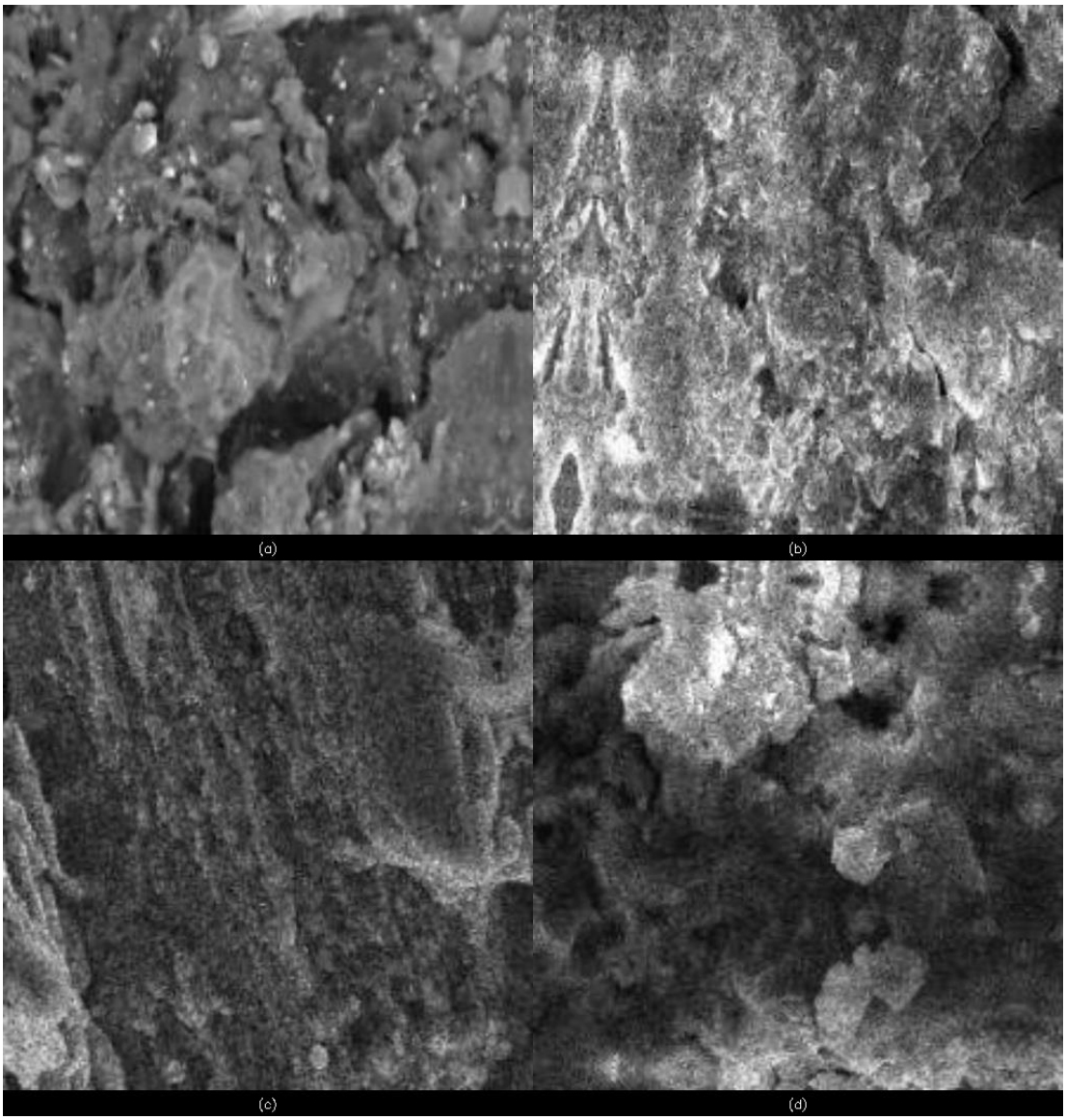

Figura 4.6. Exemplos de previsões incorretas. Casos a) e b) com classe real corrosão e previsão sobrecarga. Caso c) com classe real sobrecarga e previsão fadiga. Caso d) com classe real sobrecarga e previsão corrosão.

4.2 Resultados do Modelo de Deteção

4.2.1 Desempenho Após Cross Validation

O desempenho do modelo de deteção de fissuras foi avaliado com base nos conceitos TP, FP, TN e FN, definidos da seguinte forma:

- **TP:** Previsão correta de fissura.

- **FP:** Previsão incorreta de fissura.

- **TN:** Previsão correta de ausência de fissura em imagem sem anotação, isto é, não há qualquer previsão de caixa delimitadora feita pelo modelo.

- **FN:** Previsão incorreta de ausência de fissura, ou seja, não há qualquer previsão de caixa delimitadora numa imagem que possui fissura.

A lógica utilizada para determinar cada um destes casos baseia-se em critérios cuidadosamente definidos com base nas necessidades do problema em questão. Em primeiro lugar, as previsões geradas pelo modelo são filtradas através de um valor de *threshold* definido como 0,6, o que garante que apenas previsões com uma confiança acima de 60% são consideradas. O ajuste deste valor deve ser feito de forma equilibrada, pois um valor de *threshold* muito alto favorece o aparecimento de casos FN e um valor muito baixo favorece o aparecimento de casos FP.

O principal critério utilizado para classificar as previsões acima do valor de confiança é a métrica IoU. Sempre que o valor de IoU entre a previsão e a anotação for superior a 0,75, a previsão é considerada TP. O valor limite definido para este critério foi escolhido de modo a assegurar que as previsões corretas possuam sempre uma alta correspondência com as anotações. Nos casos em que o valor de IoU está no intervalo entre 0,5 e 0,75, aplica-se um critério adicional baseado na percentagem da área da anotação que está coberta pela área da previsão. Se a área da previsão cobrir 100% da área da anotação, esta é considerada TP. Por outro lado, se a área da previsão não englobar totalmente a área da anotação, é classificada como FP. Por fim, as previsões com IoU inferior a 0,5 são imediatamente consideradas FP. Esta lógica de decisão foi implementada com o principal objetivo de mitigar a penalização de previsões com IoU moderado, mas que conseguem abranger a totalidade da fissura. A Figura 4.7 ilustra diferentes cenários com os respetivos valores de IoU.

Relativamente aos casos FN, estes são todos aqueles em que o modelo não foi capaz de realizar qualquer previsão em imagens com fissuras, ou então a confiança da previsão foi abaixo do valor de threshold estipulado. Para as imagens sem anotações o processo é mais simples: se nenhuma previsão for feita, o resultado é considerado como TN, mas se houver previsões, estas são tratadas como FP.

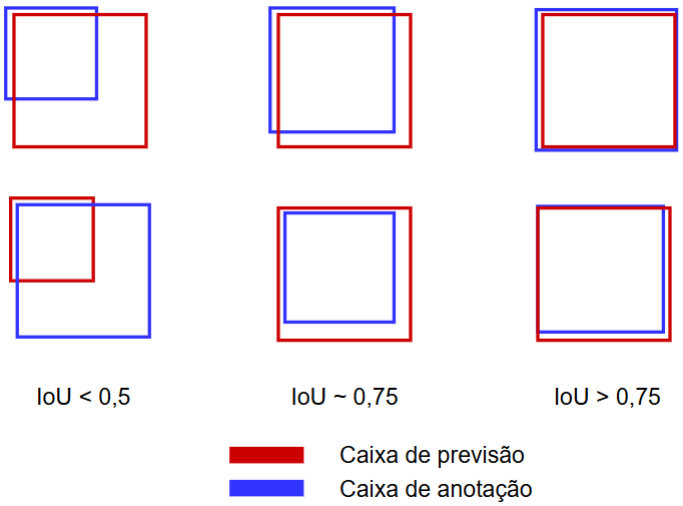

Figura 4.7: Diferentes casos de sobreposição das caixas delimitadores e respetivos valores de IoU.

Com base no que foi definido, foram calculadas diversas métricas, amplamente utilizadas em tarefas de deteção (S. Y. Wang & Guo, 2021), para avaliar o desempenho do modelo. A Tabela 4.4 apresenta os resultados obtidos pelo modelo tanto para cada divisão como para a média das cinco após a realização de *cross validation*.

Tabela 4.4: Resultados do modelo de deteção após *cross validation*.

Métricas (%)	Divisão 1	Divisão 2	Divisão 3	Divisão 4	Divisão 5	Média (%)
Precisão	84,77	87,34	86,56	88,62	87,69	87,0
Recall	83,34	85,36	86,04	88,92	87,13	86,2
F1-score	83,81	85,82	85,78	88,57	86,65	86,1
Exatidão	84,42	86,25	85,81	88,59	86,67	86,3

Os resultados observados indicam uma consistência entre as diferentes divisões, com variação máxima de 5,58% verificada apenas para o *recall*, refletindo o desempenho estável do modelo independentemente do conjunto de treino e validação utilizado.

O *F1-score* destaca-se pelo seu valor médio de 86,1%, demostrando que o modelo apresenta uma boa relação entre evitar falsos positivos e detetar corretamente as fissuras. A exatidão média de 86,3% reflete o desempenho global do modelo, indicando que uma grande parte das previsões foram realizadas corretamente. Estas observações reforçam que o modelo é adequado para a tarefa de deteção de fissuras, evidenciando um comportamento robusto em diferentes divisões de dados, cumprindo assim o objetivo da *cross validation*, isto é, avaliar o seu desempenho de forma confiável antes da sua aplicação nos dados de teste.

Em relação à função de perda, esta é essencial para avaliar o desempenho do modelo, pois reduz diversos parâmetros a um único valor representativo, permitindo entender o quão próximas as previsões estão dos valores reais. No contexto de problemas de deteção a perda total é composta por várias componentes, como a *loss_box_reg*, *loss_cls*, *loss_rpn_cls* e *loss_rpn_loc*. Cada um destes componentes reflete diferentes aspetos da aprendizagem do modelo, desde a precisão da regressão das caixas delimitadoras, a classificação dos objetos detetados e das propostas de regiões, entre outros. A Figura 4.8 demonstra como se comportou a função de perda total durante o treino e validação do modelo para cada uma das divisões, permitindo analisar como foi o seu ajuste ao conjunto de dados.

Tal como é esperado, para todas as divisões, a função de perda começa por apresentar uma redução inicial mais acentuada, indicando um ajuste rápido às características mais gerais dos dados. Também é de notar que, durante todo o treino do modelo, a tendência da curva da *training loss* é sempre de constante descida, ainda que nas iterações finais se verifique uma menor taxa de redução do seu valor. Este comportamento é esperado, visto que o modelo continua a ajustar progressivamente os seus parâmetros para representar da melhor forma as características dos dados de treino.

Nas curvas da *validation loss*, o comportamento diverge entre as diferentes divisões. As divisões 1 e 2 apresentam sinais de ligeiro *overfitting* precoce, com os valores da perda a oscilar em torno de um valor relativamente alto (~0,4) após cerca de 3000 iterações. Este comportamento indica que o modelo teve dificuldades em generalizar para os dados de validação, ainda que continuasse a melhorar o seu desempenho nos dados de treino. Por outro lado, as divisões 3, 4 e 5 apresentam um desempenho relativamente superior, com a *validation loss* a estabilizar em valores entre 0,2 e 0,3, com ligeiro *overfitting* a ser constatado mais tarde (entre 6000 e 8000 iterações). Estes resultados permitem concluir que os conjuntos de validação utilizados nas divisões 3, 4 e 5 são mais representativos ou simplesmente apresentam características mais fáceis de serem aprendidas pelo modelo.

Apesar do ligeiro *overfitting* observado, os modelos finais de cada divisão foram sempre guardados com base no *checkpoint* que apresentou o melhor desempenho nas métricas calculadas para os dados de validação. As iterações posteriores mantinham ou pioravam o desempenho, enquanto as anteriores apresentavam resultados inferiores. A Tabela 4.5 resume o tempo total de treino e o *checkpoint* final selecionado para cada divisão,

destacando os momentos críticos em que o desempenho nos dados de validação foi maximizado. O tempo médio de execução por iteração foi de 1,8 segundos.

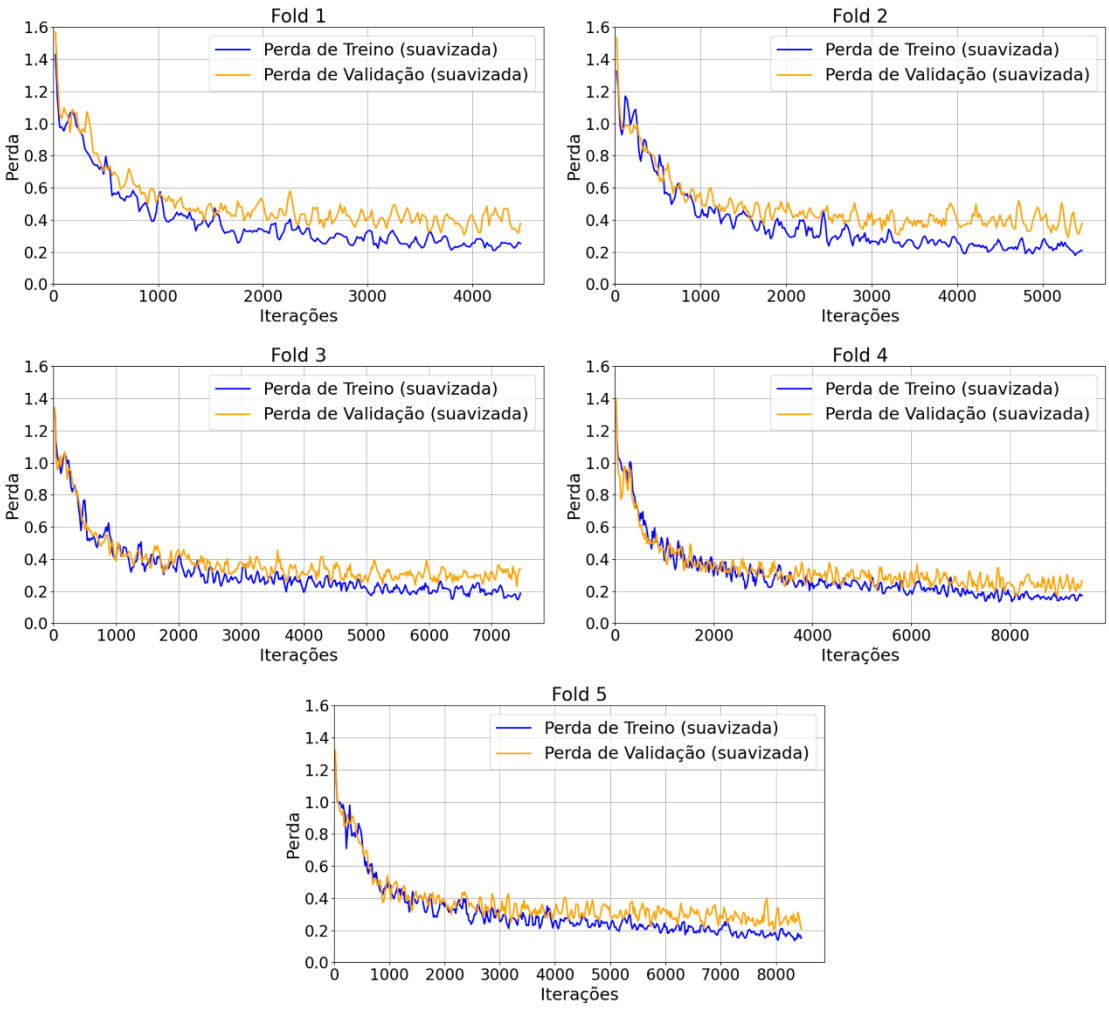

Figura 4.8: Gráficos da *training loss* e *validation loss* para cada uma das divisões de dados.

Tabela 4.5: Número total de épocas e tempo total de treino para cada divisão de dados.

Divisão	Total de Iterações	Tempo de Treino (h)
1	4499	2,12
2	5499	2,65
3	7499	3,99
4	9499	4,50
5	8499	4,10

4.2.2 Desempenho nos Dados de Teste

A avaliação do modelo de deteção nos dados de teste seguiu o procedimento análogo ao que foi realizado também para o modelo de classificação. Também com um total de 80

imagens, o conjunto de teste permitiu avaliar o desempenho real do modelo. A Tabela 4.6 apresenta os resultados obtidos pelo modelo final nos dados de teste.

Tabela 4.6: Resultados do modelo de deteção nos dados de teste.

Caso	Precisão (%)	*Recall* (%)	*F1-score* (%)	Exatidão (%)
Positivo	85,42	95,35	90,11	**89,3**
Negativo	94,44	82,93	88,31	
Média	**89,9**	**89,1**	**89,2**	

A precisão obtida para os casos positivos apresentou um valor de 85,42%, o que indica que o modelo é moderadamente confiável em evitar falsos positivos. Por outro lado, o *recall* de 95,35% sugere que o modelo consegue detetar a maioria das fissuras, um aspeto essencial para minimizar a quantidade de casos falsos negativos. O *F1-score* de 90,11% representa assim um bom equilíbrio entre a precisão e *recall* do modelo. Apesar das diferenças observadas principalmente nas métricas de precisão e *recall*, os resultados mostram que o modelo possui um desempenho global sólido para os casos positivos.

Nos casos negativos a precisão foi mais elevada do que para os casos positivos, com um valor de 94,44%, indicando que, de entre todas as previsões de ausência de fissuras, a maioria foi correta. No entanto, o *recall*, que mede a proporção de imagens sem fissuras corretamente identificadas, apresentou um valor relativamente mais baixo, de 82,93%, sendo o menor de entre todas as métricas calculadas. Este desempenho inferior pode ser explicado pelo menor número de imagens sem fissuras no *dataset* comparativamente ao número de imagens com fissuras, combinado com o facto de o número de falsos positivos verificado ser substancialmente maior que o de falsos negativos. O *F1-score* de 88,31%, reflete um desempenho geral sólido para os casos negativos, mas ainda ligeiramente inferior ao observado nos casos positivos.

Por fim, a exatidão obtida foi bastante elevada com um valor de 89,3%, indicando que o modelo apresenta um desempenho global bastante robusto e capaz de generalizar corretamente para novas imagens. De um modo geral, o modelo apresenta como principal ponto forte a sua alta capacidade de identificar fissuras, refletida no elevado *recall* para casos positivos, minimizando falsos negativos, o que é essencial em aplicações práticas onde a deteção de fissuras é crítica. Além disso, a precisão para casos negativos demonstra elevada confiabilidade em prever corretamente a ausência de fissuras. No entanto, a precisão para casos positivos e o *recall* para casos negativos indicam uma

maior presença de falsos positivos em comparação aos falsos negativos, o que pode originar, na prática, inspeções desnecessárias.

Para uma análise complementar foi também construída uma *confusion matrix*, apresentada na Figura 4.9.

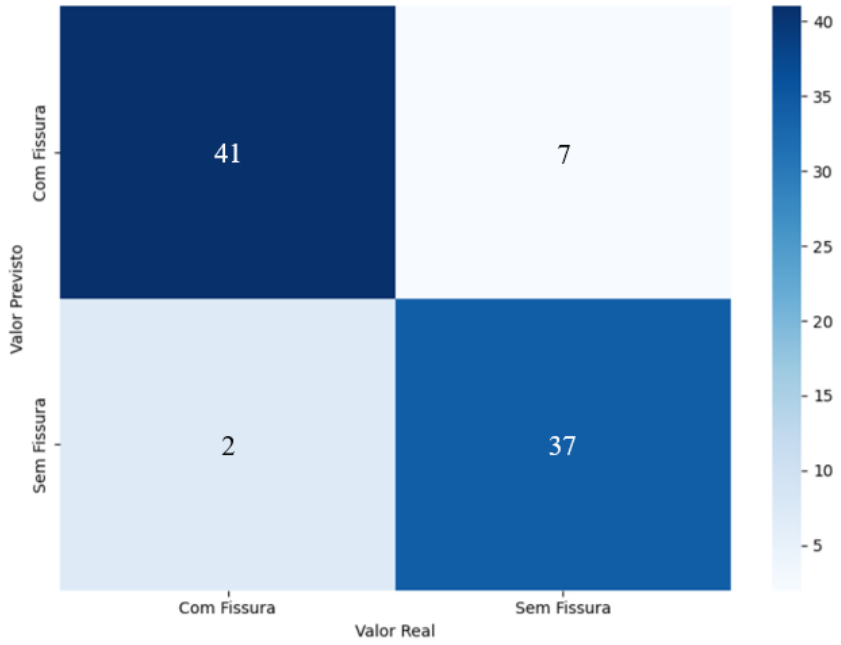

Figura 4.9: *Confusion matrix*.

Na Figura 4.10, são apresentados exemplos de previsões de fissuras realizadas corretamente. No caso (a), a previsão da fissuras apresenta uma confiança bastante alta (96%), com um IoU baixo (55%) e inferior ao valor limite estabelecido (75%). Isto acontece, pois, a caixa de previsão é relativamente maior do que a caixa de anotação segundo o eixo horizontal. Ainda assim, tendo em conta que a caixa de previsão consegue englobar a totalidade da caixa de anotação, a previsão é considerada correta. A capacidade de o modelo fazer mais do que uma previsão correta numa só imagem é evidenciada pelo exemplo (b), com duas previsões de elevada confiança e IoU. Os exemplos (c) e (d) demonstram também a capacidade de o modelo fazer previsões corretas para diferentes cenários, com fissuras de formas variadas.

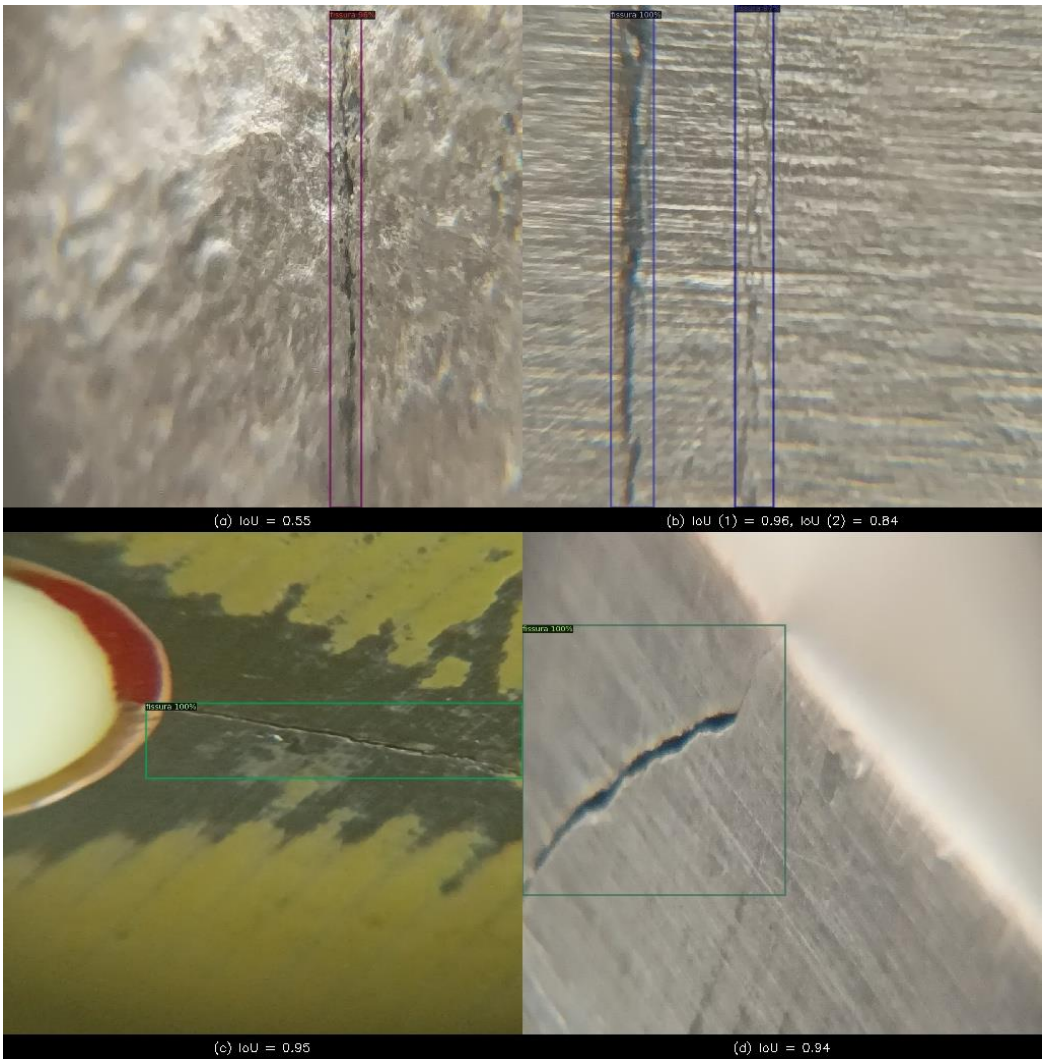

Figura 4.10. Exemplos de previsões corretas de fissuras.

Relativamente às previsões corretas de casos negativos, conforme evidenciado nos exemplos da Figura 4.11, é possível observar a capacidade do modelo em identificar corretamente superfícies sem fissuras e evitar falsos positivos. Como se pode verificar, os exemplos de superfícies sem fissuras apresentados são bastante distintos, variando em textura, iluminação, entre outros. No caso (a) é percetível que a superfície possui bastantes riscos e manchas que tornam a tarefa mais exigente. O caso (b), por outro lado, representa uma superfície com uma abertura, daí o contraste entre duas zonas com diferente iluminação. Os exemplos (c) e (d) representam outros cenários com texturas e cores diferentes, mantendo características já referidas como riscos e manchas.

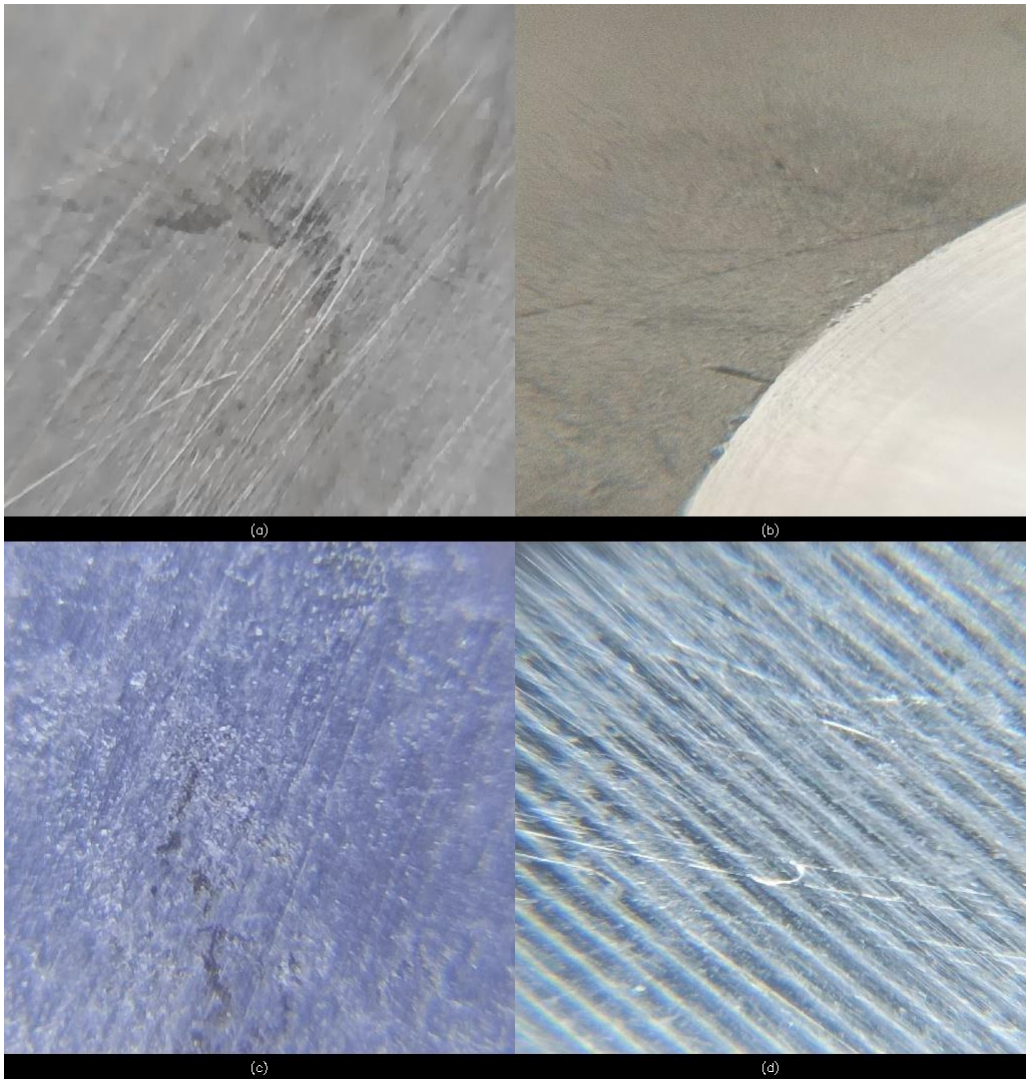

Figura 4.11. Exemplos de previsões corretas de casos TN.

No que toca às previsões incorretas, nomeadamente os casos falsos positivos, a Figura 4.12 apresenta dois exemplos onde o modelo realizou a previsão de uma fissura numa imagem que não possui nenhuma. O caso (a) demonstra uma previsão com 99% de confiança numa superfície com algumas características e padrões que facilmente podem ser confundidos com fissuras. Já no caso (b), a previsão feita possui uma confiança de apenas 68%. Como já foi mencionado, o valor do *threshold* definido para considerar ou não uma previsão foi 60%. Ao aumentar esse valor para 70%, por exemplo, esta previsão não seria considerada e passaria a ser um caso verdadeiro negativo, no entanto essa alteração iria favorecer o aparecimento de casos falsos negativos, que são considerados críticos para o problema em questão.

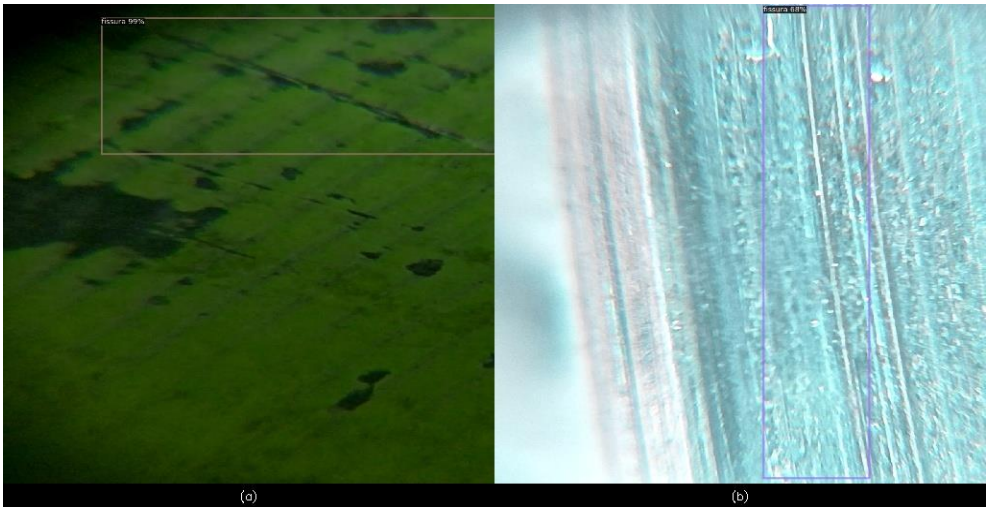

Figura 4.12: Exemplos de previsões de falsos positivos.

As restantes previsões incorretas dizem respeito aos falsos negativos e foram os casos registados em menor número, com apenas dois casos no conjunto de teste. Na Figura 4.13 são apresentados os dois exemplos de previsões FN, bastante semelhantes, onde o modelo foi incapaz de realizar a previsão sobre uma caixa de anotação existente. Tanto no caso (a) como no (b), as imagens possuem duas caixas de anotação, onde uma das fissuras foi corretamente prevista e uma fissura secundária, indicada a vermelho, não obteve qualquer previsão correspondente. Também é de notar que as fissuras que passaram despercebidas são de tamanho relativamente pequeno quando comparadas à maioria das que constituem todo o *dataset*.

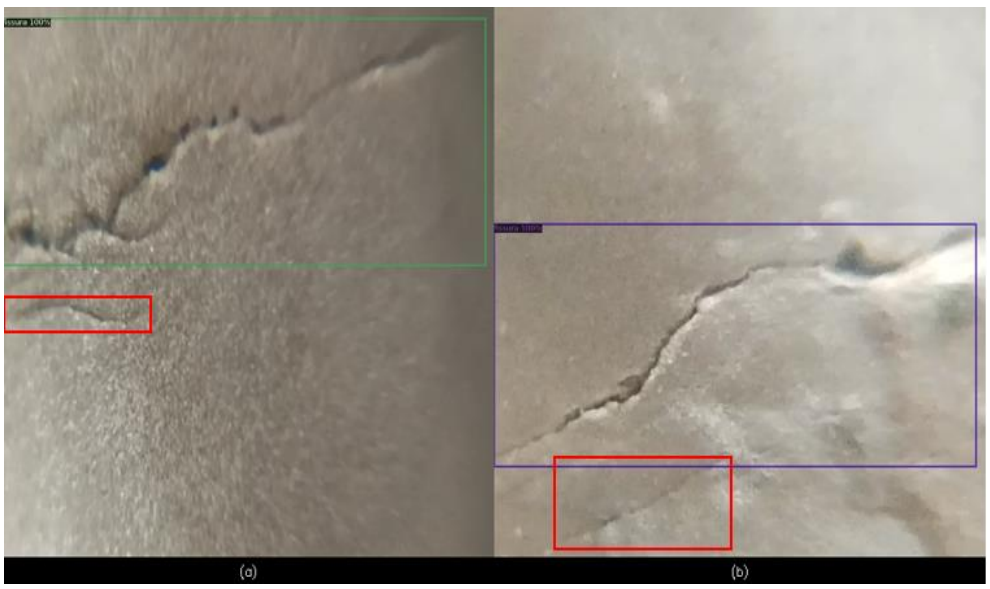

Figura 4.13: Exemplos de previsões de falsos negativos.

Apesar de se terem observado algumas limitações, os resultados nos dados de teste confirmam a capacidade do modelo de deteção de fissuras em generalizar adequadamente para novos dados. A análise do desempenho mostra que o modelo é capaz de realizar previsões corretas na maioria dos casos, mesmo com variações nos padrões ou complexidade dos dados. Esta robustez demonstra o potencial da sua aplicação em cenários reais, embora os erros obtidos devam ser considerados para trabalhos futuros.

Capítulo 5

5 Conclusões

Neste capítulo são apresentadas as conclusões finais do trabalho realizado, bem como as limitações encontradas e sugestões para trabalhos futuros.

Na presente dissertação foram analisados dois modelos de ML aplicados a problemas distintos: classificação de mecanismos de falha e deteção de fissuras em ligas metálicas usadas em componentes aeronáuticos. Ambos os modelos utilizaram CNNs como extratores de características, fazendo uso da técnica de *transfer learning* para aproveitar o conhecimento aprendido previamente em *datasets* maiores, o que é particularmente importante quando os *datasets* disponíveis são limitados.

Relativamente ao modelo de classificação, foi implementada uma arquitetura VGG 16 pré-treinada na ImageNet e adaptada para realizar a classificação dos três principais mecanismos de falha observados em ligas metálicas de componentes aeronáuticos das aeronaves da FAP. A avaliação nas imagens de teste revelou uma exatidão de 92,5% indicando que o modelo possui uma grande capacidade de generalização e é capaz de classificar com elevada confiança os três mecanismos de falha.

Para o modelo de deteção de fissuras foi implementado um detetor baseado em Faster R-CNN com ResNet 101 e FPN como extratores de características pré-treinados no *dataset COCO*. Este modelo foi adaptado para localizar e identificar fissuras em superfícies de ligas metálicas de aeronaves da FAP, bem como prever a ausência das mesmas. Com uma exatidão de 89,3% nas imagens de teste, o modelo demonstrou ser capaz de cumprir o seu objetivo de forma eficaz, validando assim a abordagem utilizada.

5.1 Trabalhos Futuros

Apesar dos resultados obtidos reforçarem o potencial dos modelos para aplicações práticas, também foram enfrentadas algumas limitações e dificuldades que podem ser corrigidas em trabalhos futuros.

No modelo de classificação, a principal limitação encontrada foi o tamanho reduzido do *dataset*, que comprometeu a capacidade de generalização do modelo, especialmente em relação à classe com características mais complexas, como a falha por sobrecarga. Apesar da recolha de imagens ter sido feita durante quatros dias, o processo é demorado e a quantidade de imagens obtidas por dia foi reduzida. Para além disso, o uso do computador pessoal para treino do modelo impôs algumas restrições como a redução do número total de épocas para evitar estender demasiado o treino em troca de pequenas melhorias. Como trabalho futuro, seria ideal:

- Aumentar a variedade e quantidade do *dataset*, para todas as classes.

- Explorar ferramentas computacionais mais capacitadas para a classificação de mecanismos de falha.

Para o modelo de deteção as limitações encontradas incluem também o tamanho reduzido do *dataset* e o poder computacional reduzido, mas também a arbitrariedade associada às anotações manuais das fissuras e o uso de caixas delimitadoras retangulares fixas, que podem capturar áreas irrelevantes do *background*. Algumas recomendações para eventuais trabalhos futuros seriam:

- Recolha de mais imagens para melhorar a capacidade de generalização do modelo, principalmente imagens com fissuras de pequeno tamanho.

- Definição de critérios mais rigorosos para os limites das anotações.

- Uso de caixas delimitadores rotativas.

Referências

Alom, M. Z., Taha, T. M., Yakopcic, C., Westberg, S., Sidike, P., Nasrin, M. S., Hasan, M., Van Essen, B. C., Awwal, A. A. S., & Asari, V. K. (2019). A state-of-the-art survey on deep learning theory and architectures. *Electronics*, *8*(3), 292. https://doi.org/10.3390/electronics8030292

ASM Handbook Committee. (1987). *ASM handbook: Fractography*. *Volume 12* (9th ed., Vol. 12). ASM International.

Bastidas-Rodriguez, M. X., Polania, L., Gruson, A., & Prieto-Ortiz, F. (2020). Deep learning for fractographic classification in metallic materials. *Engineering Failure Analysis*, *113*. https://doi.org/10.1016/j.engfailanal.2020.104532

Berrar, D. (2018). Cross-validation. in *Encyclopedia of Bioinformatics and Computational Biology* (Vol. 1, pp. 542–545). Elsevier. https://doi.org/10.1016/B978-0-12-809633-8.20349-X

Broek, D. (1984). Mechanisms of fracture and crack growth. In *Elementary Engineering Fracture Mechanics* (3rd ed., pp. 31–59). Martinus Nijhoff.

Callister Jr., W. D., & Rethwisch, D. G. (2013). Failure. In *Materials Science and Engineering: An Introduction* (9th ed., pp. 253–270). Wiley.

Chen, L., Li, S., Bai, Q., Yang, J., Jiang, S., & Miao, Y. (2021). Review of image classification algorithms based on convolutional neural networks. *Remote Sensing*, *13*(22), 4712. https://doi.org/10.3390/rs13224712

Cheng, B., Wei, Y., Shi, H., Feris, R., Xiong, J., & Huang, T. (2018, September). Revisiting RCNN: On awakening the classification power of faster RCNN. *Proceedings of the European Conference on Computer Vision (ECCV), 2018*.

Chollet, F. (2017). Fundamentals of deep learning. In *Deep Learning with Python* (1st ed., pp. 10–60). Manning.

Costa, R. D. F. da. (2020). *Detection and classification of road and objects in panoramic images on board the ATLASCAR2 using deep learning* [Dissertação de Mestrado]. Universidade de Aveiro.

CVAT: Computer Vision Annotation Tool. (n.d.). Retrieved October 18, 2024, from https://www.cvat.ai/

Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 248–255. https://doi.org/10.1109/CVPR.2009.5206848

Dowling, N. E. (2012). Fracture of cracked members. In *Mechanical Behavior of Materials: Engineering Methods for Deformation, Fracture, and Fatigue* (4th ed., p. 344). Pearson.

Duarte, D., Marado, B., Nogueira, J., Serrano, B., Infante, V., & Moleiro, F. (2016). An overview on how failure analysis contributes to flight safety in the Portuguese Air Force. *Engineering Failure Analysis*, *65*, 86–101. https://doi.org/10.1016/j.engfailanal.2016.03.003

Endo, A., Furuya, Y., Nagata, K., Yoshikawa, H., & Shouno, H. (2022). Fracture mode classification by texture analysis of fracture surface scanning electron microscope images. *Science and Technology of Advanced Materials: Methods*, *2*(1), 129–138. https://doi.org/10.1080/27660400.2022.2065185

Facebook AI Research. (n.d.). *Detectron2 Model Zoo and Baselines*. Retrieved November 7, 2024, from https://github.com/facebookresearch/detectron2/blob/main/MODEL_ZOO.md

Géron, A. (2019). The fundamentals of machine learning. In *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems* (2nd ed., pp. 3–8). O'Reilly Media, Inc.

Girshick, R. (2015). Fast R-CNN. *2015 IEEE International Conference on Computer Vision (ICCV)*, 1440–1448. https://doi.org/10.1109/ICCV.2015.169

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 580–587. https://doi.org/10.1109/CVPR.2014.81

Gonçalves, G. (2020). *A comparative study of data augmentation techniques for image classification: Generative Models vs. Classical Transformations* [Dissertação de Mestrado]. Universidade de Aveiro.

Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., Wang, L., Wang, G., Cai, J., & Chen, T. (2018). Recent advances in convolutional neural networks. *Pattern Recognition*, *77*, 354–377. https://doi.org/10.1016/j.patcog.2017.10.013

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778. https://doi.org/10.1109/CVPR.2016.90

Howard, J., & Gugger, S. (2020). Image classification. In *Deep Learning for Coders with fastai & PyTorch: AI Applications Without a PhD* (1st ed., pp. 202–203). O'Reilly Media.

Jabbar, H. K., & Khan, R. Z. (2015). Methods to avoid over-fitting and under-fitting in supervised machine learning (comparative study). *Computer Science, Communication & Instrumentation Devices*, 163–172.

Janssen, M., Zuidema, J., & Wanhill, R. J. H. (2002). Mechanisms of fracture in metallic materials. In *Fracture Mechanics* (2nd ed., pp. 294–308). VSSD.

Joiya, F. (2022). Object detection: Yolo vs faster R-CNN. *International Research Journal of Modernization in Engineering Technology and Science*, 1911–1915. https://doi.org/10.56726/irjmets30226

Khalifeh, A. (2020). Stress corrosion cracking behavior of materials. In *Engineering Failure Analysis* (pp. 1–21). IntechOpen. https://doi.org/http://dx.doi.org/10.5772/intechopen.90893

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, *60*(6), 84–90. https://doi.org/10.1145/3065386

Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324. https://doi.org/10.1109/5.726791

Leonard, J. K. (2019). Image classification and object detection algorithm based on convolutional neural network. *Science Insights*, *31*(1), 85–100. https://doi.org/10.15354/si.19.re117

Lin, T., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., & Zitnick, C. L. (2014). Microsoft COCO: Common objects in context. *European Conference on Computer Vision (ECCV)*, 740–755. https://doi.org/10.1007/978-3-319-10602-1_48

Meyers, M. A., & Chawla, K. K. (2012). Fracture: Microscopic aspects. In *Mechanical Behavior of Materials* (2nd ed., pp. 468–480). Cambridge University Press.

Möser, M. (1987). Fractography with the SEM (Failure Analysis). In H. Bethge & J. Heydenreich (Eds.), *Materials Science Monographs 40: Electron Microscopy in Solid State Physics* (pp. 366–385). Elsevier.

Narciso, T. C. (2022). *Deep learning para a classificação de imagens de raio-X* [Dissertação de Mestrado]. Universidade de Aveiro.

Pantazopoulos, G. A. (2011). Damage assessment using fractography as failure surface evaluation: Applications in industrial metalworking machinery. *Journal of Failure Analysis and Prevention*, *11*(6), 588–594. https://doi.org/10.1007/s11668-011-9503-7

Pothuganti, S. (2018). Review on over-fitting and under-fitting problems in machine learning and solutions. *International Journal of Advanced Research in Electrical Electronics and Instrumentation Engineering*, *9*, 3692–3695. https://doi.org/10.15662/IJAREEIE.2018.0709015

Rainio, O., Teuho, J., & Klén, R. (2024). Evaluation metrics and statistical tests for machine learning. *Scientific Reports*, *14*(1), 6086. https://doi.org/10.1038/s41598-024-56706-x

Ren, S., He, K., Girshick, R., & Sun, J. (2016). Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*(6), 1137–1149. http://arxiv.org/abs/1506.01497

Rosebrock, A. (2017). Faster R-CNNs. In *Deep Learning for Computer Vision with Python: Practitioner Bundle* (1st ed., pp. 252–253). PyImageSearch.

Russo, M. (1978). Analysis of fractures utilizing the SEM. In *Metallography in Failure Analysis* (pp. 65–95). Springer US. https://doi.org/10.1007/978-1-4613-2856-8_3

Santos, F. (2020). *Deep learning for multi-class skin lesion diagnosis* [Dissertação de Mestrado]. Universidade de Aveiro.

Simonyan, K., & Zisserman, A. (2015, April 10). Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations*. http://www.robots.ox.ac.uk/

Suguna, S. K., Dhivya, M., & Paiva, S. (2021). Computer vision concepts and applications. In *Artificial Intelligence (AI): Recent Trends and Applications* (1st ed., pp. 112–122). CRC Press.

Tsopanidis, S., Moreno, R. H., & Osovski, S. (2020). Toward quantitative fractography using convolutional neural networks. *Engineering Fracture Mechanics*, *231*, 106992. https://doi.org/10.1016/j.engfracmech.2020.106992

Wang, Q., Ma, Y., Zhao, K., & Tian, Y. (2022). A comprehensive survey of loss functions in machine learning. *Annals of Data Science*, *9*(2), 187–212. https://doi.org/10.1007/s40745-020-00253-5

Wang, S. Y., & Guo, T. (2021). Transfer learning-based algorithms for the detection of fatigue crack initiation sites: A comparative study. *Frontiers in Materials*, *8*. https://doi.org/10.3389/fmats.2021.756798

Wang, S. Y., Zhang, P. Z., Zhou, S. Y., Wei, D. B., Ding, F., & Li, F. K. (2020). A computer vision-based machine learning approach for fatigue crack initiation sites recognition. *Computational Materials Science*, *171*, 109259. https://doi.org/10.1016/j.commatsci.2019.109259

Weng, W.-H. (2020). Machine learning for clinical predictive analytics. In *Leveraging Data Science for Global Health* (pp. 199–217). Springer. https://doi.org/10.1007/978-3-030-47994-7_12

Wilimitis, D., & Walsh, C. G. (2023). Practical considerations and applied examples of cross-validation for model development and evaluation in health care: Tutorial. *JMIR AI*, *2*(1), e49023. https://doi.org/10.2196/49023

Yamagiwa, K. (2020). Application of deep learning for classification of fracture surface's SEM image. *Journal of the Society of Materials Science, Japan*, *69*(9), 644–649. https://doi.org/10.2472/jsms.69.644

Zhao, L., Pan, Y., Wang, S., Zhang, L., & Islam, M. S. (2021). A hybrid crack detection approach for scanning electron microscope image using deep learning method. *Scanning*, *2021*, 5558668. https://doi.org/10.1155/2021/5558668