isec
Engenharia

MESTRADO EM INFORMÁTICA E SISTEMAS

DEFINITIVO

Otimização de redes convolucionais para classificação de imagens

Autor

Gustavo Rodrigues Almeida

Orientador

Doutor Carlos Manuel Jorge da Silva Pereira

Coimbra, Março de 2021

INSTITUTO POLITÉCNICO
DE COIMBRA

INSTITUTO SUPERIOR
DE ENGENHARIA
DE COIMBRA

isec
Engenharia

DEPARTAMENTO DE INFORMÁTICA E SISTEMAS

Otimização de redes convolucionais para classificação de imagens

Relatório de Trabalho de Projeto para a obtenção do grau de Mestre em Informática e Sistemas

Especialização em Desenvolvimento de Software

Autor

Gustavo Rodrigues Almeida

Orientador

Doutor Carlos Manuel Jorge da Silva Pereira

INSTITUTO POLITÉCNICO
DE COIMBRA

INSTITUTO SUPERIOR
DE ENGENHARIA
DE COIMBRA

Coimbra, Março de 2021

AGRADECIMENTOS

Ao meu orientador Dr.º. Carlos Pereira pelo profissionalismo e compromisso que sempre demonstrou ao longo do desenvolvimento do projeto.

Um agradecimento especial aos meus pais que me incentivaram a nunca desistir e a ultrapassar as dificuldades sentidas. Foi também graças a eles que foi possível realizar o projeto, fornecendo todo o *hardware* necessário à sua realização. O esforço e trabalhado desenvolvido são lhe dedicados.

Aos meus avós, por serem uns segundos pais para mim. Sendo também responsáveis pela educação que me foi instruída.

Por último, quero agradecer também aos restantes familiares e amigos que contribuíram, indiretamente, para a conclusão do projeto.

RESUMO

As redes convolucionais têm demonstrado eficácia na resolução de diversos tipos de problemas, nomeadamente classificação de imagens, reconhecimento ou localização de objetos, reconhecimento de som, entre outros. A elevada robustez destas redes permite a sua difusão em áreas de extrema importância para a sociedade, tal como a área biomédica, que inclui um vasto conjunto de tarefas de análise de imagens e diagnóstico clínico.

Contudo, o sucesso destes modelos está dependente da identificação da estrutura e dos valores de outros hiperparâmetros, que melhor se ajustam à resolução de um problema específico, sendo que esta tarefa requer elevado esforço computacional e conhecimento pericial. De modo a reduzir estas dificuldades, este trabalho propõe uma metodologia de otimização autónoma dos hiperparâmetros de diferentes arquiteturas convolucionais.

A metodologia baseia-se na utilização do algoritmo de inteligência coletiva, *Particle Swarm Optimization* (PSO), aplicado na otimização de quatro conhecidas arquiteturas convolucionais: *AlexNet*, *VGGNet*, *ResNet* e *DenseNet* e para resolução de três problemas de diagnóstico médico. Procurando-se identificar a solução que proporciona o melhor desempenho com a menor complexidade possível.

Os resultados obtidos demonstraram a eficácia e a rapidez do PSO na identificação de soluções efetivas, sendo alcançados resultados superiores comparativamente a outras abordagens, em duas das três *benchmarks* em estudo.

A técnica *ensemble* proposta permitiu a obtenção de um $F1Score_{macroAVG}$ de 91.1% e de 96.6%, respetivamente, para as *benchmarks Breast Histopathology* e *Colorectal Histopathology*.

Os modelos estão disponíveis numa plataforma *web* que possibilita a realização de diagnósticos de imagens biomédicas, através do uso e da partilha de modelos convolucionais na plataforma.

A plataforma demonstrou ser extremamente eficiente e célere na resposta ao diagnóstico de imagens, sendo obtidas as previsões num intervalo inferior a 10 segundos. O seu acesso é público e está disponível no repositório https://github.com/bundasmanu/ProjetoMestrado.

Palavras-Chave: Redes de convolução, Inteligência coletiva, Classificação de imagens, Diagnóstico médico

ABSTRACT

Convolutional networks have shown effectiveness in solving various types of problems, including image classification, recognition or location of objects, sound recognition, among others. The high robustness of these networks allows their diffusion in areas of extreme relevance to society, such as the biomedical area, which includes a vast set of tasks of image analysis and clinical diagnosis.

However, the success of these models depends on the identification of the structure and hyperparameters that best suit the resolution of a specific problem, which requires a high computational effort and expert knowledge. In order to reduce these difficulties, this work proposes a methodology for autonomous optimization of hyperparameters of different convolutional architectures.

The methodology uses a collective intelligence algorithm, *Particle Swarm Optimization* (PSO), applied to the optimization of four well-known convolutional architectures: *AlexNet*, *VGGNet*, *ResNet* and *DenseNet* and to solve three medical diagnostic problems. Seeking to find the solution that provides the best performance with the lowest possible complexity.

The results achieved showed the effectiveness and speed of PSO in identifying efficient solutions, with higher results being reached compared to other approaches in two of the three benchmarks under study.

The proposed ensemble technique allowed the achievement of a $F1Score_{macroAVG}$ of 91.1% and 96.6%, respectively, for the Breast Histopathology and Colorectal Histopathology benchmarks.

The models are available on a web platform that allows the diagnosis of biomedical images by the use and sharing of convolutional models in the platform.

The platform has proven to be extremely efficient and fast in the response to the diagnosis of images, with the predictions being obtained in a range of less than 10 seconds. The platform is available at https://github.com/bundasmanu/ProjetoMestrado.

Keywords: *Convolution Neural Networks, Swarm Intelligence, Image classification, Medical diagnosis*

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE QUADROS

SIGLAS E ACRÓNIMOS

ACC	*Accuracy*
AdaGrad	*Adaptative Gradient Algorithm*
Adam	*Adaptative Moment Estimation*
API	*Application Programming Interface*
BD	Base de Dados
CGD	*Classic Gradient Descent*
CNN	*Convolutional Neural Network*
DNN	*Deep Neural Network*
EBI	*European Bioinformatics Institute*
GAP	*Global Average Pooling*
GPU	*Graphics Processing Units*
HTML	*HyperText Markup Language*
HTTP	*HyperText Transfer Protocol*
IDC	*Invasive Ductal Carcinoma*
ILSVRC	*Image Large Scale Visual Recognition Competition*
ISIC	*International Skin Imaging Collaboration*
LRN	*Local Responsive Normalization*
MAFS	*Macro Average F1Score*
MAR	*Macro Average Recall*
MIP	*Medical Informatics Plataform*
ML	*Machine Learning*
MLP	*Multilayer Perceptron*
MVT	*Model-View-Template*
NCBI	*National Center of Biotechnology Information*
PSO	*Particle Swarm Optimization*
ReLU	*Rectifier Linear Unit*
RProp	*Resilient Propagation*
SGD	*Stochastic Gradient Descent*

URL	*Uniform Resource Locator*

CAPÍTULO 1
INTRODUÇÃO

Este capítulo descreve sumariamente o âmbito do projeto e os objetivos propostos, sendo ainda apresentada a estrutura e a organização deste documento.

1.1 Visão e Âmbito

As doenças cancerígenas representam uma das principais causas de morte em todo o mundo. O processo de diagnóstico dos pacientes é exaustivo e complexo, sendo necessário recorrer a diversas técnicas de análise e a opiniões de diferentes médicos.

Nas duas últimas décadas, o avanço tecnológico permitiu a disseminação de uma grande variedade de dados médicos, o que promoveu a divulgação do conhecimento e a análise desta informação recorrendo a técnicas de *Machine Learning* (Ortega-Navas, 2017). O uso destas técnicas ajuda os profissionais de saúde em diversas tarefas, nomeadamente no diagnóstico, tratamento ou prognóstico de doenças.

As principais técnicas de diagnóstico de doenças cancerígenas baseiam-se no estudo e observação de tecidos humanos, imagens dermatológicas, ressonâncias magnéticas ou tomografia assistida por computador (Frangioni, 2008).

As *Convolutional Neural Network's* (CNN) representam uma sub-classe de redes neuronais e remetem ao final da década de 80 (Goodfellow et.al 2016). Nos últimos anos, a constante evolução tecnológica permitiu a sua difusão, sendo introduzidas inúmeras arquiteturas convolucionais, entre as quais a *AlexNet*, *VGGNet*, *ResNet, Inception, MobileNet* ou *DenseNet*. A sua utilização tem demonstrado enorme eficácia na classificação de imagens médicas (Greenspan et.al 2016). A configuração dos hiperparâmetros das redes convolucionais está dependente do problema em resolução, sendo que na maioria dos casos, este processo é extremamente exigente, quer ao nível do trabalho requerido ao cientista de dados, quer ao nível do tempo desperdiçado.

A tentativa manual de identificação dos valores destes hiperparâmetros, ou a aplicação de técnicas como *Random Search* ou *Grid Search*, revela-se pouco eficiente e impraticável. Dessa forma, recentemente, têm surgido diversas abordagens que exploram a utilização de algoritmos de otimização, para a consequente identificação dos valores dos hiperparâmetros de uma arquitetura CNN, que melhor se adequam à resolução de um problema específico.

A utilização de um algoritmo de otimização reduz o esforço necessário à correta identificação dos valores dos hiperparâmetros da rede, uma vez que permite a exploração "racional e automática" do espaço do problema. Para além disso, promove a conjugação de um *trade-off*

entre complexidade e desempenho, permitindo assim a identificação de modelos eficientes com a menor complexidade possível.

O *Particle Swarm Optimization* (PSO) é um algoritmo de *Swarm Intelligence* que é análogo ao comportamento social de indivíduos, como por exemplo, pássaros. O seu objetivo reside na procura pela solução global ótima de um problema. Para tal, as partículas movem-se ao longo do espaço de pesquisa, considerando o seu próprio conhecimento (aprendizagem cognitiva) e ainda o conhecimento de todo o bando (aprendizagem social).

O projeto a desenvolver visa a exploração do algoritmo PSO, na otimização da estrutura e de outros hiperparâmetros de quatro conhecidas arquiteturas de redes CNN: *AlexNet*, *VGGNet*, *ResNet* e *DenseNet*. As CNN serão aplicadas à resolução de três problemas de classificação de imagens médicas (doenças cancerígenas).

Como complemento ao projeto foi desenvolvida uma plataforma *web* de diagnóstico médico. Esta plataforma permite aos utilizadores o acesso e a partilha de modelos convolucionais, treinados para a resolução de problemas de classificação médicos. Os modelos submetidos na *app* são utilizados no diagnóstico de amostras médicas.

1.2 Objetivos e Contribuições

O principal objetivo do projeto compreende a utilização e a otimização de diferentes arquiteturas CNN na resolução de problemas médicos. O PSO deve identificar para cada arquitetura, a estrutura e os valores de outros hiperparâmetros, que promovem o melhor desempenho na resolução dos problemas propostos, sendo esperados resultados competitivos, quando comparados com abordagens de outros autores.

É promovida, ainda, a exploração das dificuldades existentes no setor médico, dessa forma as *benchmarks* propostas descrevem as principais limitações da área: informação limitada, classes não balanceadas e complexidade elevada.

A plataforma *web* deve ser robusta, proficiente e deve garantir rapidez na resposta ao diagnóstico de amostras.

As principais contribuições deste trabalho são:

- Implementação das quatro arquiteturas convolucionais estudadas, para consequente resolução de três problemas de classificação biomédicos;
- Otimização das arquiteturas recorrendo ao algoritmo PSO e respetiva análise das soluções obtidas;
- Análise comparativa entre os resultados obtidos e os alcançados por outros autores. Foram atingidos resultados superiores em duas das três *benchmarks*;
- Disponibilização pública da plataforma *web* e das soluções obtidas.

1.3 Estrutura e Organização do Relatório

Os capítulos 3, 4 e 5 descrevem o conhecimento teórico subjacente à proposta implementada, sendo contextualizados respetivamente os seguintes temas: algoritmo PSO, redes neuronais e redes CNN e técnicas de regularização das redes neuronais. Os três capítulos seguintes descrevem, respetivamente, a metodologia de otimização proposta, os resultados obtidos e a plataforma desenvolvida.

O relatório encontra-se assim organizado:

- **Capítulo 1**: Apresenta o âmbito do projeto, o que se pretende investigar e desenvolver e os objetivos propostos e contribuições esperadas;
- **Capítulo 2**: Descrição do estado da arte, em concordância com vários temas relacionados com o âmbito do projeto, nomeadamente: enumeração de diversos repositórios médicos, descrição de plataformas *web/mobile* de diagnóstico médico e contextualização de várias propostas de otimização de redes CNN recorrendo ao uso do PSO. Este capítulo resume o que já existe e quais as limitações ainda existentes;
- **Capítulo 3**: Discute o algoritmo de otimização *Particle Swarm Optimization*, através da descrição de detalhes gerais: Otimização e *Swarm Intelligence* e de detalhes técnicos: descrição matemática do algoritmo e contextualização das suas diferentes topologias;
- **Capítulo 4**: É um capítulo extenso que contextualiza com detalhe os principais conceitos inerentes às redes neuronais e às redes convolucionais.
- **Capítulo 5**: Contextualiza as diferentes técnicas de regularização das redes neuronais;
- **Capítulo 6**: Descreve a metodologia de otimização proposta, através da exposição das abordagens e considerações definidas ao longo da implementação dos objetivos indicados;
- **Capítulo 7**: Este capítulo esquematiza os resultados obtidos, sendo ainda, descrita uma análise comparativa entre os resultados alcançados e os atingidos por outros autores;
- **Capítulo 8:** Descreve a plataforma *web* desenvolvida;
- **Capítulo 9**: Sumariza todo o trabalho desenvolvido, destacando os objetivos atingidos e o trabalho futuro a desenvolver.

Para além destes capítulos, existe uma seção de Apêndices que incorpora informação adicional e complementar ao relatório, nomeadamente: a descrição e a análise das três *benchmarks* em estudo, a ilustração do resultado final da plataforma desenvolvida e a análise gráfica da variação da melhor *fitness* ao longo das iterações (para cada arquitetura otimizada).

CAPÍTULO 2
ESTADO DA ARTE

Este capítulo encontra-se dividido em três seções: estudo de repositórios que promovem a divulgação de dados médicos, análise de inúmeras abordagens que recorrem à utilização do PSO para a otimização de diferentes arquiteturas CNN e descrição de várias plataformas *web/mobile* de diagnóstico biomédico.

2.1 Repositórios de dados biomédicos

Os avanços tecnológicos facilitaram o acesso e a divulgação de dados médicos entre a comunidade. Contudo, atualmente ainda existem muitas condicionantes à sua divulgação, sobretudo devido às várias "políticas" de privacidade que estão subjacentes à sua distribuição.

O aumento do poder computacional, aliado ao baixo custo de armazenamento de enormes quantidades de dados, permitiu o surgimento de várias entidades que estimulam o acesso e a divulgação de dados médicos. Estas entidades liberam o acesso aos dados a toda a comunidade, permitindo assim que todos os utilizadores possam contribuir para a difusão de novos conhecimentos.

Estas entidades são intituladas como "repositórios", uma vez que centralizam uma vasta quantidade de informação no seu núcleo. Seguidamente, são enumerados alguns dos principais "repositórios" biomédicos.

O *European Bioinformatics Institute* (EBI) (Emmert *et al.*, 1994) é uma organização governamental que apoia a divulgação de dados e serviços bioinformáticos à comunidade. O principal foco da instituição prende-se com a divulgação de dados orientados à biologia molecular. Contudo, recentemente, têm sido investidos esforços em outras áreas, tais como: *bioimaging*, *biobanking* e fenotipagem de plantas.

O *National Center for Biotechnology Information* (NCBI) (Barrett *et al.*, 2005) faz parte do *National Institute of Health* e, tal como o EBI, promove o acesso e a divulgação de dados biomédicos entre a comunidade. O repositório *Gene Expression Omnibus* faz parte do NCBI e representa o maior repositório público contendo dados de expressão genética.

A *Medical Informatics Plataform (MIP)* (Markram *et al.*, 2011) é uma plataforma *web*, *open-source*, criada em 2016, e faz parte do projeto *Human Brain*. O *MIP* foi criado com o objetivo de estabelecer um mecanismo de comunicação interativo, entre cientistas de dados e clínicos. Permite a partilha e a divulgação de informações entre utilizadores e possibilita, ainda, o uso de algoritmos de *machine learning* no estudo e diagnóstico de doenças cerebrais. Este projeto foi criado com o apoio da Comunidade Europeia.

O *Cancer Imaging Archive* (Clark *et al.*, 2013) faz parte de um dos programas da divisão *Cancer Treatment and Diagnosis* (EUA) e promove o acesso e a partilha de *datasets* médicos.

O *Image Data Archive* (Crawford et.al., 2016) é um projeto criado em 2002 e faz parte do laboratório de neurociência da Universidade da Califórnia do Sul. Permite, também, o acesso e a partilha de imagens médicas.

O *HealthData.gov* é um *website* gerido pelo departamento de saúde dos EUA e reúne informação médica que é fornecida por diversas entidades do país.

Image Data Resource (Williams *et al.*, 2017) é uma plataforma *open-source* operada pela Universidade de *Dundee*. Esta plataforma divulga *datasets* que foram utilizados em artigos científicos.

O *SICAS Medical Image Repository* (Kistler, 2013) é um repositório público, suíço, que promove a partilha de informação médica a toda a comunidade, mas destina-se essencialmente aos seus parceiros: universidades e clínicas.

Kaggle é uma plataforma *web* que permite a partilha de conhecimento, incluindo dados médicos, entre a comunidade. A plataforma reúne um elevado conjunto de funcionalidades, nomeadamente: divulgação de dados e de soluções (*notebooks*), fórum de discussão, recrutamento, acesso a tutoriais sobre ciência de dados, disponibilização de hardware para resolução e publicação de soluções e promove ainda competições periódicas, a fim de identificar os melhores algoritmos para a resolução de problemas "reais". O sucesso do *Kaggle* é impressionante, tendo atingido, recentemente, a marca de 5 milhões de utilizadores registados.

As *benchmarks* utilizadas durante o estudo foram adquiridas através da plataforma *Kaggle*. A sua *interface* intuitiva e o vasto leque de informação disponível influenciaram esta escolha, em detrimento das restantes plataformas citadas. Para além disso, o *Kaggle* facilita a compreensão prévia de um problema, através das dúvidas e soluções partilhadas entre utilizadores.

2.2 Otimização de redes convolucionais com o algoritmo PSO

Nos últimos anos, o avanço da tecnologia permitiu aos investigadores, a exploração de atividades complexas e dispendiosas computacionalmente, como o Processamento de Imagem. A classificação de imagens representa uma das subáreas do processamento de imagem e, recentemente, o seu estudo tem revelado um enorme interesse da comunidade, uma vez que o progresso tecnológico permite a exploração de novos conceitos e abordagens, na resolução de problemas de classificação complexos (Aggarwal, 2018).

Estes avanços impulsionaram a retoma do estudo das redes convolucionais. As redes *CNN* têm-se revelado extremamente eficazes na resolução de problemas de classificação complexos, conseguindo mesmo transcender o conhecimento humano (Kokkinos, 2016; Geirhos *et al.*, 2018).

Contudo, o processo de construção de uma arquitetura *CNN* revela-se complexo, dada a tarefa árdua e demorada de identificação dos valores dos hiperparâmetros da rede, que melhor se ajustam à resolução de um problema específico.

Nos últimos anos, surgiram diversas abordagens que promovem a otimização dos hiperparâmetros das arquiteturas convolucionais. A utilização de algoritmos de *Swarm Intelligence*, mais concretamente o PSO, têm revelado resultados muito promissores, como comprovado através das abordagens descritas seguidamente, demonstrando que são inúmeras as vantagens da sua utilização, como: melhoria significativa do desempenho das arquiteturas, automatiza o trabalho do cientista de dados e permite a otimização de outros fatores, para além da avaliação do desempenho dos modelos, como por exemplo: a complexidade do modelo (número de parâmetros de treino), ou o tempo de treino.

Seguidamente são enumeradas várias abordagens, desenvolvidas por outros autores, que utilizam o PSO na otimização de redes CNN, aplicadas especificamente a problemas de classificação de imagens.

Bin Wang (2019) demonstrou a eficácia do PSO na otimização de modelos baseados na arquitetura *DenseNet*. Da sua aplicação resultaram modelos mais eficientes e menos complexos (aplicados no problema *Cifar*-10). Contudo, o autor limitou-se a otimizar o número de *composite blocks* (bloco composto por três operações consecutivas: *Batch Normalization*, *ReLU* e convolução), presentes nos 4 *dense blocks* e o hiperparâmetro *growth rate* (quantidade de informação adicionada por *composite block*).

Tatsuki Serizawa (2020) aplicou o algoritmo *Linearly Decreasing Weight Particle Swarm Optimization* na otimização de vários hiperparâmetros da arquitetura *LeNet-5*, nomeadamente: número de filtros, função de ativação e número de neurónios. Esta abordagem consiste no decréscimo linear da constante de inércia, do PSO, ao longo das iterações. É difícil julgar os resultados obtidos, no estudo dos problemas *Cifar-10* e *MNIST*, uma vez que o algoritmo otimizou apenas a arquitetura *LeNet-5*. Seria importante evidenciar a sua aplicação na otimização de arquiteturas mais complexas, ainda assim, o algoritmo demonstrou melhorias significativas no desempenho e convergência dos modelos.

Yanan Sun et.al. (2019) utilizou o PSO na otimização de *Stacked Flexible Convolutional Auto Encoders*[1], mais precisamente na identificação da estrutura do *encoder*. Um *Auto Encoder* é uma rede neuronal que promove a reconstrução dos dados de entrada, recorrendo a uma operação de *encoding* (aprende e reduz as dimensões de *input*) e de *decoding* (aprende e tenta reconstruir o *input* inicial), com vista à aproximação, mais clara possível, entre o *output* e o *input*. O autor comparou o desempenho da solução obtida, na resolução dos problemas *Cifar*-10 e *MNIST*, com outras técnicas *Auto Encoder*, tendo alcançado resultados superiores face às restantes técnicas.

[1] O *encoder* é composto por um número flexível de camadas convolucionais e de *pooling* não intercaladas, já o *decoder* representa a estrutura inversa do *encoder*.

Bin Wang et.al (2019) implementou um método híbrido baseado no algoritmo genético *(GA)* e no *PSO*. O método visa otimizar a arquitetura *DynamicNet*[2], através da utilização conjunta dos dois algoritmos. O PSO otimiza, numa primeira fase, os principais hiperparâmetros da rede: número de *dense* e de *composite blocks* e a constante *growth rate*. Posteriormente (após a definição da estrutura da solução – ao encargo do PSO) o GA define uma possível representação das conexões entre os *composite blocks*. Esta abordagem revelou bons resultados no problema *Cifar*-10, contudo é demorada e exigente computacionalmente.

Toshihiko Yamasaki et.al (2017) recorreu ao uso do *PSO* para a otimização de redes convolucionais baseadas na arquitetura *AlexNet*. Os autores assumem que a estrutura da rede já se encontra pré-definida, ou seja, limitam-se a otimizar os principais hiperparâmetros das camadas convolucionais e de *pooling*, nomeadamente: número de *feature maps*, *padding*, dimensões do *kernel* e técnica de *pooling*. Esta abordagem permitiu a obtenção de resultados ligeiramente superiores (problemas *Cifar*-10/100), face à não utilização do PSO.

Fe Ye (2017) utilizou o PSO na otimização dos hiperparâmetros do algoritmo *Steepest Gradient Descent* (algoritmo utilizado na otimização dos pesos das redes neuronais). O estudo evidenciou ligeiras melhorias no desempenho das redes, tendo em conta a resolução do problema de classificação *MNIST*.

Pablo Lorenzo et.al (2017) aplicou o *PSO* na otimização das arquiteturas *LeNet-4* e *SimpleNet*. O estudo comparou e demonstrou a elevada rapidez de convergência do algoritmo para "boas soluções" (problemas *Cifar*-10 e *MNIST*), em contraste com as restantes técnicas em análise, nomeadamente: *Grid* e *Random Search*.

As metodologias descritas, previamente, demonstraram os benefícios promovidos pelo algoritmo PSO na otimização de redes CNN, destacando-se: a automatização do processo de otimização das arquiteturas, a melhoria do desempenho das redes e a redução do esforço e tempo gasto na procura pela solução mais adequada para um problema. Foi ainda possível comprovar que o tempo despendido na otimização pode aumentar drasticamente, caso os problemas reúnam um elevado número de amostras, ou caso seja definido um elevado número de iterações, partículas ou *epochs* de treino. Sendo assim, estes parâmetros devem ser definidos, em conformidade com as limitações de *hardware* e de tempo existentes, caso contrário a utilização do PSO torna-se inviável.

A generalidade das metodologias citadas, restringem-se unicamente à otimização de uma ou de duas arquiteturas CNN e, para além disso, cingem-se à resolução de problemas de classificação genéricos, como: *Cifar*-10 ou *MNIST*. Dessa forma, é essencial analisar as vantagens que o algoritmo PSO promove, quando otimiza um maior número de diferentes arquiteturas CNN e tendo em conta a sua aplicação em problemas de cariz mais específico e com maiores limitações, como: maior complexidade, baixo número de amostras, amostras com dimensões elevadas e classes não balanceadas.

[2] Arquitetura semelhante à rede *DenseNet*, mas apresenta uma abordagem distinta e flexível de conexão entre *composite blocks*, isto é, cada *composite block* pode estar, ou não, ligado aos blocos subsequentes.

A proposta deste trabalho visa dar resposta às condicionantes descritas no parágrafo anterior. Sendo assim, serão analisados os benefícios promovidos pelo PSO, considerando a otimização de quatro conhecidas arquiteturas convolucionais: *AlexNet*, *VGGNet*, *ResNet* e *DenseNet*, a serem aplicadas na resolução de três problemas de classificação de imagens, referentes a doenças cancerígenas: cancro da mama, pele e colorretal, sendo expectável a obtenção de soluções com desempenho superior, face a outras abordagens.

2.3 Plataformas de diagnóstico médico

Recentemente, têm surgido algum esforço no estudo interativo de problemas de classificação de imagens e na promoção de um diagnóstico rápido e simples de diversas patologias. Ainda assim, o número de plataformas *web/mobile* desenvolvidas com o intuito de proporcionar essas vantagens à comunidade, é ainda muito reduzido.

A *ImJoy* (Ouyang *et al.*, 2019) é uma plataforma *web open-source* que promove o estudo de variados tipos de problemas, recorrendo a uma interface simples e intuitiva. As suas funcionalidades são acessíveis via *plugins* e permitem a resolução de problemas de classificação de imagens, segmentação de imagens ou a previsão da função de proteínas. É uma plataforma poderosa, contudo a realização de tarefas específicas ou a utilização de modelos convolucionais e dados externos requer a criação de *plugins*, exigindo que o utilizador reúna conhecimentos na área da programação.

StackML (StackML, 2020) é um projeto *open-source* que permite o acesso rápido a modelos pré-configurados na resolução de problemas de classificação de imagens, deteção de objetos ou classificação numérica. O acesso à plataforma é via *browser*, sendo que, os modelos treinam também eles no *browser* (recurso à CPU). Esta plataforma é robusta, contudo exige que o utilizador tenha conhecimentos informáticos, uma vez que requer a criação de código para a consequente realização das tarefas. Para além disso, apenas permite a utilização de modelos CNN pré-configurados ou criados na plataforma (através da sua funcionalidade de treino, não permite a submissão de modelos).

A *RunwayML* (Tsay *et al.*, 2018) é uma aplicação *desktop* semelhante à plataforma *StackML*, que disponibiliza um conjunto mais abrangente de funcionalidades e fornece uma abordagem de utilização mais simples e intuitiva. Possibilita a criação e o treino de modelos de *machine learning*, sem a necessidade de implementação de código que dê suporte a estas atividades. O treino dos modelos é processado, através de máquinas compostas por *hardware* poderoso e disponibilizadas através de serviços na nuvem. A *app* permite a submissão de dados externos, sendo possível interagir com diversos tipos de dados, como: imagens, vídeo, texto e som, possibilita a exportação de resultados e dispõe de um elevado conjunto de extensões (*Photoshop* ou *Unity*). Como contrapartida, a aplicação é paga, ou seja, são cobradas taxas durante a execução das máquinas remotas.

A ml5 (ml5-library, 2020) é um projeto *open-source* que apresenta um âmbito idêntico à plataforma *StackML*. O acesso à plataforma realiza-se, de igual forma, através de um *browser*

e os modelos são treinados, também eles, via *browser*. A *Application Programming Interface* (API) utilizada por ambas as plataformas é também muito semelhante entre si. Ao contrário da plataforma *StackML*, a ml5 permite o estudo e a resolução de problemas de classificação e reconhecimento de texto.

Existem ainda outras aplicações que apresentam um âmbito mais concreto, focando-se unicamente no estudo de um único problema, como a *app mobile SkinVision* (Carvalho *et al.*, 2019), que deteta lesões associadas ao cancro de pele.

A aplicação desenvolvida, *DiagnosisViewer*, apresenta um âmbito mais restrito, incidindo unicamente no diagnóstico de amostras biomédicas. Ao contrário das aplicações descritas previamente, o *DiagnosisViewer* possibilita a interação e a partilha de conhecimento entre utilizadores, através do acesso e da submissão de modelos convolucionais na plataforma. Para além disso, reduz as dependências de *hardware*, uma vez que os modelos submetidos já foram previamente treinados, proporcionando assim o acesso rápido ao diagnóstico e não requer a implementação de código.

CAPÍTULO 3
OTIMIZAÇÃO

Este capítulo descreve sinteticamente a teoria e os princípios inerentes à formulação de um problema de otimização, contextualiza o conceito de *Swarm Intelligence* e descreve os principais fundamentos do PSO.

3.1 Sistematização de um problema de otimização

Citando Parsopoulos e Vrahatis (2010): "Otimização é uma disciplina científica que procura as soluções ótimas para um determinado problema". Segundo Arora (2015): "Otimização consiste na identificação da melhor solução, entre muitas soluções viáveis".

O conceito de otimização é aplicado por toda a sociedade, mesmo nas tarefas mais corriqueiras. Todos os setores empresariais necessitam de otimizar custos de transporte, recursos humanos, ou inventário. É uma área que gradualmente revela maior importância, dadas as vantagens que promove quando aplicada corretamente.

Um algoritmo de otimização procura a melhor solução do problema com recurso à minimização ou maximização de uma função objetivo (Parsopoulos e Vrahatis, 2010). A função objetivo é responsável por aferir a qualidade das soluções.

A abordagem representada seguidamente esquematiza uma possível formulação matemática de um problema de minimização (Arora, 2015).

$$Minimizar\ f(x) \tag{1.1}$$

Sujeito a,

$$g_i(x) \leq 0, com\ i = 1,2,\dots,m < n \tag{1.2}$$

$$h_j(x) \geq 0, com\ j = 1,2,\dots,r < n \tag{1.3}$$

$$x_l \leq x \leq x_u\ , sendo\ \ x = [x_1, x_2, \dots x_n]$$

A função objetivo encontra-se declarada através da função 1.1. As funções 1.2 e 1.3 descrevem as restrições do problema, sendo responsáveis por reduzir o espaço de pesquisa admissível do problema (Parsopoulos e Vrahatis, 2010). Cada variável de decisão $x = [x_1, x_2, \dots x_n]$ é delimitada por um valor mínimo e máximo, representados respetivamente através das constantes x_l *e* x_u.

As variáveis de decisão podem ser discretas ou contínuas. Variáveis discretas refletem valores que são obtidos através de uma contagem finita, como por exemplo: número de alunos na sala.

Já as variáveis contínuas correspondem a uma contagem não finita, por exemplo: altura de uma pessoa.

Os algoritmos de otimização deterministas e estocásticos representam dois paradigmas utilizados na resolução de problemas de otimização (Venter, 2010). Um algoritmo é determinista, quando o resultado obtido depende das condições iniciais do problema. O algoritmo Trepa-Colinas (na sua abordagem "padrão") é determinista, uma vez que dadas as mesmas condições e parâmetros iniciais retorna sempre o mesmo resultado. Por outro lado, os algoritmos estocásticos revelam algum tipo de aleatoriedade, levando a que o algoritmo quando exposto às mesmas condições e parâmetros iniciais, possa evidenciar resultados distintos. Um exemplo de algoritmo estocástico é o *Particle Swarm Optimization*.

Segundo Parsopoulos e Vrahatis (2010) a resolução de um problema de otimização pode revelar-se um processo complexo, sendo várias as atenuantes que dificultam a otimização:

- Elevado número de dimensões;
- Presença de "ruído" na avaliação das soluções (erros de medida, incapacidade de reproduzir resultados);
- Dificuldade em distinguir soluções ótimas locais e globais (pode originar convergência prematura para ótimos locais);

3.2 Swarm Intelligence

Segundo Panigrahi *et.al* (2011), *Swarm Intelligence*: "descreve a relação de indivíduos poucos sofisticados que interagem com o ambiente, de modo a construir algoritmos na resolução de problemas". Para Parsopoulos e Vrahatis (2010) "representa um ramo da inteligência artificial que estuda o comportamento colectivo e as propriedades de sistemas complexos, auto-organizativos e com uma estrutura social definida".

Resumidamente, *Swarm Intelligence* baseia-se na estimulação de comportamentos estabelecidos por múltiplos agentes, que comunicam entre si, com o intuito de alcançar um objetivo comum (Clerc, 2010). A sua definição é análoga a comportamentos sociais que são emitidos por algumas espécies da sociedade, como por exemplo pássaros, insetos ou peixes (Panigrahi et.al 2011).

Cada individuo do enxame, por si só, não apresenta "inteligência", isto é, as suas capacidades de decisão estão dependentes das ações proporcionadas por todo o bando, promovendo assim o conceito de aprendizagem coletiva (Hassanien e Alamry, 2015).

Segundo Kennedy e Eberhart (2001) existem três princípios que influenciam o comportamento dos agentes:

- Avaliação: Os agentes avaliam os seus estímulos;
- Comparação: Os agentes comparam o seu comportamento com os demais;

- Imitação: Os agentes replicam os estímulos realizados por outros agentes.

O conceito de *Swarm Intelligence* foi aplicado inicialmente por Beni e Wang (1993) através da implementação de um algoritmo, baseado em enxames de partículas, para controlo e coordenação de robôs. A partir daí, surgiram inúmeros algoritmos de *Swarm Intelligence*, como por exemplo: *Particle Swarm Optimization* ou *Ant Colony Optimization*.

3.2.1 Particle Swarm Optimization

Citando Clerc (2010) "o PSO é um método iterativo representado por um coletivo anárquico, com enfâse na cooperação entre indivíduos".

O PSO é um algoritmo de otimização estocástica que se baseia no conceito de *Swarm Intelligence*, simulando mais precisamente, o comportamento de pássaros (Wang et.al 2017).

O princípio geral do PSO consiste na distribuição aleatória de *n* partículas, ao longo do espaço de procura do problema. Cada partícula representa uma possível solução do problema. Ao longo das iterações, as partículas movimentam-se no espaço, explorando outras possíveis soluções. O movimento das partículas é condicionado por vários fatores: velocidade atual da partícula, conhecimento prévio da partícula (memória) e pelo conhecimento global do enxame.

O algoritmo foi introduzido inicialmente por Eberhart e Kennedy (1995), estendendo e aperfeiçoando o modelo *Boid* (Reynolds, 1987). Desde então, foram desenvolvidas inúmeras adaptações do algoritmo.

Shi e Eberhart (1998) propuseram a adição de um novo hiperparâmetro no cálculo da velocidade das partículas, a constante de inércia ω. A inércia é uma constante que varia geralmente entre 0 e 1 e atua como um regulador no ato de movimentação das partículas, tentando reduzir a divergência do *swarm*.

Van den Bergh e Engelbrecht (2004) desenvolveram uma abordagem de cooperação entre diferentes *swarms*. Para tal, consideraram a decomposição do espaço do problema em subdivisões menores, sendo que cada *swarm* é responsável por otimizar uma subdivisão do espaço dividido.

Yang et.al (2007) introduziram duas novas constantes, *evolution speed factor* e *aggregation degree factor*, ao algoritmo. Estas duas constantes influenciam dinamicamente a constante de inércia, reduzindo a probabilidade das partículas ficarem "presas" em ótimos locais.

Alrijadjis Djoewahir et.al (2012) desenvolveram o algoritmo PSO-NDW (*Nonlinear Decreasing Weight*). Este algoritmo implementa um decréscimo não linear da constante de inércia ao longo das iterações, tendo demonstrado eficácia na otimização de um motor ultrassónico (otimização da velocidade de rotação, *holding torque*, *etc*).

Tanweer et.al (2016) reformularam o conceito de movimentação das partículas, propondo um mecanismo dinâmico de auto-regulação dos seus movimentos. A abordagem consiste na divisão das partículas em três grupos distintos: tutores, aprendizes e aprendizes independentes. Cada

grupo revela estratégias diferenciadas de atualização das suas velocidades: os tutores movimentam-se de acordo com o seu conhecimento e de todo o grupo, os aprendizes são orientados pelos tutores e os aprendizes independentes exploram o espaço livremente.

3.2.1.1 Formulação Matemática

Como fora indicado previamente, o PSO é um algoritmo estocástico baseado na aprendizagem em enxame. O primeiro passo subjacente à sua implementação, consiste na definição do número total de partículas do enxame e do número de iterações a considerar. O número de partículas e de iterações está dependente da complexidade do problema, isto é, problemas simples requerem um menor número de partículas e/ou iterações. Segundo Clerc (2010) o número de partículas *standard* situa-se entre 20 e 30. Diferentes estudos (Cui *et al.*, 2017; Prathabrao et.al., 2017) demonstraram o impacto que a escolha adequada destes parâmetros promove na solução final.

Posteriormente é necessário identificar as dimensões do problema (variáveis a otimizar). Cada dimensão é delimitada por um limite inferior e superior, estabelecendo o espaço de procura admissível do problema (Panigrahi et.al., 2011).

Do mesmo modo, é necessário definir a função objetivo. Tipicamente (problema de objetivo-único), a função objetivo é semelhante à equação traduzida através da função 1.1. Em cada iteração, cada partícula é avaliada recorrendo à função objetivo definida. Após a aplicação dos passos anteriores, o processo iterativo do PSO pode ser iniciado.

Primeiramente são definidas as posições e velocidades iniciais das partículas. São diversas as abordagens que podem ser utilizadas neste processo. A metodologia usual consiste na dispersão aleatória das partículas pelo espaço admissível do problema, recorrendo a uma distribuição uniforme (Eberhart e Kennedy, 1995).

As equações 2 e 3 descrevem, respetivamente, o processo de inicialização da posição e velocidade de uma partícula (Panigrahi et.al., 2011).

$$x(0)_d = u\,(x_{min,d}, x_{max,d}) \tag{2}$$

$$\mathrm{v}\,(0)_{\,d} = 0.5\,(u\,(x_{min,d}, x_{max,d}) - x(0)_d) \tag{3}$$

A equação 2 inicializa a posição de uma partícula no espaço, recorrendo à geração de um valor aleatório (distribuição uniforme) para cada dimensão *d* do problema. A equação 3 é semelhante à equação anterior, isto é, gera um valor aleatório para cada dimensão *d* do problema, sendo estes valores subtraídos pela consequente posição inicial da partícula (equação 2). Finalmente é aplicada a técnica *half-diff*, que consiste na multiplicação do vetor resultante (subtração) por 0.5.

Ao longo das iterações, as partículas movem-se no espaço do problema. Os movimentos executados pelas partículas são avaliados, em cada iteração, e têm influência na determinação das suas próximas posições. As equações 4 e 5 traduzem respetivamente o cálculo da velocidade e da posição de uma partícula, na iteração *t* (Shi e Eberhart, 1998).

$$v_i(t+1) = wv_i(t) + c1\,R1\,(pbest_i - x_i(t)) + c2\,R2\,(gbest - x_i(t)) \quad (4)$$

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (5)$$

A velocidade de uma partícula consiste no somatório de três componentes distintas: influência da velocidade atual da partícula, conhecimento prévio da partícula e conhecimento social.

O cálculo da 1ª componente considera a multiplicação entre a velocidade atual da partícula v_i e a constante de inércia w. Ou seja, geometricamente, esta multiplicação cria um vetor com a mesma direção e sentido, do vetor velocidade calculado na iteração anterior (Kennedy e Eberhart, 2001; Clerc, 2010; Parsopoulos e Vrahatis, 2010).

A 2ª componente $c1\,R1\,(pbest_i - x_i(t))$ descreve a atração linear da partícula *i* para a posição com melhor *fitness* em que esta já esteve (memória da partícula). $c1$ representa a constante cognitiva, o seu uso determina a influência que o conhecimento prévio da partícula tem no cálculo da sua velocidade. A constante *R1* descreve um valor aleatório uniforme, entre 0 e 1, e controla estocasticamente a influência da constante cognitiva no cálculo da velocidade da partícula. *Pbest* traduz a posição com melhor *fitness* em que a partícula já esteve. $x_i(t)$ representa a posição atual da partícula.

A 3ª componente $c2\,R2\,(gbest - x_i(t))$ simboliza a atração linear da partícula *i* para a posição com melhor *fitness* identificada pelo enxame/vizinhança até ao momento. *C2* representa a constante social, isto é, determina a influência que o conhecimento coletivo do enxame apresenta no cálculo da velocidade de uma partícula. A constante *R2* reúne o mesmo propósito da constante *R1*, contudo, controla estocasticamente a influência da constante social no cálculo da velocidade da partícula. *Gbest* representa a posição com melhor *fitness* identificada, até então, pelo enxame (ou pela vizinhança da partícula). Do mesmo modo, $x_i(t)$ descreve a posição atual da partícula.

A próxima posição de uma partícula é calculada com recurso à equação 5, considerando o somatório entre a posição atual da partícula $x_i(t)$ e o vetor velocidade atualizado da partícula, $v_i(t+1)$, que fora calculado previamente através da equação 4.

A Figura 1 esquematiza visualmente o cálculo da próxima posição de uma partícula. A seta ilustrada a vermelho representa o vetor velocidade, resultante do somatório das três componentes descritas nos parágrafos anteriores (equação 4), nomeadamente: influência da velocidade atual da partícula (seta representada a amarelo), atração para a melhor posição onde a partícula já esteve (seta representada a verde) e atração para a melhor posição identificada pelo enxame (seta representada a azul).

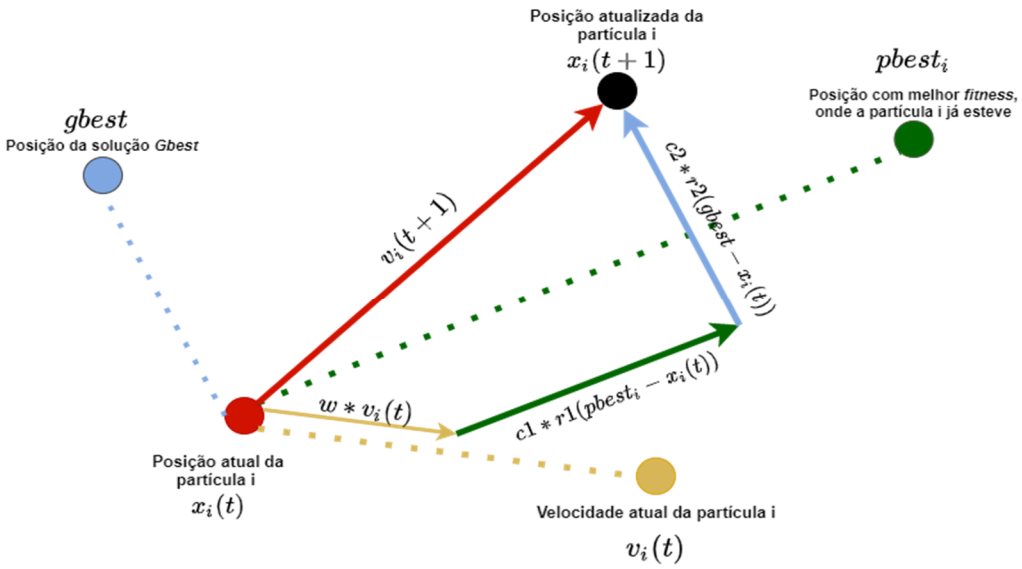

Figura 1 - Representação visual do cálculo da próxima posição de uma partícula

3.2.1.2 Formulação Computacional

A Figura 2 exibe o fluxograma do algoritmo PSO. Seguidamente são descritos sinteticamente os passos incluídos no fluxo.

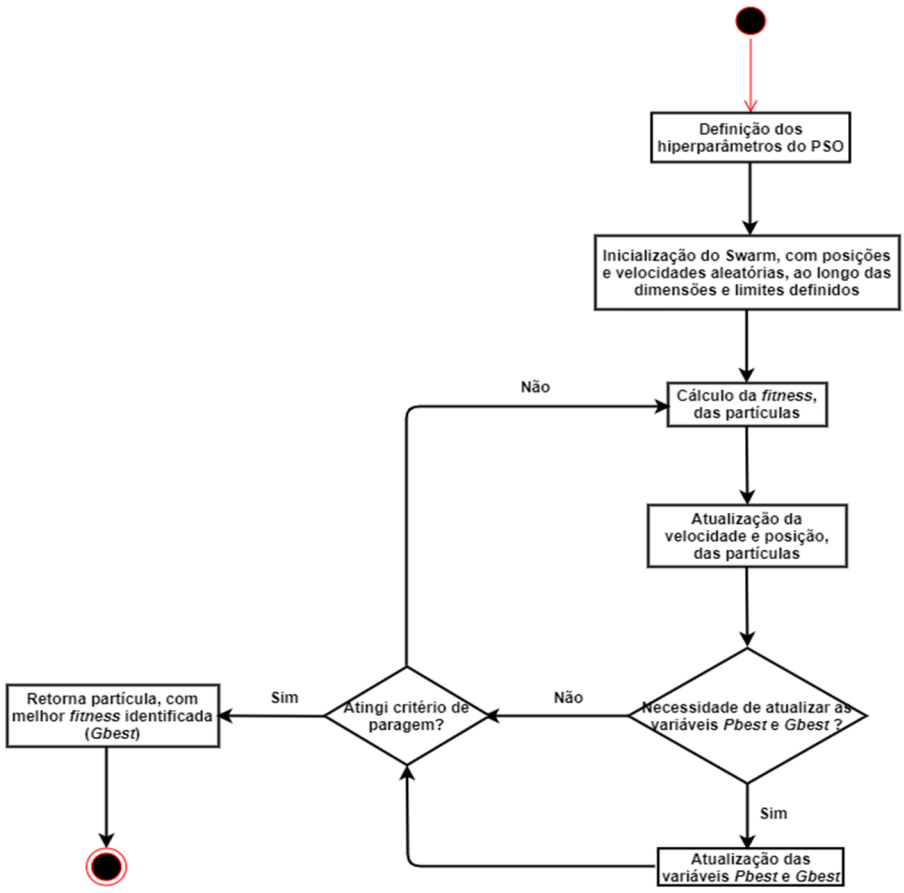

Figura 2 - Fluxo de aplicação do PSO

Preliminarmente é necessário proceder à inicialização do *swarm*, isto é, devem ser definidas as posições e velocidades iniciais das partículas, em conformidade com as dimensões e limites fixados. Para além disso, é necessário determinar os valores adequados para os hiperparâmetros do PSO, nomeadamente: constante cognitiva (*c1*), constante social (*c2*), inércia (*w*) e número de iterações e de partículas.

Em cada iteração, o movimento das partículas é avaliado e procede-se à atualização das suas posições e velocidades, considerando o uso das equações 4 e 5. Caso, a *fitness* atual de uma partícula, seja inferior à melhor *fitness* obtida pela partícula, até ao momento, então a variável *pbest* é atualizada (considerando, um problema de minimização). A variável *gbest* é alterada, caso a *fitness* atual de uma partícula, seja inferior à melhor *fitness* obtida, até ao momento, pelo enxame (considerando um problema de minimização).

A abordagem descrita nos dois parágrafos anteriores é repetida até que um critério de paragem seja atingido, como por exemplo, a *fitness* da solução *gbest* seja igual ou inferior a um valor mínimo ideal (considerando, um problema de minimização). Caso, não seja definido nenhum critério de paragem, então o algoritmo é executado durante o número de iterações que foram definidas inicialmente. No final, é retornada, a solução que apresentou melhor *fitness*.

3.2.1.3 Abordagens de controlo de soluções inválidas

À medida que as partículas se movimentam no espaço, é natural que extraviem ocasionalmente os limites das dimensões problema, tornando-se assim soluções inválidas.

Nesse sentido, existem várias abordagens que são consideradas, de modo a garantir que as soluções se mantêm dentro do espaço admissível do problema (*boundary handling approaches)*. Existem métodos que reparam as posições das partículas e também métodos que reparam a sua velocidade, podendo estes, serem utilizados em simultâneo (Xu e Rahmat-Samii, 2007; Helwig, 2010; Oldewage et.al., 2018).

Os principais métodos de reparação de posições inválidas são:

- *Random:* A posição de uma partícula, em cada dimensão excedida, é reinicializada aleatoriamente dentro dos limites do problema;
- *Reflect*: Esta técnica aplica o conceito de reflexão, ou seja, a posição da partícula em cada dimensão excedida é ajustada, aplicando um vetor ortogonal à sua velocidade, a partir da posição anterior da partícula, ou seja, o espaço excedido é espelhado para o interior dos limites da dimensão extrapolada;
- *Intermediate*: Para cada dimensão inválida, a partícula é reposicionada no ponto médio, entre a sua posição anterior (posição anterior da partícula na dimensão excedida) e o "ponto de contacto" com o limite extraviado;
- *Infinity*: Esta estratégia permite que as partículas possam extraviar os limites das dimensões do problema. De modo, a garantir que as partículas regressam novamente a

posições válidas, geralmente procede-se à penalização severa da sua *fitness* (valores muito desencorajadores), esperando que as mesmas se tornem novamente válidas;

- *Shrink*: Caso, a partícula exceda os limites de uma dimensão, então a partícula para no limite excedido;
- *Periodic*: Esta estratégia não altera a posição nem a velocidade da partícula. A técnica consiste na "extensão" do espaço do problema, ou seja, caso a partícula extrapole os limites de uma dimensão, esta reposiciona-se ao longo do outro limite (a partícula continua o seu movimento, no início do outro limite).

Do mesmo modo, existem métodos de reparação de velocidade (Xu e Rahmat-Samii, 2007; Helwig, 2010; Oldewage et.al., 2018):

- *Unmodified*: Consiste na manutenção da velocidade da partícula;
- *Zero*: Caso, a partícula extrapole os limites do problema, então a sua velocidade é colocada a 0 nas dimensões excedidas;
- *Adjust*: A velocidade da partícula é ajustada considerando a seguinte equação: $v_{i,t} = x_{i,t} - x_{i,t-1}$ (i representa o índice da partícula e t a iteração). Este método é normalmente utilizado, em simultâneo, com uma técnica de reparação de posição e atenua a probabilidade das partículas caírem novamente em posições inválidas;
- *Invert:* Inverte o sentido da velocidade da partícula em cada dimensão excedida, para tal considera o uso de uma constante $0 \leq z \leq 1$ que define a "força" do vetor inverso.

3.2.1.4 Topologias

D. J. Watts (1998; 1999) concluiu que a comunicação entre partículas era afetada por diversos fatores, entre os quais: o grau de conectividade entre os nós do enxame, o número médio de vizinhos em comum (por nó) e ainda a distância média entre nós.

O conceito de vizinhança determina quais as partículas que estão "ligadas e dependentes" entre si, num contexto de aprendizagem e de troca de informações (Sengupta et.al., 2018).

As topologias são geralmente representadas sob a forma de grafo, onde cada partícula representa um nó no espaço e as arestas simbolizam a respetiva comunicação/vizinhança entre as partículas (Clerc, 2010; Helwig, 2010; Hassanien e Alamry, 2015).

Seguidamente são contextualizadas as principais topologias do PSO.

A topologia em estrela, ou *gbest*, foi introduzida por Eberhart e Kennedy (1995) e define uma conexão global entre todas as partículas do enxame. Ou seja, todas as partículas estão conectadas entre si, garantindo que o movimento de cada partícula é influenciado pelo comportamento das demais (Kennedy e Mendes, 2002).

Esta topologia permite uma rápida convergência das partículas, uma vez que a informação é comunicada e distribuída ao longo de todo o enxame (Panigrahi et.al., 2011). Um estudo

realizado por Engelbrecht (2013) comprovou que a topologia *gbest* é eficaz na resolução de variados tipos de problemas: unimodal, multimodal, separáveis ou não separáveis.

A Figura 3 esquematiza a Topologia em Estrela.

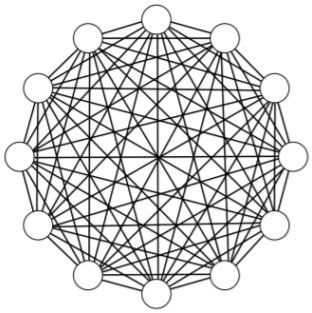

Figura 3 - Topologia em Estrela (Gbest) - Fonte: (Panigrahi et.al 2011)

A Topologia em Anel, ou *lbest*, foi introduzida por Eberhart e Kennedy (1995) e considera que uma partícula está conectada, unicamente, a outras duas partículas. Esta abordagem evita que o movimento de uma partícula seja influenciado por todas as partículas do enxame, sendo afetado apenas pelo comportamento das duas partículas às quais está conectada (Kennedy e Mendes, 2002; Sengupta et.al., 2018). A Figura 4 esquematiza a topologia em Anel.

A definição das conexões entre partículas pode ser estabelecida recorrendo a várias abordagens, entre as quais: índices das partículas (ligação às 2 partículas com índices adjacentes: *i-1* e *i+1*), ligação aleatória (as partículas ligam-se aleatoriamente a outras duas partículas) ou através da utilização de medidas de distância (distância euclidiana, distância de *Manhattan*, *etc*).

Esta topologia reduz a dependência entre partículas, demonstrando tendência para apresentar uma convergência mais lenta do que a topologia em estrela. Por sua vez, é menos propensa à "queda precoce" em ótimos locais (Kennedy e Mendes, 2002; Hassanien e Alamry, 2015).

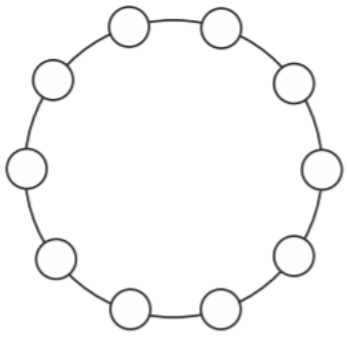

Figura 4 - Topologia em Anel - Fonte: (Kennedy e Eberhart, 2001)

A Topologia em Roda representa um caso particular da Topologia em Anel. Cada partícula comunica unicamente com uma partícula central, que, por sua vez, está conectada a todas as partículas do enxame. Esta abordagem abranda o processo de convergência das partículas (Kennedy, 1999).

Apesar de atenuar a probabilidade de queda para ótimos locais, reduz a importância da comunicação entre o enxame (Kennedy e Eberhart, 2001). Podendo resultar em baixas

performances (fracas soluções finais, muito distantes da solução ideal) e convergência muito demorada. A Figura 5 esquematiza a topologia em roda.

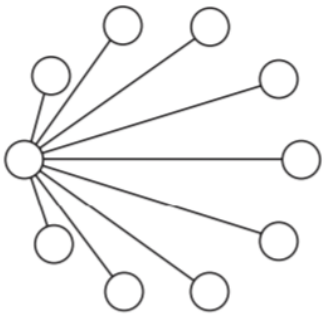

Figura 5 - Topologia em Roda - Fonte: (Kennedy e Eberhart, 2001)

A topologia em pirâmide foi introduzida por Mendes et.al (2002). Segundo Mendes (2002), a topologia em pirâmide descreve uma estrutura de ligações triangulares. As partículas são representadas através dos cumes dos triângulos, enquanto que as arestas simulam a comunicação entre partículas (Clerc, 2010). Esta topologia descreve uma generalização da estrutura em anel. A Figura 6 ilustra a topologia em Pirâmide.

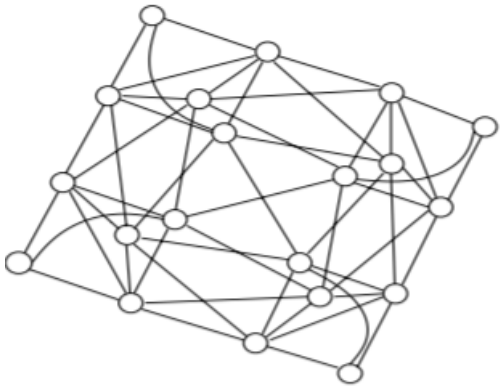

Figura 6 - Topologia em Pirâmide - Fonte: (Engelbrecht, 2007)

A topologia *Von Neumann* foi também introduzida por Mendes et.al (2002) e representa, de igual forma, uma variante da topologia *lbest*, onde cada solução está conectada a outras quatro partículas, descrevendo uma estrutura em forma de grelha. A Figura 7 ilustra a topologia *Von Neumann.*

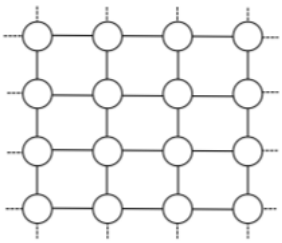

Figura 7 - Topologia *Von Neumann* - Fonte: (Panigrahi et.al., 2011)

CAPÍTULO 4
DEEP LEARNING

Este capítulo sistematiza vários conceitos essenciais e subjacentes ao âmbito do projeto, entre os quais: aprendizagem supervisionada, rede neuronal profunda e redes de convolução.

4.1 Aprendizagem Supervisionada – Classificação

A aprendizagem supervisionada consiste num processo que recorre à utilização de conhecimento prévio na tentativa de extrair e assimilar características dos dados. Como resultado é expectável que o conhecimento adquirido possa ser adaptado na previsão de novos exemplos (Vasilev *et al.*, 2019). Regressão e Classificação representam duas áreas que compõem a aprendizagem supervisionada.

Num problema de classificação é utilizado conhecimento previamente adquirido, para a consequente aprendizagem das características dos dados e com o objetivo de prever a categoria discreta de uma determinada amostra (Goodfellow et.al., 2016). Em problemas de regressão também se recorre à utilização de conhecimento prévio, mas incide-se na previsão de valores contínuos.

O objetivo de um problema de classificação supervisionado é análogo ao processo de aprendizagem entre "professor-aluno". Isto é, o aluno resolve um problema definido pelo professor, sendo que o professor é responsável por corrigir frequentemente o trabalho do aluno. Em cada correção, o professor fornece orientações ao aluno, sobre o que está mal e o que deve corrigir.

Esta analogia é muito semelhante ao que acontece na realidade num problema de classificação, onde a aprendizagem consiste na medição do afastamento, entre as previsões do modelo e os *targets* reais das amostras.

São diversos os algoritmos de classificação existentes, entre os quais, as redes neuronais.

4.2 Rede Neuronal – Contextualização Biológica

As redes neuronais representam um algoritmo de classificação supervisionado, que tenta simular o comportamento estimulado pelo cérebro humano (Aggarwal, 2018).

O cérebro humano é constituído por milhares de neurónios que estão organizados em redes muito profundas. Os neurónios comunicam, entre si, através de ligações *axons* e *dendrites* (Coolen, 1998; Aggarwal, 2018), sendo estas ligações encapsuladas numa *sinapse*.

A Figura 8 esquematiza o processo de comunicação entre neurónios, através da descrição visual dos termos citados no parágrafo anterior.

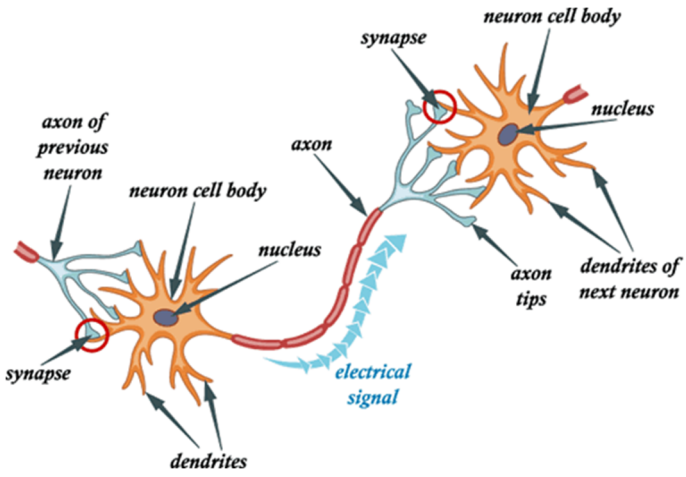

Figura 8 - Comunicação entre neurónios: Fonte: http://andreeasanatomy.blogspot.com/2011/04/you-need-to-step-up-on-step-to-reach_23.html

Segundo Coolen (1998), o conceito de um neurónio (humano) é análogo a um processador lento e simplista. A elevada capacidade de comunicação, entre os neurónios (cada neurónio está ligado a 10^4 neurónios), permite que o cérebro humano seja mais resiliente e poderoso do que os computadores na resolução de problemas reais.

Os neurónios promovem uma aprendizagem contínua, através do envio de impulsos elétricos entre si (Coolen, 1998; Nielsen, 2015; Aggarwal, 2018). A atualização dinâmica dos pesos das conexões sinápticas é um estímulo baseado na resposta obtida a um acontecimento.

4.3 Rede Neuronal – Perceptrão

Frank Rosenblatt (1958) introduziu no início da década de 60 a rede perceptrão. Esta rede simula o comportamento de um neurónio humano, ou seja, transforma um conjunto de entradas (*inputs*) numa saída (*output*). A Figura 9 exemplifica a rede perceptrão.

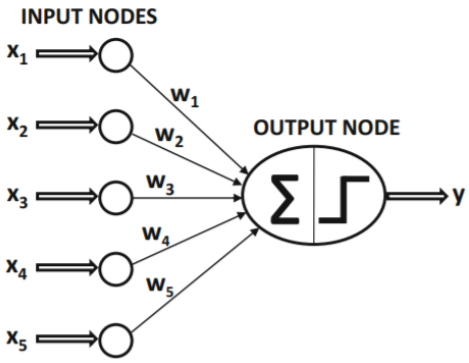

Figura 9 - Rede perceptrão – Fonte: (Nielsen, 2015)

A rede perceptrão é simples, sendo constituída por duas únicas camadas: de *inputs* e de *outputs*. A camada de *inputs* é constituída por um conjunto de nós (*inputs*) que, geralmente, é condizente com o número de *features* do problema (Nielsen, 2015).

A equação 6 traduz o cálculo do *output* da rede perceptrão.

$$output = \begin{cases} 0 \; if \; \sum_j w_j \, x_j \leq threshold \\ 1 \; if \; \sum_j w_j \, x_j > threshold \end{cases} \tag{6}$$

O cálculo do *output* consiste no somatório ponderado entre a multiplicação de cada *input* x_j e o peso da ligação que lhe está associado w_j (Aggarwal, 2018). O retorno do somatório é comparado com a constante *threshold*. A constante *threshold* representa um valor real, responsável por determinar o *output* da rede, ou seja, caso o valor do somatório seja igual ou inferior à constante *threshold*, então o *output* da rede é 0, caso contrário o *output* é 1 (Nielsen, 2015).

A equação 6 pode ser simplificada, para tal, os *inputs* e os pesos da rede devem ser transformados em dois vetores, respetivamente *x* e *W*. A constante *threshold* é renomeada de *bias* (Nielsen, 2015). A equação 7 formaliza a simplificação da equação 6.

$$output = \begin{cases} 0 \; if \; W \cdot x + b \leq 0 \\ 1 \; if \; W \cdot x + b > 0 \end{cases} \tag{7}$$

4.4 Redes MLP

As redes *multilayer perceptrons* (MLP) representam uma aglomeração organizada de um conjunto de perceptrões (Mitchell, 1997). A Figura 10 ilustra a morfologia da rede MLP.

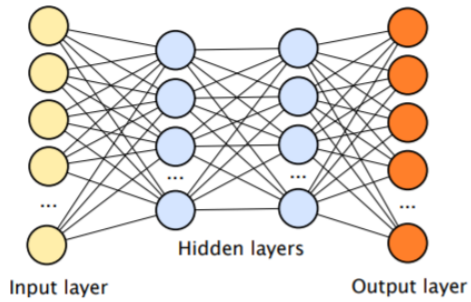

Figura 10 – Rede *Multilayer Perceptron* - Fonte: (Marcus, 2018)

As redes MLP foram introduzidas com o objetivo de tornar viável a resolução de problemas complexos, dado que a utilização das redes *perceptron* é apenas indicada, para a resolução de problemas simples e linearmente separáveis (Mitchell, 1997).

A principal diferença entre uma rede perceptrão e uma rede MLP reside no número de camadas utilizadas (profundidade).

As redes MLP são constituídas no mínimo por três camadas, ao contrário das duas camadas da rede perceptrão. A primeira camada da rede MLP corresponde à camada de *inputs*, que já fora mencionada previamente. As redes MLP são ainda compostas por uma ou várias camadas intermédias – *hidden layer's* e situam-se entre a camada de *inputs* e a camada de *outputs*. O objetivo das camadas intermédias consiste na aprendizagem das características dos dados, através da aprendizagem das relações não lineares que existem entre a camada de *input* e a camada de *output* (Goodfellow et.al., 2016). A camada de *outputs* apresenta a mesma utilidade que fora descrita para a rede perceptrão.

As redes MLP, tal como as redes *perceptron*, consideram a computação dos *outputs* seguindo um único caminho, isto é, movimento *forward*.

O movimento *forward* considera que a informação circula num único sentido, ou seja, é seguido um fluxo que se inicia na camada de *inputs* e termina na camada de *outputs*. Os *outputs* obtidos em cada camada são transmitidos como *input* para a camada seguinte (Nielsen, 2015).

A equação 8 traduz a equação de pré-ativação de um neurónio (Marta, 2016). A equação 9 é uma representação simplificada da equação 8.

$$pre - ativação_{neuron} = w_1 x_1 + w_2 x_2 + w_i x_i + \ldots + b = \sum_{i=0}^{i=N} (w_i x_i) + b \quad (8)$$

$$pre - ativação_{neuron} = W * x + b \quad (9)$$

A equação 10 descreve a função de pós-ativação de um neurónio (*output* de um neurónio). Para tal, é utilizada uma função de ativação (descrita através da notação Φ) que introduz não-linearidade à rede, determinando se o neurónio deve, ou não, ser ativado.

$$output = \Phi(pre - ativação_{neuron}) \quad (10)$$

Seguidamente irão ser contextualizados vários pontos relacionados com o processo de criação, treino e previsão das redes MLP, entre os quais: funções de ativação, conceito de *back-propagation*, algoritmos de otimização dos pesos da rede (baseados em gradiente) e funções de custo.

4.4.1 Funções de Ativação

São diversas as funções de ativação existentes. Cada função de ativação apresenta propriedades distintas e a sua utilização deve ser coerente com as características do problema em resolução (Marta, 2016; Nwankpa *et al.*, 2018).

As seções que se seguem expõem as funções de ativação mais utilizadas atualmente.

4.4.1.1 Identidade

A função identidade simula uma ativação linear, ou seja, o *output* de um neurónio é proporcional ao seu correspondente valor de pré-ativação (Nielsen, 2015). A sua utilização é restrita, sendo tipicamente utilizada em problemas de regressão (Aggarwal, 2018).

4.4.1.2 Sigmóide

A função sigmóide descreve uma ativação não linear que pode ser utilizada, quer na ativação de neurónios (nas camadas escondidas), quer na obtenção de distribuições probabilísticas (problemas binários). Contudo, e dadas as limitações desta função (saturação dos gradientes[3]), atualmente, é apenas utilizada na estimação da probabilidade de ocorrência de um acontecimento binário, ou na modelação de tarefas de regressão logística (Goodfellow et.al., 2016; Nwankpa *et al.*, 2018). A equação 11 traduz matematicamente a função sigmóide.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{11}$$

A Figura 11 ilustra graficamente a função sigmóide.

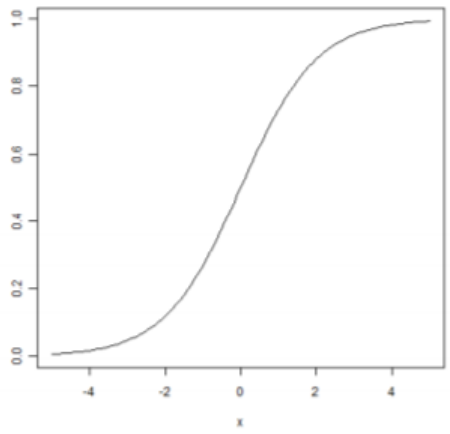

Figura 11 - Função Sigmóide - Fonte: (Heaton, 2012)

4.4.1.3 Tangente Hiperbólica

A tangente hiperbólica é uma função de ativação não linear semelhante à função sigmóide. Contudo, o *output* da função tangente é centrado em zero e estende-se numa gama mais

[3] Quando o valor de pré-ativação de um neurónio é muito baixo ou muito elevado (zonas de saturação da função sigmóide), o seu gradiente aproxima-se de zero, abrandando a capacidade de convergência da rede, já que as alterações efetuadas aos pesos da rede são extremamente pequenas, originando uma tendência de estagnação (Nwankpa *et al.*, 2018).

abrangente, [-1, 1] do que a função sigmóide, [0,1] (visível ao longo do eixo das ordenadas) (Aggarwal, 2018). A Figura 12 ilustra graficamente a função tangente hiperbólica.

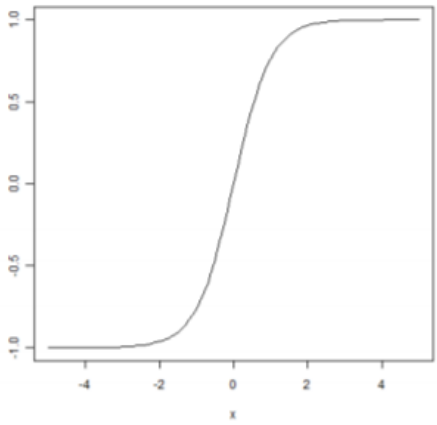

Figura 12 - Função Tangente Hiperbólica - Fonte: (Heaton, 2012)

A equação 12 formaliza matematicamente a função tangente hiperbólica.

$$f(x) = \frac{e^{2x} - 1}{e^{2x} + 1} \tag{12}$$

4.4.1.4 Softmax

A função *softmax* representa uma extensão da função sigmóide, sendo normalmente utilizada na definição de distribuições probabilísticas associadas a problemas multi-classe (Goodfellow et.al., 2016).

O *output* da função *softmax* estima para cada uma das classes do problema, a respetiva probabilidade de representar um determinado acontecimento (soma das probabilidades de cada classe é igual a 1) (Nwankpa *et al.*, 2018). A equação 13 traduz matematicamente a função *softmax*.

$$f(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}} \tag{13}$$

4.4.1.5 ReLU

A função *rectified linear activation* (*ReLU*) é a mais recente de todas as funções de ativação descritas. Tendo sido criada com o objetivo de mitigar a principal limitação das funções sigmóide e tangente hiperbólica, isto é: a saturação dos gradientes (Goodfellow et.al., 2016).

A equação 14 descreve matematicamente a função *ReLU*.

$$f(x) = max\,(0, x) = \begin{cases} x_i, & if\ x_i \geq 0 \\ 0, & if\ x_i < 0 \end{cases} \tag{14}$$

A sua simplicidade garante uma elevada rapidez de cálculo da saída dos neurónios (Goodfellow et.al., 2016). Segundo Aggarwal (2018), a função *ReLU* atenua a saturação dos gradientes, uma vez que para qualquer $x > 0$, o seu gradiente será sempre igual a 1 (a derivada da função *ReLU* é 1, quando $x > 0$) . A Figura 13 esquematiza a função *ReLU*.

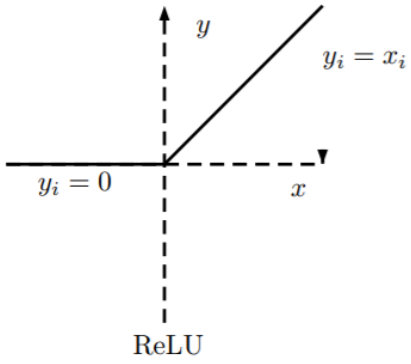

Figura 13 - Função ReLU - Fonte: (Xu *et al.*, 2015)

4.4.2 Backpropagation – Algoritmos de Treino

O cálculo da saída de uma rede MLP segue um movimento *forward*, tal como já fora mencionado anteriormente. Ou seja, a informação segue um fluxo unidirecional, no sentido *inputs* para *outputs*. Mas, a rede necessita de ajustar continuadamente os seus pesos, de modo a garantir que o erro da rede decresce.

O processo de ajuste dos pesos de uma rede neuronal é executado iterativamente, através da execução de passos *backward* ao longo da rede. O passo *backward* descreve uma abordagem contrária do passo *forward*, isto é, o ajuste dos pesos dá-se num sentido único e unidirecional, tendo início na camada de *outputs* e terminando na camada de *inputs* (Heaton, 2012).

O conceito de *backpropagation* refere-se a um processo de treino iterativo, que através da execução de passos *forward* e *backward*, permite a modificação recursiva e camada-a-camada dos pesos da rede (Goodfellow et.al., 2016). Este método é aplicado iterativamente até que uma determinada condição seja atingida, por exemplo: a execução de um número máximo de iterações de treino.

Seguidamente são contextualizados os principais algoritmos de otimização dos pesos da rede, baseados em gradiente e utilizados na minimização do erro da rede.

4.4.2.1 Classic Gradient Descent

O *Classic Gradient Descent* (CGD) descreve a "abordagem clássica" de otimização dos pesos de uma rede neuronal. Genericamente, o CGD altera os pesos da rede através da computação dos seus gradientes (Marta, 2016). Os gradientes determinam a variação do ajuste a aplicar em

cada peso, com vista à minimização do erro da rede (Heaton, 2012). Sendo aplicado um ajuste dos pesos, inverso, à direção dos gradientes calculados (minimização do erro).

A equação 15 traduz o processo de atualização de um peso da rede neuronal (Heaton, 2012).

$$\Delta w_{ij}^{k+1} = \epsilon \frac{\partial E^k}{\partial w_{ij}} + \alpha \Delta w_{ij}^k \quad (15)$$

A constante α simboliza o *momentum*. O uso de *momentum* ajuda a prevenir oscilações que possam ocorrer na atualização dos pesos, através do ajuste de um peso em concordância com a direção dos gradientes calculados anteriormente para esse peso (acumulação exponencial média dos gradientes). Esta abordagem previne que a rede "fique presa" em ótimos locais e acelera a convergência do algoritmo (Goodfellow et.al., 2016; Aggarwal, 2018).

A constante *learning rate* ϵ determina o "passo" de atualização dos pesos da rede e, contrariamente à constante *momentum*, define a importância do gradiente atual de um peso, na estimação do ajuste de um peso (Goodfellow et.al., 2016). Um *learning rate* elevado acelera a convergência do algoritmo, mas dá origem a maiores oscilações e pode afastar-se dos valores ótimos dos pesos da rede. Por outro lado, um *learning rate* baixo aumenta o tempo de convergência da rede e tende a cair facilmente em ótimos locais.

O gradiente individual de um peso w_{ij} é descrito, através da derivada parcial da função de custo com respeito ao peso w_{ij}, sendo assim determinado com recurso à expressão: $\frac{\partial E^k}{\partial w_{ij}}$. A equação 16 traduz matematicamente o cálculo do gradiente individual de um peso w_{ij} (Heaton, 2012).

$$\frac{\partial E^k}{\partial w_{ij}} = \delta_j * o_i \quad (16)$$

o_i traduz o *output* do neurónio *i*. Já δ_j descreve uma constante *node delta* que é calculada para cada neurónio da rede e, à medida que os erros são propagados ao longo da rede (Heaton, 2012).

A equação 17 formaliza o cálculo da constante δ, dado um nó *j*.

$$\delta_j = \begin{cases} -Ef_j' \text{ , nós de saída} \\ f_j' \sum_m w_{mj}\delta_m \text{ , nós interiores} \end{cases} \quad (17)$$

A constante *E* representa o erro do neurónio *j*, que é calculado através da subtração entre o *target* real da amostra e a respetiva previsão (*output*) do neurónio (Aggarwal, 2018). f_j' representa a derivada parcial da função de ativação utilizada pelo neurónio. A segunda expressão $f_j' \sum_m w_{mj}\delta_m$ traduz o somatório entre o produto de cada conexão do nó *j*, com a constante *node delta* δ_m referente ao neurónio *m* que partilha a conexão (peso) com o neurónio *j*, sendo o somatório multiplicado pela derivada parcial da função de ativação.

Este algoritmo atualiza os pesos da rede, assim que a rede terminar a aprendizagem de todas as amostras do problema. Esta abordagem desacelera o processo de convergência da rede, uma

vez que os pesos são atualizados uma única vez por *epoch*. Aumenta ainda as exigências computacionais, devido à aprendizagem simultânea de todas as amostras do problema.

De modo a atenuar estas limitações foram introduzidos outros algoritmos de otimização, baseados em gradiente.

4.4.2.2 Stochastic Gradient Descent

O *Stochastic Gradient Descent* (SGD) é, provavelmente, o algoritmo mais utilizado na otimização dos parâmetros de uma rede neuronal (Goodfellow et.al., 2016).

Esta técnica foi introduzida com o objetivo de minimizar as principais limitações do algoritmo CGD. O CGD é um algoritmo de convergência lenta, uma vez que os pesos da rede são atualizados uma única vez por *epoch*[4], independentemente, do tamanho do *dataset*. Esta limitação requer o uso de um maior número de *epochs*, de modo a assegurar que o erro da rede decresce para os valores pretendidos (Heaton, 2012).

O CGD é muito exigente computacionalmente, especialmente, quando os *datasets* reúnem um elevado número de amostras, uma vez que em cada passo de atualização dos pesos da rede, são armazenados em memória os gradientes estimados para todas as amostras do problema, tornando a sua utilização imprópria na maioria dos casos (Nielsen, 2015).

O SGD introduziu inicialmente o conceito de *online learning*. Esta abordagem atualiza os pesos da rede, logo após a aprendizagem de uma única amostra (processa uma única amostra de cada vez) (Nielsen, 2015). A técnica *online* apresenta assim um procedimento contrário ao CGD, isto é, o número de atualizações efetuadas aos pesos da rede por *epoch*, é igual ao número de amostras do *dataset* (ou seja *batch size* = 1) (Ruder, 2017).

A técnica *online learning* introduz uma flutuação excessiva dos gradientes, tornando a sua estimação pouco realista (Nielsen, 2015; Aggarwal, 2018).

O conceito de *mini-batch* foi introduzido com o objetivo de minimizar este problema. *Mini-batch* considera a utilização de um *batch size* superior a 1 e inferior ao tamanho do *dataset* (sendo normalmente considerados múltiplos de 2). O *batch size* define o número de amostras que irão ser propagadas ao longo da rede, num único passo de atualização dos seus pesos (Nielsen, 2015). Ou seja, o número de atualizações a efetuar aos pesos da rede por *epoch*, é traduzido através da relação: $\frac{número_{amostras}}{batchsize}$.

O uso de um *batch size* inferior ao tamanho do *dataset*, é também partilhado por outros algoritmos de otimização baseados em gradiente, como o *RMSProp* ou o *Adam*.

[4] Uma *epoch* só é concluída, quando todas as amostras de treino forem propagadas ao longo da rede. Por exemplo, se existirem 200 amostras de treino e o *batch size* = 100, os pesos da rede serão atualizados duas vezes numa *epoch* (dois passos de atualização dos pesos da rede, ou duas iterações).

4.4.2.3 RProp

O algoritmo *Resilient Propagation* (RProp) foi introduzido por Riedmiller e Braun (1993) e segundo os autores é um algoritmo que executa uma adaptação direta do "passo" de atualização do peso (*learning rate*), recorrendo a informação local dos gradientes.

O *RProp* não utiliza diretamente os gradientes no processo de atualização dos pesos da rede, para tal, considera unicamente o seu sinal (Riedmiller e Braun, 1993).

A cada peso da rede é associada uma constante "valor atualizado", que determina a variação a impor ao valor atual do peso (Riedmiller e Braun, 1993). As constantes "valor atualizado" são otimizadas ao longo do processo de treino, em concordância com o sinal do gradiente individual dos pesos (Riedmiller e Braun, 1993).

A equação 18 traduz o processo de atualização da constante "valor atualizado", de um determinado peso w_{ij}. O produto entre o gradiente atual do peso w_{ij} e o gradiente obtido na *epoch* anterior: $\frac{\partial E^{(t)}}{\partial w_{ij}} * \frac{\partial E^{(t-1)}}{\partial w_{ij}}$, determina se existiu uma alteração do sinal do gradiente (do peso w_{ij}) entre duas *epochs* consecutivas. Caso, não tenha existido uma variação do sinal do gradiente (1ª condição, equação 18), então existe viabilidade para acelerar a convergência do algoritmo. A constante "valor atualizado" é atualizada, considerando a sua multiplicação pela constante *positive eta* η^+.

Por sua vez, se existiu uma variação do sinal do gradiente do peso w_{ij} (2ª condição, equação 18), então a última atualização efetuada ao peso foi muito severa (muito grande), levando à queda do algoritmo para um ótimo local (Riedmiller e Braun, 1993). O reajuste da constante "valor atualizado" é determinado pela constante *negative eta* η^-.

Se, por outro lado, o resultado do produto entre o gradiente atual e o gradiente da *epoch* anterior for zero, então o valor da constante "valor atualizado" deve manter-se inalterado, uma vez que qualquer alteração, pode originar a variação do sinal do gradiente (Riedmiller e Braun, 1993).

As constantes, *positive eta* e *negative eta*, devem respeitar a seguinte condição: $0 < \eta^- < 1 < \eta^+$.

$$\Delta_{ij}^{t} = \begin{cases} \eta^+ * \Delta_{ij}^{(t-1)}, & if \ \frac{\partial E^{(t-1)}}{\partial w_{ij}} * \frac{\partial E^{(t)}}{\partial w_{ij}} > 0 \\ \eta^- * \Delta_{ij}^{(t-1)}, & if \ \frac{\partial E^{(t-1)}}{\partial w_{ij}} * \frac{\partial E^{(t)}}{\partial w_{ij}} < 0 \\ \Delta_{ij}^{(t-1)}, & else \end{cases} \quad (18)$$

Finalmente procede-se à atualização dos pesos da rede, recorrendo à equação 19.

$$\Delta w_{ij}^{t} = \begin{cases} -\Delta_{ij}^{(t)} \ , \ if \ \dfrac{\partial E^{(t)}}{\partial w_{ij}} > 0 \\ +\Delta_{ij}^{(t)} \ , \ if \ \dfrac{\partial E^{(t)}}{\partial w_{ij}} < 0 \\ 0 \ , \ else \end{cases} \quad (19)$$

Caso, o gradiente atual do peso w_{ij} seja superior a 0, então procede-se à redução do valor atual do peso. Caso o gradiente seja inferior a 0, então procede-se ao aumento do valor do peso (Riedmiller e Braun, 1993).

4.4.2.4 RMSProp

O algoritmo *RMSProp* foi introduzido por Tieleman e Hinton (2012). *Hinton* criou o *RMSProp* com o objetivo de dar resposta a uma limitação do algoritmo *Adaptative Gradient Algorithm* (*AdaGrad*).

O *AdaGrad* procura aumentar a rapidez de convergência da rede e reduzir as oscilações que ocorrem na atualização dos seus pesos, em comparação com o algoritmo SGD. Para tal, considera que cada peso têm o seu próprio *learning rate*, sendo esta constante atualizada dinamicamente e em conformidade com o respetivo gradiente do peso a que está associada. O *Adagrad* visa garantir que os pesos com gradientes elevados são corrigidos, através de um rápido decréscimo do seu *learning rate*, por outro lado, pesos com gradientes baixos devem diminuir ligeiramente o valor do seu *learning rate* (Goodfellow et.al., 2016).

O *AdaGrad* escala os *learning rates*, considerando a divisão entre o gradiente atual e a raiz quadrada da soma dos gradientes quadráticos, calculados anteriormente para esse peso, ou seja, à medida que as iterações vão sendo executadas, a soma dos gradientes quadráticos também aumenta, originando assim um rápido decréscimo dos *learning rates*, o que dificulta a convergência da rede (algoritmo torna-se lento precocemente) (Aggarwal, 2018).

O *RMSProp* corrige este problema, através do cálculo da média exponencial dos gradientes quadráticos (prévios) de um peso, ao invés do cálculo da soma (Tieleman e Hinton, 2012).

A equação 20 atualiza a média exponencial dos gradientes quadráticos (A_i) de um peso w_i. $\rho \ \epsilon \ (0,1)$ representa um fator de ponderação, que é responsável por atribuir um peso distinto, entre os gradientes calculados previamente (iterações anteriores) e o gradiente atual (Tieleman e Hinton, 2012).

$$A_i = \rho A_i + (1-\rho)\left(\frac{\partial L}{\partial w_i}\right)^2 \quad (20)$$

A equação 21 atualiza um determinado peso da rede. A constante α simboliza o *global learning rate* (valor fixo, definido inicialmente), sendo esta constante dividida pela raiz quadrada da média exponencial dos gradientes quadráticos (prévios) do peso. Ou seja, o *learning rate* individual de cada peso é atualizado, através da operação: $\frac{\alpha}{\sqrt{A_i}}\left(\frac{\partial L}{\partial w_i}\right)$.

$$w_i = w_i - \frac{\alpha}{\sqrt{A_i}} \left(\frac{\partial L}{\partial w_i}\right) \quad (21)$$

4.4.2.5 Adam

O método *Adaptive Moment Estimation* (*Adam*) foi introduzido por Kingma e Ba (2017) e segundo os autores: "executa os *learning rates* individuais e adaptativos de cada peso da rede, através da estimação do 1º e 2º momento dos gradientes".

Esta técnica estima dois momentos distintos dos gradientes, isto é, para além de estimar a média exponencial dos gradientes quadráticos de um peso (1º momento), estima, ainda, a variação não centrada dos gradientes (2º momento). As equações 20 e 22 traduzem respetivamente a estimação do 1º e do 2º momento dos gradientes.

$$F_i = \rho_F F_i + (1 - \rho_F) \frac{\partial L}{\partial w_i} \quad (22)$$

O cálculo da média exponencial, dos gradientes quadráticos de um peso, mantém-se inalterado, face ao que fora descrito para o algoritmo *RMSProp*. O 2º momento estima a variação não centrada do gradiente, ou seja, introduz o conceito de *momentum* para consequente melhoria da convergência da rede (Aggarwal, 2018). A constante ρ_F apresenta um objetivo semelhante à constante ρ, sendo os seus valores padrão, respetivamente: 0.9 e 0.99 (Kingma e Ba, 2017).

O ato de estimação dos dois momentos de um gradiente introduz uma limitação que deve ser corrigida, isto é, numa fase inicial do treino da rede, *Kingma* evidenciou uma ligeira tendenciosidade no cálculo de ambos os momentos do gradiente de um peso. Esta limitação advém do facto, dos vetores A_i e F_i serem inicializados em 0, o que, em conformidade com os valores iniciais das constantes ρ e ρ_F (próximos de 1), resulta na estimação de valores próximos de zero nas primeiras iterações, para os dois momentos do gradiente (Kingma e Ba, 2017).

Kingma introduziu assim duas novas equações que corrigem a tendenciosidade evidenciada no cálculo do 1º e 2º momento dos gradientes de um peso. As equações 23 e 24 descrevem respetivamente a correção do 1º e do 2º momento dos gradientes. *t* representa a atual iteração de treino.

$$\hat{A}_i = \frac{A_i}{1 - \rho^t} \quad (23)$$

$$\hat{F}_i = \frac{F_i}{1 - \rho_f^t} \quad (24)$$

Finalmente, os pesos são atualizados através da computação da equação 25. α simboliza a constante *global learning rate* (valor fixo, definido inicialmente). O *learning rate* individual de cada peso é atualizado, através da operação: $\frac{\alpha}{\sqrt{\hat{A}_i + \epsilon}} \hat{F}_i$.

$$w_i = w_i - \frac{\alpha}{\sqrt{\hat{A}_i} + \epsilon} \hat{F}_i \quad (25)$$

O *Adam* é um algoritmo muito eficiente e promove uma elevada rapidez de convergência, sendo um dos algoritmos mais robustos e utilizados atualmente (Goodfellow et.al., 2016; Kingma e Ba, 2017).

4.4.3 Função de Custo

O processo de treino de uma rede neuronal consiste num problema de otimização, que visa identificar os pesos da rede que mais se aproximam de um erro ideal. O erro da rede é calculado com recurso a uma função de custo.

Segundo Marta (2016), uma função de custo (problema de classificação) é matematicamente traduzida através da equação 26.

$$loss = \Psi\,(real_{output}, previsão_{output}) \quad (26)$$

$real_{output}$ descreve um vetor que é composto pelos *targets* reais do conjunto de amostras. $predicted_{output}$ consiste num vetor que agrega as previsões, obtidas ao longo do processo de treino do modelo.

Seguidamente, são contextualizadas as principais funções de custo utilizadas na resolução de problemas de classificação.

4.4.3.1 Cross-Entropy

Cross-Entropy quantifica a distância entre duas distribuições probabilísticas, mais precisamente entre os *targets* reais e as previsões do modelo (Nielsen, 2015). Esta função é robusta, sendo aplicada em diferentes contextos, nomeadamente: resolução de problemas de classificação binários ou multi-classe.

A equação 27 traduz a função *multi-class cross-entropy* (Vasilev *et al.*, 2019).

$$H(p,q) = -\sum_{i=1}^{n} p_i(x)\, log(q_i(x)) \quad (27)$$

Resumidamente, a equação 27 traduz o erro associado à previsão de uma amostra q_i, tendo em consideração o seu *target* real p_i. O erro de previsão de uma amostra, resulta do somatório entre a diferença da probabilidade estimada para cada classe e o seu respetivo *target* (Vasilev *et al.*, 2019). O resultado indica o quão "longe" está o modelo de prever corretamente a amostra (ideal, custo = 0).

O vetor, que contém os *targets* das amostras do problema, deve ser submetido ao processo *one-hot-encoding*. Esta abordagem transforma o *target* de uma amostra num vetor com *n* posições,

sendo *n* o número de classes do problema. Apenas, a posição referente à classe à qual pertence a amostra (classe correta) é traduzida pelo valor 1, enquanto, que as restantes posições são representadas pelo valor 0 (classes incorretas).

A equação 28 traduz a função *binary cross-entropy* (Vasilev *et al.*, 2019).

$$H(p,q) = -p_i(x)\, log(q_i(x)) + (1 - p_i(x))log(1 - q_i(x)) \quad (28)$$

A função *binary cross-entropy* é utilizada na resolução de problemas de classificação binários. No cálculo do erro assume-se uma dependência direta, entre as duas classes (Nasr et.al., 2002).

4.4.3.2 Kullback Leibler Divergence

Kullback Leibler Divergence, ou *KL divergence*, é uma medida que afere a quantidade de informação "perdida" quando se aproximam duas funções probabilísticas (Vasilev *et al.*, 2019). A função *KL divergence*, contrariamente à função *cross-entropy*, não é considerada uma medida de distância, mas, sim, de divergência. A equação 29 formaliza a função *KL divergence*.

$$D_{KL}(p \parallel q) = \sum_{i=1}^{N} p(x_i) * log\frac{p(x_i)}{q(x_i)} \quad (29)$$

Esta medida calcula a divergência (o quão distantes) entre duas funções probabilísticas p e q (Goodfellow et.al., 2016). A equação 29 estima a divergência das funções probabilísticas p e q, quando se usa q para aproximar p (entropia relativa de p em relação a q).

Esta função não é simétrica, $D_{KL}(p \parallel q) \neq D_{KL}(q \parallel p)$, exigindo assim algum cuidado na sua utilização.

A função *KL divergence* é utilizada maioritariamente na resolução de problemas de classificação ou de inferência *bayesiana*, no entanto, comparando com a função *cross-entropy*, o seu uso é menos comum na resposta a problemas de classificação.

4.4.3.3 Hinge Loss

A função *hinge loss* foi criada com o objetivo de reduzir o número de *mismatchs* obtidos durante a tarefa de previsão, sendo traduzida através da equação 30.

$$loss(y_i, z_i) = max(0, 1 - y_i * z_i) \quad (30)$$

Esta função verifica para cada amostra, se a previsão obtida z_i é condizente com o *target* y_i da amostra. Caso, o resultado do cálculo $1 - y_i * z_i$ seja ≤ 0, então, a previsão está correta e não é atribuída qualquer penalização. Por sua vez, caso a previsão esteja incorreta, ou seja $1 - y_i * z_i > 0$, então, é atribuída uma penalização que corresponde ao valor resultante da operação: $1 - y_i * z_i$. O custo da rede consiste no somatório das penalizações referentes a todos os *mismatchs* obtidos (Aggarwal, 2018).

A função *hinge loss* é tipicamente utilizada na resolução de problemas de classificação binários, contudo pode também ser aplicada em problemas multi-classe. Esta função é utilizada pelo algoritmo *Support Vector Machines*.

4.5 Redes de Convolução

As redes MLP não são robustas e eficientes na resolução de problemas de reconhecimento e classificação de imagens, uma vez que são sensíveis à localização espacial dos pixéis (são sensíveis à deslocação dos objetos), são muito exigentes computacionalmente (elevado número de parâmetros de treino) e propensas a sofrerem de *overfitting* (Gu *et al.*, 2017). Com o objetivo de atenuar estas limitações foram introduzidas as *Convolutional Neural Networks*.

As *Convolutional Neural Networks* remontam ao final da década de 70 e inícios da década de 80. O primeiro passo foi dado por *Fukushima* (1979; 1980) através da formalização da rede *Neocognitron*. Esta rede descreve uma arquitetura, que se baseia na utilização alternada de camadas de convolução e de *pooling*.

Desde então, foram introduzidas inúmeras variantes da arquitetura *Neocognitron*. *LeCun et.al* (1989) aperfeiçoou o trabalho de *Fukushima*, aplicando a técnica *backpropagation* na resolução de um problema de classificação de dígitos. *Weng et.al* (1992) desenvolveu a arquitetura *Cresceptron*, uma variante da arquitetura *Neocognitron* que utiliza e formalizou a técnica *Max-Pooling*. *LeCun* et.al (1998) implementou a arquitetura *LeNet-5*, uma rede convolucional mais complexa e eficiente que demonstrou importância no estudo e investigação de novas abordagens e paradigmas.

O avanço da tecnologia permitiu que o estudo e a investigação das redes CNN se expandisse. *Chellapilla et.al* (2006) treinou uma rede CNN recorrendo ao uso de *graphics processing units* (GPU), demonstrando ganhos computacionais e temporais face à utilização de *central processing units*.

Os maiores avanços tecnológicos foram obtidos na última década. *Alex Krizhevsky et.al* (2012) desenvolveu uma arquitetura (*AlexNet*) manifestamente mais profunda e mais complexa do que a redes desenvolvidas até então, sendo composta por 60 milhões de parâmetros de treino e 8 camadas de profundidade. Os excelentes resultados obtidos destacaram a importância do fator profundidade na melhoria da *performance* da rede.

Desde então, têm surgido arquiteturas cada vez mais profundas e com melhores desempenhos. A rede *VGGNet* é composta por 19 camadas, a *InceptionV3* reúne 48 camadas, a *ResNet* dispõe de 152 camadas e a *DenseNet* possui 264 camadas.

As redes CNN são inspiradas pelo córtex visual dos animais. Segundo Aggarwal (2018), o córtex é composto por células agrupadas em pequenas áreas que são excitadas, de acordo com as características dos objetos espelhados no seu campo visual. Ou seja, diferentes formas, cores ou orientações provocam o estímulo de diferentes áreas do córtex.

Citando Patterson e Gibson (2017): "O objetivo de uma rede CNN consiste na identificação de características de alto nível nos dados, recorrendo ao uso de convoluções".

A Figura 14 descreve a constituição típica de uma rede CNN.

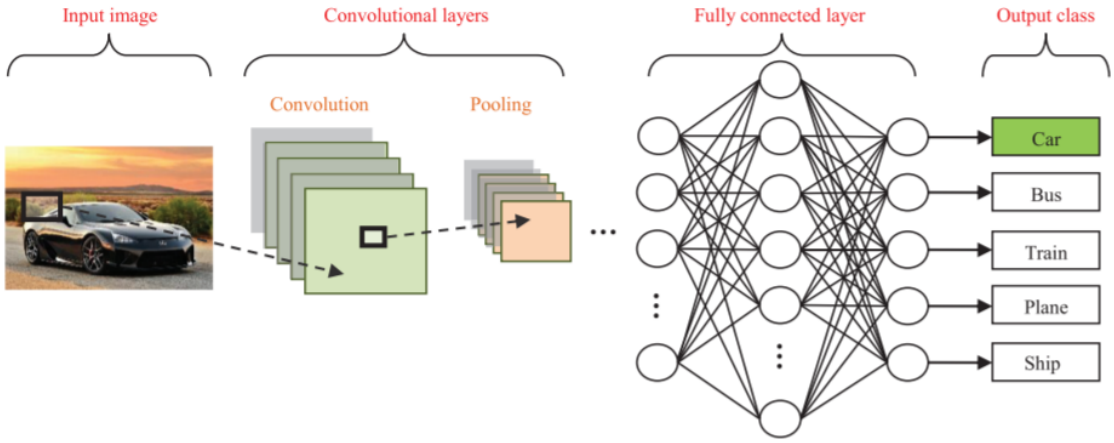

Figura 14 - Arquitetura típica de uma rede CNN – Fonte: (Rawat e Wang, 2017)

Geralmente os dados de entrada (amostras) de uma rede CNN são imagens. Contudo, estas redes podem ser utilizadas na resolução de problemas que interagem com outros tipos de dados, como por exemplo: classificação de séries temporais e processamento de linguagem natural. Dado o âmbito do projeto, daqui em diante, considera-se que os dados de entrada são imagens.

As amostras são representadas num espaço tridimensional: comprimento, largura e profundidade. A profundidade inicial é condizente com o espetro de cores das imagens (Patterson e Gibson, 2017). Uma imagem RGB é composta por três canais de profundidade (*Red, Green e Blue*), enquanto que uma imagem "em tons de cinzento" é composta por um único canal de profundidade.

As redes CNN apresentam várias particularidades que as distinguem das típicas redes MLP. A sua arquitetura é composta por camadas "exclusivas" deste tipo de rede neuronal, nomeadamente: camadas convolucionais e de *pooling*. Uma rede CNN é composta por uma ou mais camadas convolucionais (Rawat e Wang, 2017). As camadas de *pooling* são geralmente intercaladas com as camadas convolucionais.

Citando Aggarwal (2018): "uma convolução consiste na multiplicação matricial entre um conjunto de pesos e um conjunto de *features* provenientes de diferentes localizações espaciais".

Durante o processo de convolução recorre-se à utilização de matrizes de pesos – matrizes *kernel*. O *kernel* (ou filtro) é um detetor de *features* composto por um conjunto de pesos e representado através de um sólido tridimensional, que normalmente apresenta uma relação entre largura e comprimento de: 3*3, 5*5, 7*7 ou 11*11, já a sua profundidade é condizente com a profundidade do *input* da convolução (Aggarwal, 2018). Em cada convolução são utilizados vários *kernels*, que extraem características distintas dos dados.

A Figura 15 ilustra o mecanismo inerente à operação de convolução.

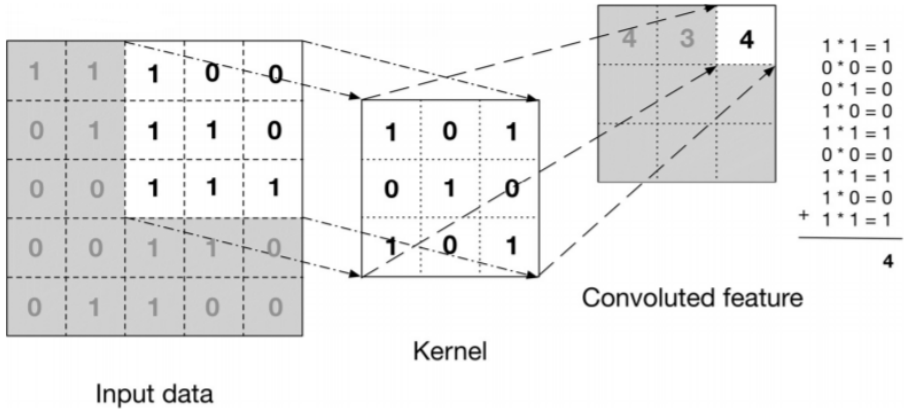

Figura 15 – Operação de convolução – Fonte: (Patterson e Gibson, 2017)

Na Figura 15, o *kernel*, com dimensões 3*3, percorre todos os pixéis da matriz *"input data"*. Na matriz *"input data"* (representa o *input* da convolução) podem ser aplicados um ou mais *kernel's*, sendo que, tendencionalmente, este número aumenta com o aumento da complexidade do problema. Cada *kernel* deteta diferentes tipos de *features*, nomeadamente, linhas, arestas ou formas, ou seja, se forem utilizados 10 *kernels*, o *output* serão 10 *convoluted features* (ou *output features maps*) distintas (Nielsen, 2015).

Cada pixel da matriz *convoluted feature* resulta da aplicação da soma pesada do produto entre a matriz *kernel* e cada conjunto de pixéis, que resultou do "deslizamento progressivo" do *kernel* ao longo da matriz *"input data"* (Patterson e Gibson, 2017). Esta operação percorre todos os pixéis da matriz *"input data"* e os pesos envolvidos na computação, de cada *convoluted feature*, são partilhados por todos os neurónios (*parameter sharing*). Concluída a operação de convolução, procede-se ao uso de uma função de ativação não linear, para consequente ativação, ou não, dos neurónios (aplicação *element-wise*). A função de ativação *ReLU* é atualmente a função padrão utilizada (Vasilev *et al.*, 2019).

A equação 31 determina o número de pesos envolvidos numa operação de convolução.

$$W = (D * F_w * F_h + 1) * M \tag{31}$$

O número de pesos utilizados numa convolução deriva do produto entre: a profundidade do *input* da convolução D (número de *input feature maps*), a largura do *kernel* F_w e o comprimento do *kernel* F_h, sendo ainda adicionado o valor 1, que corresponde ao *bias* da matriz *kernel*, e, por fim, é multiplicado o resultado da operação, $(D * F_w * F_h + 1)$, pelo número M de *kernels* utilizados (valor de M, representa a profundidade do *output* da convolução – número de *output feature maps*).

Existem dois parâmetros que controlam as dimensões (comprimento e largura) do *output* resultante da operação de convolução: o *stride* e o *padding*.

O *stride* define o "passo" de deslocação do *kernel* ao longo do *input* da convolução. Na Figura 15, o valor do *stride* é igual a 1, ou seja, o *kernel* desloca-se ao longo da matriz *"input data"* de 1 em 1 *pixel* (na horizontal e vertical). Contudo, podem ser assumidos outros valores, por exemplo, o uso de um *stride* igual a 2, significa que o *kernel* se move de 2 em 2 unidades ao

longo dos dois eixos. Ou seja, quanto maior o valor do *stride*, menores serão as dimensões das *output feature maps* extraídas (Aggarwal, 2018).

O *padding* é também um hiperparâmetro que permite controlar as dimensões das *output feature maps*. De modo, a garantir que o comprimento e a largura das *convoluted features* se mantém inalteradas, face às dimensões do *input*, recorre-se à inclusão de linhas e colunas com zeros ao longo das bordas do *input*, até que as dimensões das *convoluted features* sejam condizentes com as dimensões do *input* (Aggarwal, 2018). Esta técnica é nomeada de *same padding* (ou *half-padding*). Por sua vez, quando não se considera o uso de *padding*, as dimensões de *input* mantém-se inalteradas, sendo esta abordagem reconhecida por *valid padding* (Aggarwal, 2018).

Tal como referido anteriormente, as camadas convolucionais são geralmente intercaladas com as camadas de *pooling*. A camada de *pooling* é responsável por reduzir as dimensões (comprimento e largura) do *output* obtido nas camadas anteriores, através do uso de técnicas que extraem e sumarizam regiões de neurónios (Rawat e Wang, 2017). O processo de *pooling* também utiliza uma matriz *kernel*, contudo esta matriz não contém pesos e é utilizada, unicamente, para sumarizar a informação das *feature maps*. Os conceitos de *stride* e de *padding* aplicam-se de igual forma nesta camada.

As técnicas de *pooling* mais utilizadas são: *Max Pooling* e *Average Pooling*. A Figura 16 ilustra a aplicação de ambas as técnicas.

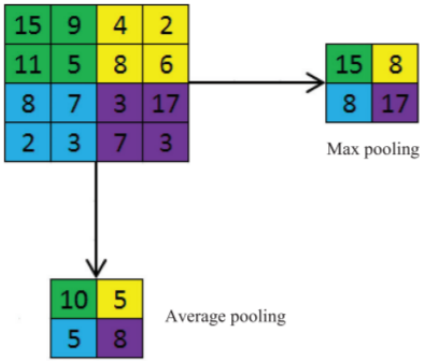

Figura 16 - Aplicação das técnicas *Max Pooling* e *Average Pooling* – Fonte: (Rawat e Wang, 2017)

A técnica *Average Pooling* reduz as dimensões do *input* (das *feature maps*), através da computação do valor médio de cada região sumarizada pelo *kernel*. Já, a técnica *Max Pooling* retém o valor máximo associado a cada região sobreposta pelo *kernel* (Patterson e Gibson, 2017).

As dimensões, padrão, do *kernel* são 2*2 (comprimento e largura) e o valor do *stride* é igual a 2. Esta abordagem permite uma redução do volume do *input*, em cerca de 75%.

Este processo alternado entre as camadas de convulsão e as camadas de *pooling* é repetido n vezes, sendo que, quanto maior o valor de n, maior será a probabilidade de serem identificados padrões complexos nos dados.

Geralmente após a aplicação deste conjunto alternado de camadas, recorre-se à utilização de várias *fully-connected layers* (os neurónios desta camada recebem *input* de todos os neurónios da camada anterior – o uso destas camadas simula uma rede MLP). As *fully-connected layers*

recebem e processam as *features* detetadas nas camadas de convolução. A sua utilização preserva a informação espacial e reduz a abstração das *features* extraídas ao longo das camadas de convolução (Vasilev *et al.*, 2019). A última *full-connected layer* corresponde à camada de *output* (distribuição probabilística).

As arquiteturas convolucionais recentes (*ResNet, DenseNet*) consideram a utilização da camada *Global Average Pooling* (GAP), ao invés das *fully-connected layers*. A camada GAP reduz as dimensões do *input*, através da computação do valor médio de cada *feature map* (Lin et.al., 2014). Ou seja, se o *input* apresentar dimensões: (15,15,10), o *output* resultante da camada GAP terá dimensões: (1,1,10).

A camada GAP evita a utilização de *fully-connected layers*, reduzindo significativamente a complexidade da rede e atenua o risco da rede vir a sofrer de *overfitting*, já que é uma operação que não introduz parâmetros de treino adicionais (Lin et.al., 2014).

4.5.1 LeNet-5

A rede Le-Net5 (Lecun *et al.*, 1998) serviu de inspiração para as arquiteturas convolucionais desenvolvidas recentemente.

A rede é composta por 2 camadas convolucionais que utilizam, respetivamente, 6 e 16 filtros com dimensões 5*5, o valor do *stride* é igual a 1 e não recorrem ao uso de *padding*. Cada convolução é intercalada por uma camada de *pooling*, que adota a técnica *average pooling*, considerando um *kernel* com dimensões 2*2 e *stride* = 2. A função sigmóide é aplicada após cada camada de *pooling*.

A arquitetura é constituída ainda por duas *fully-connected layers* compostas por 120 e 84 neurónios, respetivamente. A camada de *output* (terceira *fully-connected layer*) utiliza a função *Radial Basis Function* na normalização dos *outputs* referentes às classes do problema (Aggarwal, 2018).

A rede *Le-Net5* utilizou "conceitos" que ainda hoje vigoram:

- Utiliza a técnica de normalização *Z-Score*. As amostras são transformadas de acordo com o formato de uma distribuição normal (média 0 e desvio padrão de 1);
- As camadas de *pooling* são intercaladas com as camadas convolucionais, apesar de, atualmente, existirem abordagens que não consideram a utilização de camadas de *pooling* (Springenberg *et al.*, 2015);
- Considera o aumento do número de filtros, à medida que as dimensões do *input* diminuem (à medida que as dimensões do *input* são reduzidas, existe maior viabilidade para aumentar o número de *feature maps* a extrair, já que as exigências computacionais diminuem, em conformidade com a redução das dimensões do *input*).

4.5.2 AlexNet

A rede *AlexNet* foi introduzida por Alex Krizhevsky et.al (2012) catorze anos após a publicação da rede LeNet-5. A arquitetura foi criada no âmbito do desafio *ImageNet Large Scale Visual Recognition Competition* (ILSVRC), que visa promover o desenvolvimento de algoritmos de reconhecimento de objetos e classificação de imagens em larga escala.

A rede *AlexNet* demonstrou resultados superiores face a todos os outros competidores em prova. Krizhevsky introduziu novos conceitos na modelação das redes CNN.

A rede é composta por cinco camadas convolucionais. As duas camadas iniciais representam *single convolution block's* (não existe sobreposição de convoluções, sendo intercaladas por camadas de *pooling*), enquanto que as restantes três camadas formam um *stacked convolution block* (camadas convolucionais sobrepostas).

Ao contrário da rede *Le-Net5*, a ativação do *output* das camadas de convolução realiza-se logo após a convolução (e não, após a camada de *pooling*). Os estudos evidenciados por *Krizhevsky* demonstraram que a função *ReLU* é computacionalmente mais eficiente e promove uma maior rapidez de convergência face à função tangente hiperbólica.

Foi utilizada a técnica de generalização *Local Responsive Normalization* (LRN), logo após a ativação das duas primeiras camadas convolucionais. LRN descreve uma técnica de generalização inspirada em princípios biológicos – conceito de inibição adjacente (Krizhevsky et.al 2012).

As camadas de *pooling* (*max pooling*) são dispostas após a ativação (e da técnica LRN quando aplicada) do *output* referente a cada "bloco" de convolução, isto é, após cada *single convolution block* e após o *stacked convolution block*. *Krizhevsky* constatou que a utilização de um *kernel* de *pooling* superior ao valor do *stride* melhora a capacidade de generalização da rede. Esta técnica é conhecida por *overlapping pooling*.

Finalmente, as *features* extraídas são vectorizadas e distribuídas ao longo de três *Fully-Connected layers*, sendo a camada de *output* traduzida através da terceira e última camada. As duas primeiras *fully-connected layers* são intercaladas pela camada *Dropout* (no capítulo seguinte irá ser contextualizada esta técnica).

A Tabela 1 esquematiza a estrutura e os principais hiperparâmetros da rede *AlexNet*.

Tabela 1 - Constituição da Arquitetura *AlexNet*

Camada	Nº *Feature Maps*	Nº Neurónios	*Kernel Size*	*Stride*
Convolução	96	-	11*11	4
Max Pooling	-	-	3*3	2
Convolução	256	-	5*5	1
Max Pooling	-	-	3*3	2
Convolução	384	-	3*3	1

Convolução	384	-	3*3	1
Convolução	256	-	3*3	1
Max Pooling	-	-	3*3	2
Fully-Connected Layer	-	4096	-	-
Fully-Connected Layer	-	4096	-	-
Output Layer	-	1000	-	-

A rede *AlexNet* é significativamente mais profunda do que a rede *Le-Net5*, mas, ainda assim, superficial quando comparada com as arquiteturas mais recentes.

Krizhevsky utilizou vários conceitos que ainda hoje vigoram, entre os quais:

- Utilização da função de ativação *ReLU;*
- Uso da camada de *pooling*, após a ativação (*ReLU*) do *output* resultante da operação de convolução;
- Utilização da técnica *data augmentation* no treino da rede (técnica contextualizada no capítulo seguinte);
- Introdução da camada *Dropout*;
- Uso do otimizador SGD;
- Introdução de *Local Responsive Normalization* (estimulou a criação da técnica de regularização *Batch Normalization*, contextualizada no capítulo seguinte);

4.5.3 VGGNet

Simonyan e Zisserman (2015) desenvolveram a arquitetura *VGGNet* no âmbito do "desafio" ILSVRC de 2014.

A rede *VGGNet* ficou em 2º lugar na competição apenas atrás da arquitetura *GoogleNet*, ainda assim apresentou melhores resultados na avaliação *single-model*. Apesar de existirem arquiteturas mais recentes e com melhores desempenhos, a rede *VGGNet* é, ainda hoje, uma das arquiteturas mais utilizadas pela comunidade, devido sobretudo à natureza intuitiva da sua estrutura e à sua facilidade de criação.

Esta arquitetura apresenta várias semelhanças com a rede *AlexNet*, no entanto a rede *VGGNet* é mais profunda e demonstra melhor *performance*.

Simonyan desenvolveu várias arquiteturas *VGGNet*, de acordo com diferentes níveis de profundidade, nomeadamente compostas por 11, 13, 16 e 19 camadas. As redes mais densas revelaram melhores resultados (Simonyan e Zisserman, 2015). Daqui em diante, irá ser contextualizada a arquitetura constituída por 16 camadas (rede *VGGNet* padrão).

A rede *VGGNet* é composta unicamente por *stacked convolution blocks* e todas as camadas convolucionais partilham as mesmas características (configuração dos seus hiperparâmetros), isto é, as dimensões dos *kernels* são 3*3, recorrem ao uso de *padding* e o valor do *stride* é igual a 1. A utilização de *kernel's* com dimensões mais reduzidas, aumenta a probabilidade de serem extraídas *features* mais minuciosas e complexas (Aggarwal, 2018).

Simonyan aplicou, de igual modo, a função de ativação *ReLU* após cada convolução. Contudo, não utilizou a técnica de generalização *Local Responsive Normalization*, uma vez que não identificou quaisquer vantagens na sua utilização.

Os *stacked block's* são intercalados por camadas de *pooling* (*max pooling*). Contrariamente à rede *AlexNet*, as dimensões do *kernel* de *pooling* foram reduzidas de 3*3 para 2*2, sendo adotada novamente a estratégia de *LeCun*, *pooling* sem *overlapping*.

Simonyan considerou que cada *stacked block*, extrai o dobro das *feature maps* resultantes (extraídas) do bloco anterior. Esta abordagem atenua a redução do volume espacial do *input*, causado pelas camadas de *pooling*, através da duplicação do número de *feature maps* a extrair no bloco seguinte (aumento da profundidade, em compensação da redução do comprimento e largura, preserva também a complexidade temporal).

As *Fully-Connected layers* apresentam uma abordagem similar, ao que fora descrito para a rede *AlexNet*.

A Tabela 2 descreve a estrutura e a configuração dos hiperparâmetros da arquitetura *VGGNet-16*.

Tabela 2 - Constituição da arquitetura *VGGNet*

Camada	**Nº *Feature Maps***	**Nº Neurónios**	***Kernel Size***	***Stride***
Convolução	64	-	3*3	1
Convolução	64	-	3*3	1
Max Pooling	-	-	2*2	2
Convolução	128	-	3*3	1
Convolução	128	-	3*3	1
Max Pooling	-	-	2*2	2
Convolução	256	-	3*3	1
Convolução	256	-	3*3	1
Convolução	256	-	3*3	1
Max Pooling	-	-	2*2	2
Convolução	512	-	3*3	1
Convolução	512	-	3*3	1
Convolução	512	-	3*3	1
Max Pooling	-	-	2*2	2
Fully-Connected Layer	-	4096	-	-

Fully-Connected Layer	-	4096	-	-
Output Layer	-	1000	-	-

Esta arquitetura utiliza de igual forma as técnicas de regularização, *Data Augmentation* e *Dropout* e ainda o otimizador SGD. Resumidamente, a rede *VGGNet* adoptou as qualidades da rede *AlexNet* e introduziu algumas alterações, tais como: utilização de *kernels* com dimensões reduzidas, uso de *pooling* sem *overlapping*, a morfologia da rede é mais densa e o uso exclusivo de *stacked convolution blocks*.

4.5.4 *ResNet*

A arquitetura *ResNet* foi introduzida por Kaiming He et.al (2015a) no âmbito do desafio ILSVRC de 2015, sendo a arquitetura vencedora em diversos segmentos: classificação, localização e deteção de imagens.

A rede *ResNet* apresentou um novo paradigma de formulação das redes CNN. He definiu o fluxo da rede em forma de grafo, ao invés das abordagens em forma de plano demonstradas anteriormente. Resultando numa rede extremamente profunda e com excelente performance, tornando-se a primeira arquitetura a superar "o humano" na competição ILSVRC.

He constatou que as redes convolucionais mais densas apresentam desempenhos significativamente inferiores às redes superficiais (He *et al.*, 2015a). A rede *ResNet* foi desenvolvida com o objetivo de atenuar esta limitação.

O *residual block* (ou *building block*) é a principal inovação da arquitetura. Este bloco é constituído por duas convoluções sobrepostas, que são intercaladas pela camada *Batch Normalization* e pela função de ativação *ReLU*. O *residual block* é ainda composto por uma "conexão de salto" (*skip connection*) que representa um caminho alternativo para a propagação dos gradientes ao longo da rede (Vasilev *et al.*, 2019). A Figura 17 esquematiza um *residual block*.

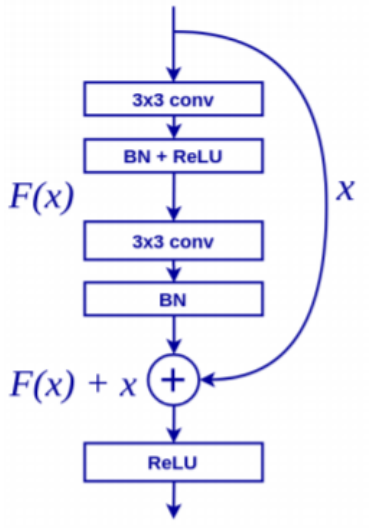

Figura 17 - *Residual Block* - Fonte: (Vasilev *et al.*, 2019)

Na Figura 17, *x* representa uma "conexão de salto" que é responsável por propagar uma cópia do *input* do *residual block* (*output* do *residual block* anterior), enquanto que *F(x)* representa o *output* da computação das camadas de convolução presentes no bloco. O *output* de um *residual block* consiste na adição *element-wise* entre *x* e *F(x)*. Segundo Atienza (2020), a inclusão de *skip connections* previne a degradação dos gradientes, visto que o seu uso facilita a circulação da informação ao longo de toda a rede, especialmente, para as camadas mais superficiais.

As várias "conexões de salto", dispostas ao longo da rede, permitem que o processo de *backpropagation* conte com vários caminhos alternativos (desde a camada de *output* até à camada de *input*), com maior ou menor extensão, na propagação dos gradientes (Aggarwal, 2018). Foi evidenciado, pelos autores, que a adição de *skips connections* melhora a convergência da rede e o seu uso não apresenta qualquer impacto computacional (não introduzem parâmetros de treino adicionais).

A Figura 18 descreve um excerto da rede *ResNet* 18 (rede *ResNet* com menor profundidade).

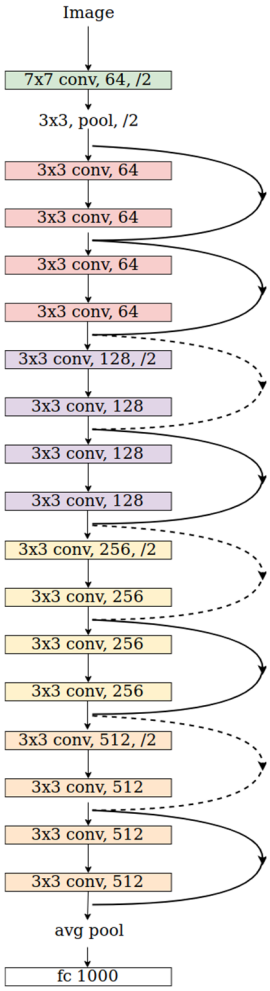

Figura 18 - *ResNet 18* – Fonte: https://www.researchgate.net/figure/Architecture-of-ResNet-18-Figure-from-reference-18_fig1_332303940

A primeira convolução da rede *ResNet* (ilustrada a verde na Figura 18) reduz as dimensões iniciais das amostras, para tal, considera o uso de *padding*, o valor do *stride* é igual a 2 e as *dimensões* do *kernel* são 7*7. A técnica *Max Pooling* é aplicada logo após a convolução (*stride* = 2 e o *kernel* apresenta dimensões 3*3).

Os autores dividiram a rede em conjuntos de *residual blocks*. Cada conjunto é composto por um número *n* de *residual blocks*, sendo estes conjuntos responsáveis por reduzir as dimensões de *input* (*downsampling* realizado pela 1ª camada convolucional, do 1º *residual block* de cada conjunto) e por duplicar o número de *feature maps* resultantes do conjunto anterior. Segundo os autores, a redução das dimensões (comprimento e largura) e a consequente duplicação da profundidade, possibilita o equilíbrio do volume espacial e preserva a complexidade temporal (He *et al.*, 2015a). À exceção do primeiro conjunto (*residual blocks* ilustrados a rosa na Figura 18), todos os outros reduzem as dimensões de *input*. Na Figura 18, os vários conjuntos de *residual blocks* estão ilustrados com cores distintas.

A rede *ResNet* é constituída por dois tipos de *skip connections*: *identity shortcuts* (curvas representadas a negrito na Figura 18) e *projection shortcuts* (curvas representadas a tracejado na Figura 18). As *identity shortcuts* limitam-se a efetuar uma cópia do *input* de um *residual block* e são aplicadas, quando as dimensões (comprimento, largura e profundidade) do *input* do

bloco são iguais ao seu consequente *output*. Por outro lado, as *projection shortcuts* aplicam uma projeção linear sobre a cópia do *input* do *residual block*, e são aplicadas quando existe uma redução das dimensões do *input* (passagem entre conjuntos de *residual blocks*). As *projection shortcuts* ajustam as dimensões (comprimento, largura e profundidade) da cópia, referente ao *input* de um *residual block*, em conformidade com as dimensões de saída do bloco, de modo a possibilitar a adição *element-wise* entre *x* e *F(x)* (Vasilev *et al.*, 2019). A projeção linear consiste numa convolução que faz uso de *padding*, *stride* = 2 e as dimensões do *kernel* são 1*1.

Os autores desenvolveram várias arquiteturas *ResNet*. A Figura 19 ilustra as diferentes densidades da rede.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				

Figura 19 - Diferentes arquiteturas *ResNet* - Fonte: (He *et al.*, 2015a)

O *residual block* ilustrado na Figura 17 é utilizado apenas pelas redes *ResNet* menos densas, isto é, compostas por 18 e 34 camadas. O número de *residual blocks*, que estão agregados a cada conjunto de *residual blocks* é variável (com exceção da rede menos densa – *ResNet*-18).

As redes mais densas consideram o desenho dos *residual blocks* em forma de *bottleneck*. Nestas abordagens, o *residual block* é composto por três camadas convolucionais, ao invés das duas camadas demonstradas previamente (Figura 17). A convolução 1*1 (dimensões do *kernel*) é responsável por reduzir a profundidade do *input* do *residual block* (reduz número de *feature maps*). A convolução 3*3 extrai, assim, *features* num volume espacial mais reduzido, uma vez que a convolução anterior reduziu a profundidade do *input*, atenuando as exigências temporais e computacionais. A profundidade inicial do *residual block* é restabelecida pela última convolução (1*1). *He* evidenciou que o uso de *residual blocks* em forma de *bottleneck*, propiciam o aumento da profundidade da rede sem sobrecarga da sua complexidade (o número de parâmetros de treino da rede *ResNet-34* é idêntico à rede *ResNet-50*).

Por último, a rede é composta por uma camada *Global Average Pooling* que é responsável por reduzir as dimensões do *output* resultante da aplicação dos *residual blocks*, sendo sucedida pela camada de *output*.

A rede *ResNet* utiliza o otimizador SGD, recorre a *Data Augmentation*, aplica regularização em todas as camadas de convolução e não usa *dropout*.

4.5.5 DenseNet

A rede *DenseNet* foi introduzida por Huang et.al (2018) e a sua implementação não está relacionada com o desafio ILSVRC, ao contrário das arquiteturas analisadas previamente. Segundo Huang, a rede *DenseNet* foi criada com o intuito de promover uma propagação mais eficiente dos gradientes, face à rede *ResNet*.

A arquitetura *DenseNet* considera, de igual forma, o uso de *skip connections*, contudo introduz várias modificações, quer na estrutura da rede, quer no mecanismo de comunicação entre camadas. A rede demonstrou um excelente desempenho, superando a arquitetura *ResNet*.

Huang definiu um novo fluxo de passagem de informação entre camadas, propondo que existisse um mecanismo de comunicação direta entre todas as camadas (dentro do mesmo *dense block*). Esta abordagem facilita o processo de propagação dos gradientes ao longo da rede, garante uma maior eficiência na reutilização do conhecimento prévio (*feature reuse*) e reduz o poder computacional exigido (Huang *et al.*, 2018).

A rede *Densenet* é constituída por um conjunto de *dense blocks*. Cada bloco é composto por um conjunto de *composite's blocks*, que partilham entre si o mesmo propósito, a extração e a distribuição de conhecimento ao longo da rede. Um *composite block* H_l, segundo os autores, consiste num conjunto de três operações consecutivas: *Batch Normalization, ReLU* e convolução (*kernel* com dimensões 3*3, *stride*=1 e uso de *padding*).

Um *composite block* recebe informação de todos os *composites blocks* que lhe precedem no *dense block*. O *input* de um *composite block l* consiste na concatenação das *feature maps* extraídas por todos os *composite blocks* que lhe precedem no *dense block* $(x_0, x_1, \ldots, x_{l-1})$. A rede *DenseNet* apresenta, assim, uma abordagem distinta da rede *ResNet*. A equação 32 traduz o *input* de um *composite block l*.

$$x_l = H_l([x_0, x_1, \ldots, x_{l-1}]) \tag{32}$$

A rede *DenseNet* ao aplicar a concatenação em detrimento da adição, garante que a informação extraída pelos *composite blocks* é preservada e dissipada ao longo da rede (Huang *et al.*, 2018).

Cada *composite block* é responsável por adicionar um conjunto de *k feature maps* à rede. A constante *k* é descrita pelos autores como: *growth rate*. Segundo *Huang*, a constante *growth rate* regula a quantidade de "nova informação" que é introduzida por um *composite block*. *Huang* evidenciou, ainda, que um baixo valor de *k* é suficiente para obter excelentes desempenhos, nomeadamente $k = 12$ ou $k = 24$. A reutilização de conhecimento reduz a necessidade de aumentar a complexidade da rede (evita o uso de valores de *k* elevados), uma vez que toda a informação extraída ao longo da rede é preservada e distribuída para as camadas seguintes, o que evita a extração de *features* redundantes (Huang *et al.*, 2018; Vasilev *et al.*, 2019).

A equação 33 determina a profundidade do *input* de um *composite block l* (número de *feature maps*) (Atienza, 2020). k_0 representa a profundidade do *input* do *dense block* e l o índice do *composite block* (no interior do *dense block*).

$$inputFeatureMaps_l = k_0 + k * (l - 1) \quad (33)$$

A Figura 20 esquematiza um *dense block*.

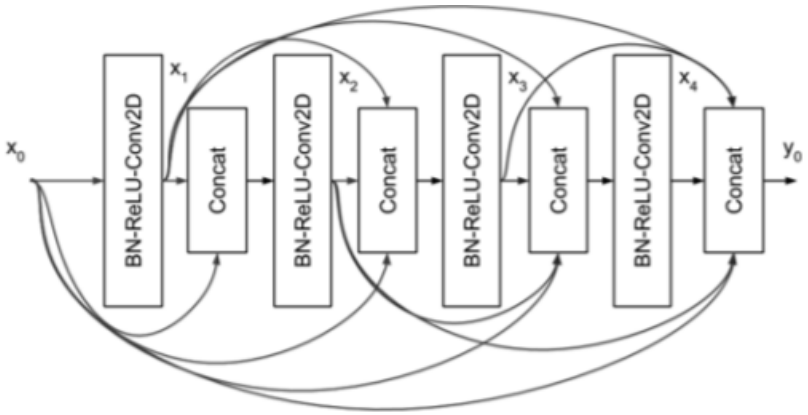

Figura 20 - *Dense Block* - Fonte: (Atienza, 2020)

Os autores evidenciaram a necessidade de inclusão de *composite blocks*, em forma de *bottleneck*, com o objetivo de reduzir o poder computacional exigido. Para tal, recorreram à utilização de uma convolução 1*1 (dimensões do *kernel*) imediatamente antes da convolução 3*3. Ambas as convoluções são precedidas pela operação *Batch Normalization* e pela função de ativação *ReLU*. Segundo *Huang*, a convolução 1*1 deve reduzir a profundidade do *input* para 4*k feature maps*.

A utilização de *bottleneck's* reduz o número de *feature maps* a serem processadas pela convolução 3*3. Ou seja, ao invés de serem processadas as *features maps* resultantes do cálculo da equação 33, são consideradas apenas *4k feature maps*. Esta abordagem melhora a eficiência computacional da rede, que pode ser rapidamente limitada pelo consequente aumento da sua profundidade (Atienza, 2020). Esta arquitetura em forma de *bottleneck* é conhecida por *DenseNet-B*.

Todos os *dense blocks*, à exceção do último, são intercalados por um *transition block*. Este bloco é responsável por reduzir as dimensões (comprimento e largura) resultantes do *output* de um *dense block*. O *transition block* é constituído por uma convolução 1*1 (*stride* = 1 e uso de *padding*) e ainda por uma camada de *pooling* (técnica *average pooling*, *stride* = 2 e *kernel* com dimensões 2*2). A Figura 21 ilustra a utilização alternada de ambos os blocos.

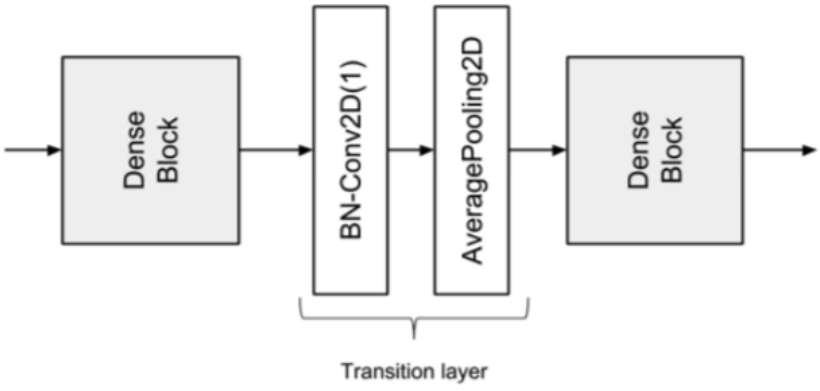

Figura 21 - Utilização alternada dos blocos *Dense e Transition* – Fonte: (Atienza, 2020)

O *transition block* pode ainda ser utilizado na redução da profundidade do *output* de um *dense block*, sendo esta redução controlada pela constante *compression rate* $0 < \theta \leq 1$. Se $\theta = 1$, a profundidade do *output* de um *dense block* mantém-se constante, por sua vez, um $\theta = 0.5$ reduz a profundidade em 50%. Quando a constante $\theta < 1$, a rede *DenseNet* é nomeada de *DenseNet-C* (quando, não se considera o uso de *composite blocks* em forma de *bottleneck*) ou *DenseNet-BC* (quando, se considera o uso de *composite blocks* em forma de *bottleneck*).

A camada *Global Average Pooling* é aplicada após o último *dense block*, sendo sucedida pela camada de *output*.

4.5.6 Outras Arquiteturas Convolucionais

Para além das arquiteturas descritas previamente, existem outras redes que por razões diversas têm despertado o interesse da comunidade.

Ano após ano, surge um número significativo de novas arquiteturas convolucionais. O avanço tecnológico, aliado à constante introdução de novos conceitos e paradigmas, permite que a utilização destas redes seja difundida na resposta a diferentes propósitos.

Atualmente, existem arquiteturas convolucionais ajustadas para a resolução dos mais variados tipos de problemas e enquadradas em diversos tipos de ambientes operacionais. Destacam-se as duas seguintes arquiteturas:

- **Redes *Inception***: A rede *Inception* apresenta um conceito inovador, sendo a par das arquiteturas descritas previamente, uma das redes mais utilizadas pela comunidade. *Szegedy et.al* (2014) reconheceu que a *performance* de uma rede convolucional aumenta, quando esta tem a flexibilidade de aprender recorrendo à extração de *features* com diferentes granularidades (Aggarwal, 2018), uma vez que as *features* extraídas correspondem a padrões com complexidade e tamanho espacial distinto (*kernel's* com dimensões distintas detetam diferentes objetos, isto é, com maior ou menor escala). Este paradigma é formulado através de um *inception block*. Um *Inception block* define vários "caminhos" convolucionais que são executados paralelamente e extraem características com diferentes granularidades dos dados. O *output* deste bloco corresponde à concatenação das *features* extraídas pelos diferentes "caminhos". Esta arquitetura pode tornar-se muito complexa, devido ao elevado número de convoluções presentes em cada *inception block*. De modo, a atenuar esta limitação foram introduzidas convoluções 1*1 (*bottleneck*) antes de cada convolução. Estas convoluções reduzem o número de *feature maps* a serem processadas pelas convoluções com *kernels* mais elevados, melhorando assim a eficiência computacional. A Figura 22 ilustra a arquitetura de um *inception block*.

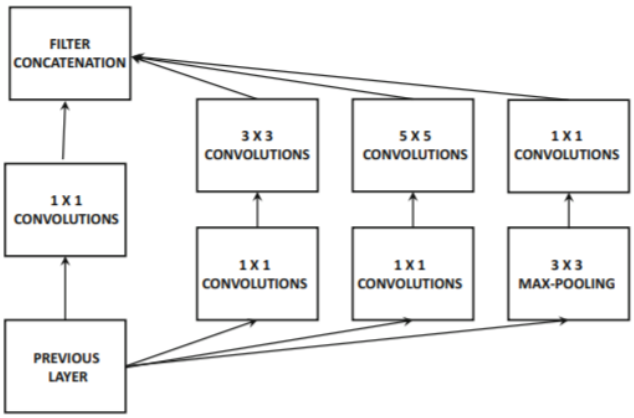

Figura 22 - *Inception block* referente à rede *GoogleNet* - Fonte: (Aggarwal, 2018)

- **Rede *MobileNet***: A rede *MobileNet* (Howard *et al.*, 2017) permite a criação de modelos convolucionais eficientes na resposta a problemas que exigem rápida resolução e/ou requerem baixa complexidade. Esta rede foi pensada e é normalmente utilizada em ambientes limitados, nomeadamente: sistemas embebidos ou *smartphones*. O *hardware* específico destes aparelhos requer a utilização de arquiteturas CNN que promovam rápida computação e requisitem baixo poder computacional. As vantagens da rede *MobileNet* são obtidas recorrendo ao uso de *Depthwise Separable Convolutions*. Estas camadas reformulam o conceito de convolução, ou seja, o *output* de uma convolução requer a execução de dois passos, em detrimento do único passo habitual. Numa primeira fase e contrariamente às convoluções padrão, cada *kernel* da *Depthwise Convolution* é responsável por percorrer um único canal de profundidade do *input*, resultando assim num *output* que apresenta o mesmo volume do *input* (número de *kernels* utilizados = número de *input feature maps*). Finalmente, as *pointwise convolutions* (convoluções 1*1) são responsáveis por criar uma combinação linear do *output*, resultante da aplicação da *depthwise convolution*. Esta abordagem permite a criação de modelos eficientes, com pouca complexidade (baixo número de parâmetros de treino) e facilmente adaptáveis aos mais variados tipos de ambiente. A Figura 23 descreve a estrutura da *Depthwise Separable Convolution*.

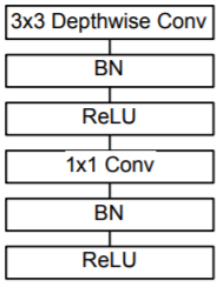

Figura 23 - Estrutura de uma *Depthwise Separable Convolution* – Fonte: (Howard *et al.*, 2017)

CAPÍTULO 5
TÉCNICAS DE REGULARIZAÇÃO DE REDES NEURONAIS

As redes CNN são muito propícias a sofrerem de *overfitting*, que consiste na baixa generalização de um modelo, isto é, a rede limita-se a memorizar os dados utilizados durante o processo de treino, não conseguindo generalizar corretamente a aprendizagem para novos casos (Aggarwal, 2018). Ou seja, pequenas alterações nos dados provocam comportamentos distintos no modelo (elevada variância). Esta limitação causa uma redução drástica do desempenho dos modelos.

Seguidamente, são descritas algumas técnicas que atenuam esta limitação.

5.1 Normas de Regularização – *L1* e *L2*

Uma das técnicas habituais de redução de *overfitting* consiste na regularização dos parâmetros de treino das várias camadas de uma rede CNN. Redes compostas por pesos elevados são mais instáveis e apresentam maior variância, comparativamente às redes compostas por pesos baixos (próximos de 0) (Aggarwal, 2018). De modo, a garantir que os pesos se mantêm baixos, considera-se a utilização de técnicas de penalização dos pesos da rede. Estas abordagens introduzem uma constante na função de custo que é responsável por regularizar os pesos da rede (Vasilev *et al.*, 2019).

Segundo Goodfellow et.al (2016), a função de custo regularizada é definida em concordância com a equação 34.

$$loss_{regularized} = \Psi\,(real_{output}, predicted_{output}) + \alpha\Omega(\theta) \qquad (34)$$

A equação 34 adiciona unicamente uma norma de penalização $\Omega(\theta)$ à equação 26. α define o contributo da norma Ω, em conformidade com o objetivo da função de custo θ. Quanto maior o valor de α, maior será a regularização exercida.

As normas *L1 e L2* descrevem duas normas distintas de penalização dos pesos da rede. O objetivo de ambas as normas, é comum, garantir que os pesos da rede se aproximam de 0.

A norma *L1* considera $\Omega(\theta) = ||w||_1 = \sum_{i=1}^{n} |wi|$. Ou seja, à função de custo é adicionado o somatório dos valores absolutos dos pesos individuais (Goodfellow et.al., 2016).

Por sua vez, a norma *L2* define $\Omega(\theta) = ||w||_2^2 = \sum_{i=1}^{n} wi^2$. Isto é, adiciona à função de custo o somatório quadrático dos pesos individuais (Goodfellow et.al., 2016; Aggarwal, 2018).

A norma *L2* permite que pesos elevados decaiam mais rapidamente, já que o decréscimo do valor de cada peso é proporcional ao seu valor atual. Por sua vez, a norma *L1* garante que os

pesos se aproximam mais rapidamente de 0, já que a norma *L2* tende a desacelerar a redução dos valores dos pesos, à medida que estes se tornam mais pequenos (Goodfellow et.al., 2016).

5.2 Dropout

Dropout é provavelmente a técnica de regularização mais utilizada. Esta técnica foi inicialmente introduzida por *Hinton et.al* (2014).

A sua utilização visa a melhoria da capacidade de generalização de um modelo, através da modificação da rede ao longo do processo de treino. Para tal, remove de forma aleatória e periodicamente, uma percentagem dos neurónios presentes numa determinada camada da rede (os neurónios são apenas removidos temporariamente, continuam a existir) (Aggarwal, 2018).

A constante $dropout_{rate}$ define a probabilidade de um neurónio ser removido da rede, sendo que $0 < dropout_{rate} < 1$. Ou seja, caso um neurónio seja removido da rede (*output* do neurónio passa a ser 0), todas as suas ligações com as restantes camadas são também removidas. A Figura 24 ilustra a aplicação da técnica *dropout*.

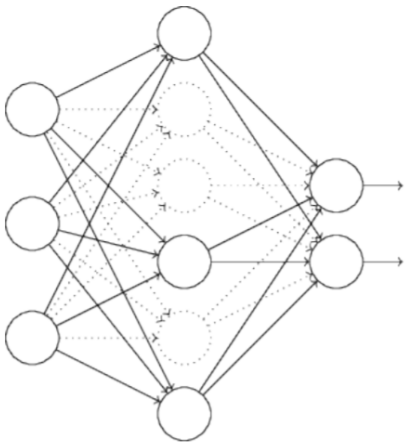

Figura 24 - *Dropout* - Fonte: (Nielsen, 2015)

Esta técnica é aplicada ao longo do processo de treino, isto é, para cada *mini-batch* são criadas várias sub-redes (uma sub-rede por amostra, cada sub-rede remove aleatoriamente um conjunto de neurónios), sendo estimados os gradientes e atualizados os parâmetros da rede (Nielsen, 2015). Este processo ocorre de forma iterativa ao longo das *epochs*. *Hinton* constatou que a técnica não se revela particularmente exigente, uma vez que as várias "sub-redes" criadas, partilham parâmetros entre si (*parameter sharing*).

A técnica *dropout* atenua o problema de *overfitting*, uma vez que reduz a dependência entre neurónios. Ou seja, ao longo das várias *epochs* (e consequentes *batches*) a rede é confrontada com diferentes cenários (diferentes sub-redes), permitindo uma adaptação e aprendizagem mais eficiente (Aggarwal, 2018).

5.3 Batch Normalization

Batch Normalization é uma técnica de regularização que visa atenuar o problema *internal covariate shift*.

Internal covariate shift é um problema que surge, naturalmente, ao longo do processo de treino da rede. Os parâmetros da rede são atualizados ao longo das *epochs*, o que implica que ao fim de algumas *epochs*, a distribuição das entradas das camadas escondidas se altere (Aggarwal, 2018). Estas alterações exigem uma reaprendizagem da distribuição, dificultando e desacelerando a convergência da rede (Vasilev *et al.*, 2019).

Dessa forma, Ioffe e Szegedy (2015) introduziram a técnica *Batch Normalization* com vista à melhoria do desempenho dos modelos e ao aumento da sua rapidez de convergência.

A técnica *Batch Normalization* propõe a utilização de camadas de normalização entre as camadas convolucionais e/ou as *Fully-connected-layers*. As entradas de uma determinada camada são normalizadas, de modo a que a sua média seja próxima de 0 e o desvio padrão próximo de 1 (em concordância com a distribuição das *batches*). Estas constantes correções das ativações melhoram o desempenho das redes.

Segundo Bjorck et.al (2018), esta técnica torna possível a utilização de *learning rates* mais elevados, o que, por sua vez, acelera o processo de convergência da rede. A sua utilização previne ainda que os gradientes estejam dependentes das variações de escala dos parâmetros (evita que os gradientes se tornem muitos pequenos ou muito grandes) (Aggarwal, 2018).

5.4 Early Stopping

Early Stopping é uma abordagem de regularização intuitiva e simples de aplicar. Esta técnica controla o progresso de treino da rede, isto é, automaticamente e ao fim de um determinado número de *epochs*, verifica se o modelo apresenta indícios de *overfitting*, caso existam, dá por concluída a fase de treino. Tenta assim garantir que não se desperdiça tempo e recursos computacionais, quando já não existe viabilidade para o modelo melhorar a sua *performance* (Goodfellow et.al., 2016).

A monitorização consiste na avaliação do processo treino do modelo recorrendo a várias métricas, entre as quais: erro de treino, erro de validação, *accuracy* de treino, *accuracy* de validação, entre outras.

Geralmente é inspecionada a variação do erro de validação, através da sua comparação com o erro de treino. Caso, o erro de validação ou de treino não decresça ao fim de um determinado número de *epochs*, então o modelo apresenta indícios de *overfitting*, sendo concluída a fase de treino.

A utilização desta técnica é geralmente combinada com a diminuição do hiperparâmetro *learning rate* (caso, sejam utilizados algoritmos de treino com *learning rates* adaptativos, é

diminuído o valor da constante *global learning rate*). Isto é, de modo a evitar uma conclusão precoce do processo de treino da rede, verifica-se preliminarmente o comportamento do modelo, após a diminuição do valor da constante *learning rate* (Goodfellow et.al., 2016). A redução do *learning rate* (normalmente, divisão do valor atual por um valor compreendido entre [2 - 10]) permite a realização de ajustes mais levianos nos parâmetros da rede, o que, em muitos casos, garante saídas de ótimos locais e atenua problemas como *overshooting* (ajustes severos nos parâmetros da rede).

5.5 Técnicas de inicialização dos pesos da rede

O processo de inicialização dos pesos de uma rede CNN tem suscitado vasta investigação ao longo dos últimos anos. De tal forma, que na última década foram introduzidas diferentes abordagens (Glorot e Bengio, 2010; Sutskever *et al.*, 2013; Saxe et.al., 2014; He *et al.*, 2015b), todas partilhando um objetivo comum, a melhoria da capacidade de generalização e de convergência das redes neuronais.

A correta inicialização dos pesos da rede, é um fator preponderante e que afeta expressivamente o seu desempenho. Caso os pesos iniciais da rede apresentem valores muito baixos, os gradientes serão de tal forma pequenos que a convergência da rede será muito lenta, por outro lado, caso os pesos sejam inicializados com valores muito grandes, então o custo irá oscilar severamente em torno do valor mínimo (divergência) (Goodfellow et.al., 2016). Ou seja, é necessário inicializar adequadamente os pesos da rede, de modo a garantir que os gradientes permanecem estáveis (Aggarwal, 2018).

Duas técnicas preliminares de inicialização dos pesos da rede baseiam-se na geração aleatória dos pesos, considerando uma distribuição gaussiana com média 0 e desvio padrão $\sqrt{\frac{1}{r}}$, sendo *r* o número de entradas do neurónio (Aggarwal, 2018). A outra abordagem recorre a uma distribuição uniforme com intervalo: $-\sqrt{\frac{1}{r}}\ e\ \sqrt{\frac{1}{r}}$ (Aggarwal, 2018).

Na última década foram introduzidas várias técnicas que demonstraram melhorias significativas na *performance* das redes, destacando-se as técnicas *Glorot* e *He*.

Segundo Goodfellow et.al (2016), a técnica *Glorot* (Glorot e Bengio, 2010) "visa garantir que todas as camadas da rede apresentam a mesma variância de ativação e ainda a mesma variância nos seus gradientes".

Esta técnica pode inicializar os pesos recorrendo a uma distribuição gaussiana com média 0 e desvio padrão: $\sqrt{\frac{2}{r_{in} + r_{out}}}$, ou através de uma distribuição uniforme com intervalo: $[-\sqrt{\frac{6}{r_{in} + r_{out}}}, \sqrt{\frac{6}{r_{in} + r_{out}}}]$. r_{in} representa o número de entradas de um neurónio, enquanto que r_{out} representa o seu número de saídas (Aggarwal, 2018).

Glorot implementou esta técnica baseando-se em ativações do tipo sigmóide. Contudo, e após a divulgação da rede *AlexNet*, a função *ReLU* tornou-se a função de ativação padrão (redes CNN).

A técnica *Glorot* não se revela ótima na presença da função *ReLU* (He *et al.*, 2015b). Sendo assim, He et.al (2015b) desenvolveram uma nova abordagem de inicialização dos pesos da rede neuronal. A técnica He visa adaptar a ideia introduzida por Glorot, em conformidade com a utilização da função de ativação *ReLU*. A única diferença introduzida por He, consiste na multiplicação da variância dos pesos por 2.

Esta necessidade está relacionada com a representação gráfica da função *ReLU* (Figura 13), isto é, se a ativação dos neurónios for negativa (probabilidade de 50%, reduzindo a variância em 50%), então a saída de um neurónio e o seu consequente gradiente serão iguais a 0. Os neurónios encontram-se assim "desativados", não contribuindo para a atualização dos pesos da rede (Lu *et al.*, 2020). A técnica He garante que existe um reajuste da variância, através da duplicação da variância dos pesos (Hanin e Rolnick, 2018).

5.6 Data Augmentation

A técnica de "*Data Augmentation*" consiste no aumento do número de dados de treino, recorrendo a variações da informação disponível e utilizada em treino (Perez e Wang, 2017). Ou seja, a informação utilizada em treino aumenta, através da geração de novas amostras contendo "pequenas" alterações do conteúdo original (dados de treino). Esta técnica é especialmente eficiente, quando a informação disponível é reduzida ou as classes não são balanceadas, uma vez que o aumento da informação propicia a melhoria da capacidade de convergência dos modelos.

O tipo de variações a aplicar ao conteúdo está diretamente relacionado com a natureza do problema, isto é, devem ser ponderadas quais as alterações que melhor se ajustam ao problema e que não comprometem e inviabilizam o desempenho do modelo.

Quando os dados do problema são imagens, as alterações (*augmentations*) mais comuns são: rotação, inversão horizontal/vertical, *zoom in/out*, deslocação horizontal/vertical e ajuste do brilho. A aplicação desta técnica é menos usual em problemas que reúnam dados tabulares, ainda assim, frequentemente, é considerada a replicação de amostras ou a adição de *noise*.

CAPÍTULO 6
METODOLOGIA

Este capítulo sintetiza a *pipeline* seguida durante o desenvolvimento do projeto, nomeadamente: o fluxo de pré-processamento dos dados, a metodologia de otimização das arquiteturas CNN e as considerações e retificações tomadas para a otimização das diferentes arquiteturas.

6.1 Datasets

A metodologia proposta foi avaliada em três *benchmarks* médicas. As três *benchmarks* focam-se numa patologia comum, o estudo de doenças cancerígenas, justamente: cancro da mama, cancro da pele e cancro colorretal.

Os *datasets Breast Histopathology*[5], *Skin Mnist*[6] e *Colorectal Histopathology*[7] foram adquiridos através da comunidade pública *Kaggle*.

6.1.1 Breast Histopathology

O *Breast Histopathology* (Cruz-Roa *et al.*, 2014) é um problema de classificação binário, composto por um elevado conjunto de tecidos, com presença de *Invasive Ductal Carcinoma* (IDC - tipo de cancro de mama mais comum) e tecidos sem evidência de *Invasive Ductal Carcinoma*.

O *dataset* é constituído por 277524 imagens RGB (três canais de profundidade) com dimensões 50*50 pixéis (largura e comprimento, respetivamente).

Das 277524 amostras, 198,738 correspondem a tecidos sem IDC ("*No IDC*"), enquanto que as restantes 78,786 imagens são referentes a tecidos com IDC ("*With IDC*"). O *dataset* não é balanceado, apresentando uma desigualdade de 2.5 - 1 amostras, respetivamente.

Mais informações sobre o *Breast Histopathology* estão descritas no Apêndice A.

[5] https://www.kaggle.com/paultimothymooney/breast-histopathology-images
[6] https://www.kaggle.com/kmader/skin-cancer-mnist-ham10000
[7] https://www.kaggle.com/kmader/colorectal-histology-mnist

6.1.2 Skin Mnist

O *Skin Mnist* (Tschandl et.al., 2018) é um problema de classificação multi-classe composto por um total de 10015 amostras, distribuídas ao longo de sete tipos de cancro de pele: *Actinic Keratoses*, *Basal cell carcinoma*, *Benign Keratosis*, *Dermatofibroma*, *Melanocytic nevi*, *Melanoma* e *Vascular skin lesions*.

À semelhança do *dataset Breast Histopathology*, o *Skin Mnist* é um problema de classificação não balanceado, porém a distribuição do número de amostras pelas diferentes classes apresenta uma desigualdade superior. A classe *Melanocytic nevi* agrega cerca de 70% do número total de amostras do problema, sendo os restantes 30% distribuídos ao longo das restantes classes, nas quais, também se evidencia um desequilíbrio da distribuição das amostras, existindo desproporções de 10-1 amostras, entre classes.

As amostras correspondem a imagens dermatológicas representadas numa escala de 600*450 pixéis (comprimento e largura, respetivamente) e são exibidas em formato RGB.

Mais informações sobre o *Skin Mnist* estão descritas no Apêndice B.

6.1.3 Colorectal Histopathology

O *Colorectal Histopathology* (Kather *et al.*, 2016) é um problema de classificação multi-classe, composto por 5000 amostras distribuídas ao longo de oito classes. Cada classe representa um diferente tecido colorretal, existindo tecidos com presença de tumor e sem presença de tumor.

Os diferentes tecidos colorretais presentes no *dataset* são: *Tumour epithelium, Simple stroma, Complex stroma, Immune cells (lymphoid follicles), Debris, Normal mucosal glands, Adipose tissue* e *no tissue*.

Ao contrário das *benchmarks* anteriores, o *Colorectal Histopathology* é um *dataset* balanceado, estando distribuídas 625 amostras pelas oito classes do problema.

Os tecidos são representados em formato RGB e apresentam um tamanho de 150*150 pixéis (largura e comprimento, respetivamente).

Mais informações sobre o *Colorectal Histopathology* estão descritas no Apêndice C.

6.2 Leitura e divisão dos dados

Primeiramente, os dados recolhidos (três *datasets*) através da plataforma *kaggle* foram armazenados localmente.

As amostras do problema são lidas em formato imagem, sendo prontamente transformadas numa matriz de *unsigned byte's*. Esta matriz é constituída pelos "códigos" das cores (1-255),

relativos aos pixéis de cada amostra (para os três canais de profundidade). Esta operação é realizada em simultâneo com a tarefa de redimensionamento das dimensões das amostras.

Apenas, foi necessário proceder à redução das dimensões das imagens do *dataset Skin Mnist*, para as restantes *benchmarks* as dimensões mantiveram-se inalteradas, uma vez que as dimensões atuais são adequadas e suportadas pelo *hardware* utilizado no estudo. O mesmo não acontece com o problema *Skin Mnist*, que é composto por amostras de elevada dimensão (600*450 pixéis), sendo necessário proceder à redução das suas dimensões, de modo a minimizar as exigências computacionais e consequente tempo gasto no processo de treino das redes. As dimensões foram reduzidas para 128*128 pixéis.

Do processo de leitura das imagens (para cada *benchmark*, resultaram duas matrizes) resultaram duas matrizes: a 1º contendo a representação das amostras do problema (*features*), isto é, os valores *unsigned byte* de cada pixel das amostras, enquanto que a 2º matriz é composta pelos respetivos *targets* das amostras (classe a que pertence cada amostra).

Finalmente, este conjunto de dados inicial (as duas matrizes contextualizadas no parágrafo anterior) é dividido em três conjuntos distintos: treino, validação e teste. A Figura 25 ilustra a abordagem de divisão dos dados considerada.

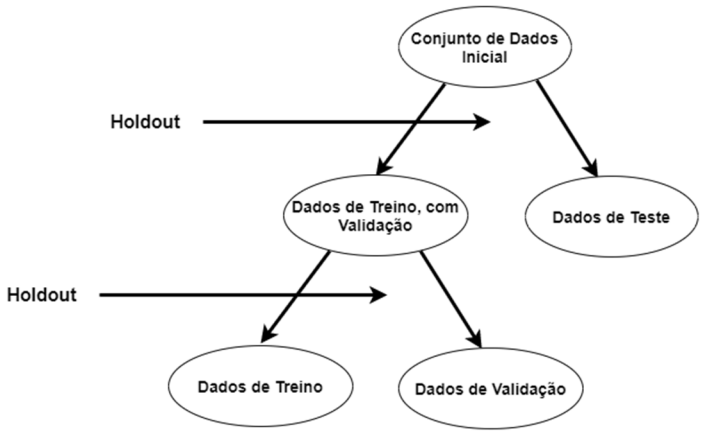

Figura 25 – Abordagem de divisão dos dados

Os dados de treino correspondem aos exemplos que a rede utiliza na aprendizagem e no ajuste dos seus parâmetros. Os dados de validação avaliam e monitorizam a capacidade de generalização da rede durante o treino. Os dados de teste são utilizados, após o treino, na avaliação do conhecimento adquirido pela rede.

O conjunto de dados inicial é dividido de forma estratificada (divide e mantém a proporção de amostras por classe inalterada, face à distribuição do conjunto de dados inicial), em dois conjuntos de dados: treino e teste.

Para os três *datasets* em estudo foi considerada a mesma abordagem de divisão do conjunto de dados inicial, isto é, o conjunto de treino corresponde a 80% do total de amostras do conjunto inicial, enquanto que os restantes 20% representam os dados de teste.

O conjunto de validação é obtido, através da divisão estratificada do conjunto de treino, ou seja, uma determinada percentagem das amostras do conjunto de treino é distribuída (dividindo e mantendo a proporção de amostras por classe inalterada) pelo conjunto de validação.

Para a *benchmark Breast Histopathology*, o conjunto de treino corresponde a 60% do total de amostras existentes e o conjunto de validação a 20%. As restantes *benchmarks* como apresentam um menor número de amostras disponíveis, o conjunto de validação corresponde a 15% do conjunto de dados inicial e o conjunto de treino a 65%.

Não foi considerado o uso de *cross validation* (*k-fold cross validation*) para consequente avaliação das soluções, uma vez que a sua utilização implicaria que cada solução fosse treinada e testada *k* vezes (*k* indica o número de subconjuntos criados), o que provocaria um aumento drástico do tempo de execução da metodologia proposta.

6.3 Pré-Processamento dos dados

Após a leitura e a separação do conjunto de dados inicial dos três problemas em análise, é necessário proceder à aplicação de um determinado conjunto de técnicas, de modo a garantir que os dados representam adequadamente os problemas e atenuam as limitações existentes, promovendo uma melhoria da generalização e convergência dos modelos.

As amostras foram transformadas recorrendo à técnica *z-score*. Esta abordagem transforma os dados, de modo a que estes sigam uma distribuição normal, ou seja, garante que as *features* (pixéis) tem média 0 e desvio padrão 1. A equação 35 traduz a técnica *z-score*.

$$X_{new} = \frac{X - \mu}{\sigma} \tag{35}$$

Esta técnica foi aplicada nos três conjuntos de dados de cada problema em análise (e resultantes do processo de separação): treino, validação e teste. A média μ e o desvio padrão σ são calculados, independentemente, para cada um dos três canais de profundidade e apenas para o conjunto de treino.

A média e o desvio padrão, de cada canal de profundidade, são apenas calculados para o *dataset* de treino, de modo a evitar *data leakage* (modelos tendenciosos). Sendo assim, a transformação dos dados de validação e de teste considera, unicamente, a informação utilizada em treino (média e desvio padrão do *dataset* de treino), garantindo assim que os modelos não têm acesso a informação "extra-treino".

Os *datasets Breast Histopathology* e *Skin Mnist* não são balanceados, dificultando a correta generalização dos modelos, sobretudo para as classes menos balanceadas. Como existe interesse na classificação correta destas classes (identificação de tecidos com IDC, ou identificação de melanoma), é necessário minimizar a desproporção existente entre as classes. Dessa forma, a rede tende a generalizar melhor os dados, em contexto geral, e principalmente para as classes menos balanceadas (More, 2016).

As técnicas *random undersampling/oversampling* e *cost sensitive learning* (técnica *post processing*, mas evidenciada, aqui, para facilitar a leitura) foram aplicadas em ambas as *benchmarks*, com o intuito de reduzir o desequilíbrio da distribuição de amostras entre as classes e com vista à melhoria da capacidade de generalização dos modelos (Branco et.al.,

2016). Foram consideradas estas técnicas, em detrimento de outras, devido à sua efetividade e à sua simplicidade de compreensão e de uso (Haixiang *et al.*, 2017).

O *Breast Histopathology* reúne um elevado número de amostras e é mais equilibrado que o *Skin Mnist*, contudo os modelos revelam dificuldades na aprendizagem da classe menos balanceada, originando desempenhos muito díspares entre as duas classes.

Como o número de amostras do problema é muito elevado, uma abordagem comum de balanceamento das classes consiste na redução do tamanho do *dataset*, através da eliminação aleatória de amostras da classe mais balanceada (apenas no *dataset* de treino) (Kotsiankis et.al., 2006). Esta técnica é conhecida por *random undersampling*.

O número de amostras da classe mais balanceada ("*No IDC*") foi reduzido, de modo a que o diferencial entre $\frac{No\ IDC_{amostras}}{With\ IDC_{amostras}}$ aumentasse de 0.4 para 0.6, promovendo assim, um maior equilíbrio entre as duas classes. A redução do número de amostras da classe mais balanceada não foi mais severa, de modo a evitar o desperdício inadequado de um enorme volume de dados, e que pode ser importante na aprendizagem dos modelos. Ainda assim, a aplicação da técnica *random undersampling*, nos moldes descritos, resultou na eliminação de cerca de 40000 tecidos "*No IDC"*. A Figura 26 ilustra o resultado da abordagem proposta.

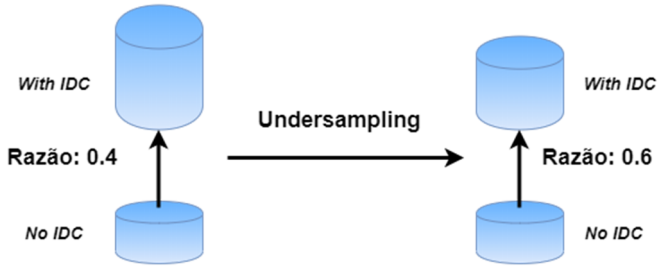

Figura 26 – Resultado da aplicação da técnica *Random Undersampling*

A técnica *random undersampling* proporcionou uma melhoria no equilíbrio da distribuição das duas classes do problema, mas ainda assim continua a existir uma diferença considerável no número de amostras de ambas as classes. De modo, atenuar a desproporção existente e sem recorrer à eliminação de mais amostras, ou eventualmente, ao aumento do número de amostras da classe menos balanceada (aumentaria a exigência computacional), foi considerada a utilização da técnica *cost sensitive learning*.

Cost sensitive learning é uma técnica geralmente aplicada em problemas não balanceados, que considera a atribuição de penalizações distintas, na função de custo, para as diferentes classes do problema (Hilario *et al.*, 2018). Em muitas situações é necessário garantir que o modelo reduz a quantidade de erros, que estão associados à classificação incorreta das amostras de uma determinada classe. Esta abordagem considera a penalização mais severa das classes menos balanceadas, com o intuito de garantir que os modelos se tornam mais "sensíveis" à correta classificação das amostras destas classes (Hilario *et al.*, 2018).

Para a *benchmark Breast Histopathology*, a atribuição de penalizações distintas entre classes, promove a minimização do número de erros que estão associados à previsão incorreta da classe menos balanceada *"With IDC"*. É fulcral garantir que o modelo revela robustez na previsão

desta classe, uma vez que a saúde dos pacientes está em risco, isto é, se o modelo prevê que um tecido não contém evidências de IDC, quando na realidade tem, é um erro muito grave e que pode comprometer a saúde do paciente. Esta abordagem garante que os modelos generalizam melhor os exemplos da classe *"With IDC"*, já que o erro da rede aumenta "exageradamente", com o aumento das previsões incorretas das amostras referentes a esta classe (maior penalização).

A definição da penalização a atribuir às diferentes classes do problema baseou-se na proposta de Hilario (2018), que calcula a penalização de cada classe, de acordo com a inversão proporcional da sua frequência ao longo do conjunto de treino. Ou seja, é calculada a penalização da classe *i*, através da divisão entre o número total de amostras do conjunto de treino e a multiplicação, entre o número de classes do problema e o número de amostras de treino da classe i. A equação 36 traduz a penalização da classe *i*.

$$penalização_i = \frac{\sum_{j=1}^{nºclasses} nº\ amostras_j}{nº\ classes\ *\ nº\ amostras_i} \quad (36)$$

A estratégia adotada para minimizar o desequilíbrio da distribuição das classes do *dataset Skin Mnist*, foi distinta da abordagem descrita atrás para a *benchmark Breast Histopathology*, uma vez que os problemas apresentam características distintas, nomeadamente: número de classes, número de amostras do problema e o grau de desproporção da distribuição das classes.

A utilização da técnica *Random Undersampling* não é viável na resolução deste problema, visto que a quantidade de informação disponível é reduzida e o número de amostras de algumas classes é extremamente baixo, por exemplo, a classe *Dermatofibroma* reúne apenas 75 amostras de treino. Existe assim, a necessidade de inclusão de um maior número de amostras de treino, referentes às classes menos balanceadas, com o intuito de promover uma melhoria na aprendizagem dos modelos.

Sendo assim, foi considerada a utilização da técnica *Random Oversampling*. Esta técnica aumenta o número de amostras de treino, através da inclusão aleatória de "cópias" das amostras das classes menos balanceadas (Kotsiantis et.al., 2006).

Para as seis classes menos balanceadas do problema foram introduzidas cópias aleatórias das suas amostras, referentes e adicionadas apenas no conjunto de treino, até que o total das amostras de cada classe representasse cerca de 50%, do total de amostras de treino da classe *Melanocytic nevi*. A Figura 27 ilustra o resultado da aplicação da técnica *random oversampling*.

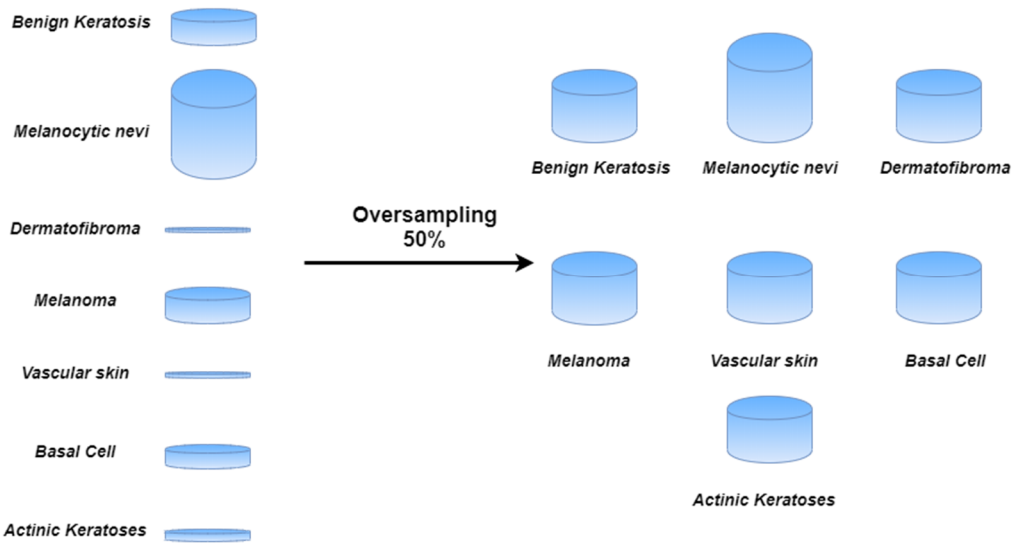

Figura 27 – Resultado da aplicação da técnica *Random Oversampling*

Não foi considerada a inclusão de um maior número de amostras (cópias) nas classes menos balanceadas, visto que não foram evidenciadas quaisquer vantagens no desempenho dos modelos, com o aumento excessivo da distribuição destas classes. O aumento imoderado do número de cópias, implica maiores exigências computacionais e resulta normalmente em *overfitting* dos modelos.

Apesar da técnica *Random Oversampling* ter promovido um maior equilíbrio na distribuição das classes, continua a existir uma discrepância significativa entre a classe mais balanceada e as restantes classes, respetivamente uma desproporção de 2 - 1 amostras.

De modo a atenuar este desequilíbrio foi proposta, a mesma abordagem utilizada na *benchmark Breast Histopathology*, ou seja, foram definidas penalizações desiguais para as diferentes classes. Os modelos tornam-se assim mais sensíveis à aprendizagem das classes menos balanceadas.

O *Colorectal Histopathology* é um problema balanceado, sendo assim a única tarefa de pré-processamento aplicada, foi a transformação dos seus dados recorrendo à técnica *z-score*, em conformidade com o que fora descrito anteriormente. Não existiu a necessidade de atribuir penalizações distintas aos erros das diferentes classes (ou eventualmente à inclusão de cópias das amostras de algumas classes), uma vez que os modelos não revelaram dificuldades na previsão das diferentes classes e o desempenho das previsões de cada classe é muito similar entre si.

6.4 Fluxo de Classificação Individual das *Benchmarks*

Concluída a fase de pré-processamento dos dados, procedeu-se à definição do fluxo de "classificação de *benchmarks*". Este fluxo descreve a *pipeline* de passos a adotar, durante a resolução de um problema de classificação, dando assim suporte a um conjunto de tarefas que são executadas ao longo do processo de classificação, nomeadamente: preparação dos dados,

criação, treino, teste e persistência de um modelo. A Figura 28 ilustra o fluxo de classificação proposto.

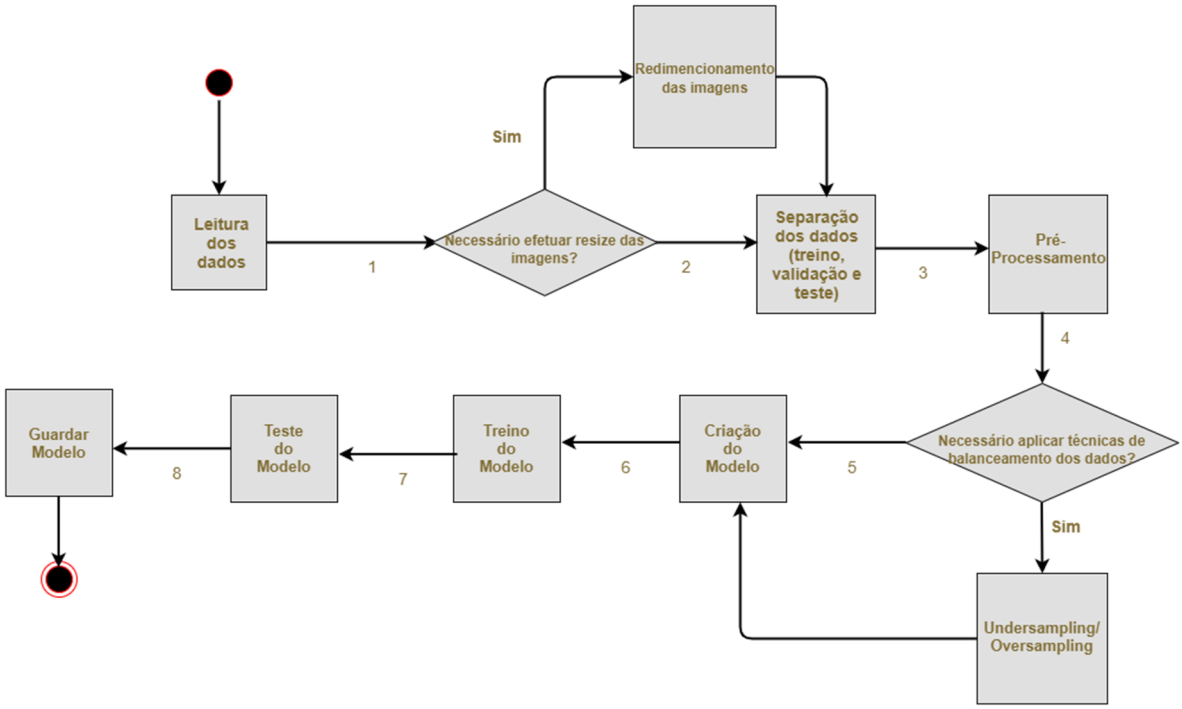

Figura 28 - Fluxo de Classificação das *Benchmarks*

A abordagem inicial do fluxo considera a realização das tarefas de preparação e pré-processamento dos dados, descritas respetivamente nas seções 6.2 e 6.3.

Os passos subsequentes sintetizam as tarefas comuns e geralmente consideradas durante a resolução de um problema de classificação. Primeiramente, é criado o modelo em conformidade com a arquitetura convolucional pretendida (*AlexNet*, *VGGNet*, *etc*), seguindo-se respetivamente as tarefas de treino e de teste do modelo.

O último passo corresponde à persistência do modelo. Este passo é especialmente importante, uma vez que possibilita a utilização futura do modelo, permitindo a sua utilização em ambiente de produção, em efeitos de *transfer learning* ou na combinação das suas previsões com outros modelos (*ensemble*).

Este fluxo complementa o fluxo de otimização que será descrito na seção seguinte. O fluxo de classificação é utilizado para estudo prévio dos problemas e das arquiteturas e também para a consequente avaliação e persistência das soluções resultantes do processo de otimização.

6.5 Metodologia de Otimização das Arquiteturas CNN

Esta seção representa o cerne do projeto, descrevendo sinteticamente todo o processo e considerações tomadas na otimização das arquiteturas convolucionais em estudo.

Para cada uma das três *benchmarks* em análise, pretende-se identificar a estrutura e os valores de outros hiperparâmetros, de quatro conhecidas arquiteturas CNN: *AlexNet*, *VGGNet*, *ResNet* e *DenseNet*, que melhor se adequam à resolução dos três problemas.

O PSO é o algoritmo proposto para otimizar as arquiteturas descritas. Para diminuir a complexidade temporal, foi considerado o uso da topologia *gbest*. Esta topologia foi considerada em detrimento das restantes, uma vez que promove uma maior rapidez de convergência das partículas, sendo este um fator decisivo na escolha da topologia, já que o tempo disponível é limitado.

O fluxo de otimização definido esquematiza de forma sucinta a *pipeline* e o método proposto para a criação e a avaliação das partículas.

O método proposto pretende garantir que a estrutura e a estratégia de treino das arquiteturas em estudo, se mantém válidas e de acordo com a definição dos seus autores. Contudo, e dadas as circunstâncias em causa: problemas de âmbito específico, tempo disponível e exigências computacionais, foi necessário proceder à alteração de "pequenos detalhes" nas arquiteturas. As retificações aplicadas são enumeradas posteriormente (seção 6.5.2).

Um processo de otimização é cíclico, demorado e muito exigente computacionalmente. Dessa forma, é necessário ter em consideração estes fatores, no momento de identificação das variáveis a otimizar e do seu consequente espaço de procura. Nas subsecções que se seguem são descritas as dimensões, de cada arquitetura a otimizar e em concordância com os três problemas em estudo, e os seus respetivos limites.

Esta seção contextualiza ainda os valores que foram atribuídos aos hiperparâmetros do PSO, as funções objetivo a otimizar (para as três *benchmarks*), a estratégia de tratamento de soluções inválidas e a técnica de *ensemble* adotada.

6.5.1 Fluxo de Otimização

A Figura 29 ilustra o fluxo proposto para a otimização das arquiteturas CNN.

Este fluxo representa, genericamente, a sequência de "passos" seguidos na otimização das arquiteturas CNN em estudo. A sua utilização é independente da *benchmark* em análise e da arquitetura a otimizar.

O fluxo é iniciado, pressupondo que o *dataset* já se encontra devidamente preparado e pré-processado (aplicadas as técnicas descritas nas seções 6.2 e 6.3).

Seguidamente, são atribuídos valores aos hiperparâmetros do PSO, nomeadamente: iterações, inércia, número de partículas e constante social e cognitiva, em concordância com a topologia utilizada.

Os dois passos que se seguem correspondem, respetivamente, à definição das dimensões e seus limites e da função objetivo. As dimensões representam as "variáveis" a otimizar, neste caso, os principais hiperparâmetros das arquiteturas CNN em estudo. A dimensões (hiperparâmetros)

a otimizar estão em concordância com a formulação original das arquiteturas (seções 4.5.2-4.5.5) e ainda com as considerações e reformulações adotadas para cada arquitetura (seções 6.5.2.2-6.5.2.5).

A cada dimensão são atribuídos limites, responsáveis por restringir a movimentação das partículas e por definir a gama de valores a otimizar para cada hiperparâmetro. Os limites das dimensões variam entre diferentes *benchmarks*, uma vez que cada *benchmark* apresenta as suas próprias limitações (número total de amostras, resolução das amostras e complexidade do problema).

A função objetivo é independente do tipo de arquitetura CNN a otimizar. Contudo, a sua definição depende da *benchmark* em análise, sendo determinada em conformidade com as características do problema. Independentemente disso, o objetivo é comum: maximizar o desempenho (métricas em avaliação) dos modelos e ao mesmo tempo minimizar a sua complexidade (minimizar o número de parâmetros de treino).

O processo de inicialização do *swarm* e a consequente atualização das posições e velocidades das partículas, mantiveram a abordagem padrão e contextualizada na seção 3.2.1.1.

A avaliação das soluções descreve uma versão alterada da abordagem sintetizada na seção 3.2.1.2. Ou seja, o fluxo padrão do PSO manteve-se inalterado, porém foi necessário estabelecer um conjunto prévio de passos, antes de proceder à avaliação das partículas, uma vez que, só é possível avaliar uma solução (isto é, aferir o desempenho e a complexidade do modelo CNN correspondente à solução – modelo corresponde à posição da partícula no espaço), após a criação, treino e teste do modelo correspondente à sua posição no espaço (ao longo de todas as dimensões).

A posição de uma partícula no espaço corresponde a um conjunto, possível e válido, de valores contínuos, referentes aos hiperparâmetros a otimizar para uma determinada arquitetura. Cada solução representa assim um possível modelo ótimo para a resolução do problema. Os modelos são construídos, considerando que os valores dos seus hiperparâmetros (hiperparâmetros em otimização) correspondem aos valores convertidos das posições das partículas ao longo das dimensões.

No ato de criação dos modelos é necessário ter em consideração que o PSO é um algoritmo utilizado na resolução de problemas contínuos, mas como as variáveis a otimizar são discretas (com exceção de uma dimensão contínua na rede *DenseNet*), torna-se necessário converter a posição das partículas, ao longo de todas as dimensões, de valores contínuos para discretos. As posições das partículas mantêm-se inalteradas (manutenção dos valores contínuos), sendo que a conversão é aplicada unicamente para efeitos de criação dos modelos, garantindo assim que não existem perdas de informação na atualização das suas posições e velocidades. A estratégia de conversão, converte o valor contínuo de cada dimensão para o, menor, valor inteiro mais próximo (técnica *floor*).

Os modelos são treinados durante um número reduzido de *epochs* (este valor depende do problema em estudo: 15, 16 e 20 *epochs*, respetivamente, para os problemas *Breast, Skin* e *Colorectal*), visto que não existe viabilidade para assumir valores superiores, dadas as

limitações computacionais e tempo útil disponível. Estes valores foram testados previamente, revelando-se "suficientes" na garantia de convergência dos modelos. Outra medida proposta para a redução do tempo de treino, consiste na utilização da técnica *early stopping*. Mais detalhes sobre o processo de treino são descritos na seção 6.5.2.1.

Concluído o processo de treino, o modelo testa o conhecimento adquirido no conjunto de dados de teste.

Posto isto, já se encontram reunidas todas as condições, que permitem proceder à avaliação das partículas (cálculo da função objetivo), já que o desempenho das partículas foi avaliado no passo anterior "Testar Modelo" e a complexidade corresponde ao número de parâmetros de treino dos modelos criados. A seção 6.5.5 descreve com maior detalhe, o processo de avaliação das partículas.

Após a conclusão do processo iterativo, o PSO retorna a *fitness* e a posição da partícula *gbest* (valores contínuos de cada dimensão otimizada).

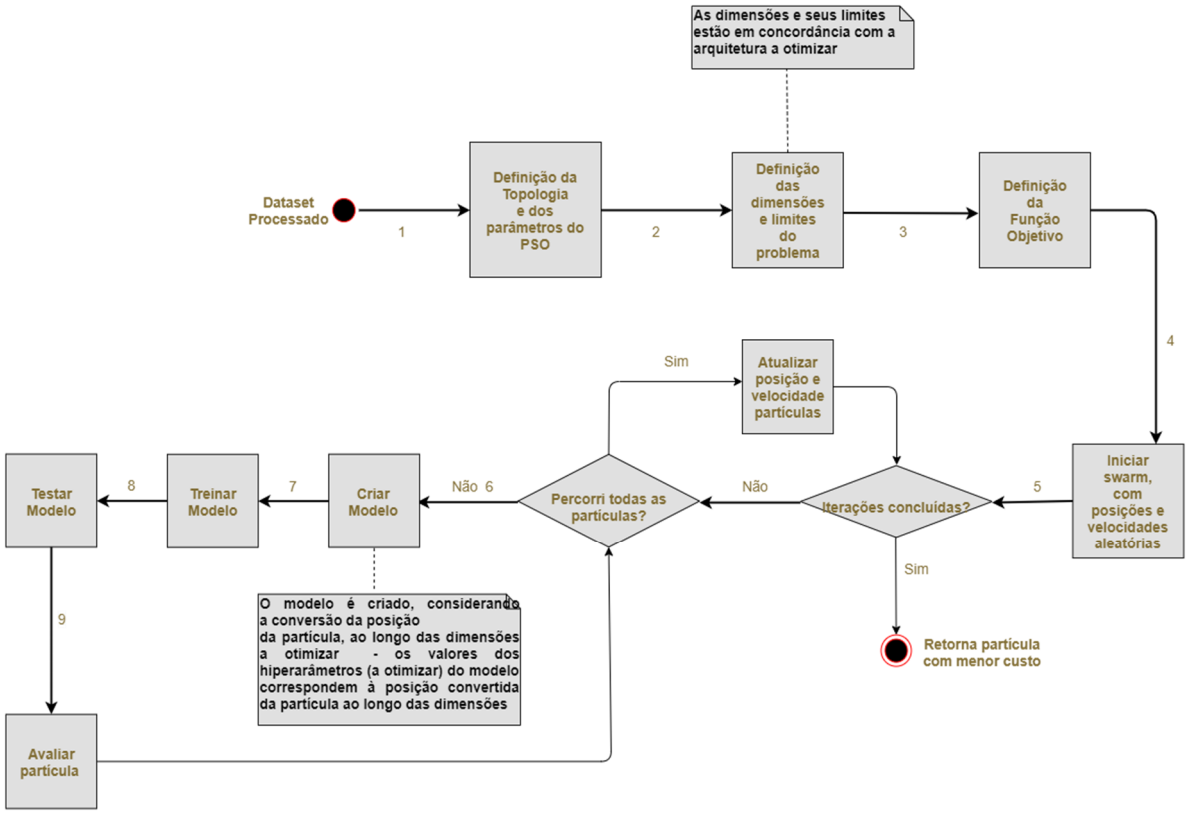

Figura 29 - Fluxo de Otimização

6.5.2 Considerações e modificações propostas, na definição e no treino das arquiteturas

A estrutura original das arquiteturas foi formulada, em conformidade com a resolução do problema *ImageNet*, que é um problema composto por imagens com dimensões elevadas e variáveis.

Sendo assim, é necessário adaptar e reformular algumas características das arquiteturas, de modo a que estas se ajustem, em concordância com os problemas em estudo e com as limitações conhecidas (exigências computacionais e tempo disponível).

Seguidamente, irá ser descrita a abordagem de treino utilizada e comum às diferentes arquiteturas em estudo. Posteriormente, são detalhadas as considerações e alterações efetuadas em cada arquitetura, sendo descritas em conformidade com as diferentes *benchmarks* em análise, já que algumas das alterações efetuadas variam entre problemas.

6.5.2.1 Abordagem de treino

A estratégia de treino utilizada pelos autores das diferentes arquiteturas em estudo é muito semelhante entre si, dessa forma foi definida uma metodologia de treino comum. As técnicas de regularização utilizadas, originalmente, pelos autores das arquiteturas em análise, foram adotadas e empregadas no estudo. Ainda assim, foi necessário reformular algumas das técnicas, de modo a garantir que estas se ajustam à natureza dos problemas e às limitações existentes.

Foi considerado o uso das seguintes técnicas de regularização: *Data Augmentation*, *Early Stopping* e norma de regularização *L2*, sendo ainda utilizadas outras abordagens, como *Dropout* ou *Batch Normalization*, mas a sua utilização (ou não) varia entre diferentes arquiteturas. Estas duas técnicas são contextualizadas especificamente nas seções seguintes (individualmente para cada arquitetura).

As *augmentations mirror* e *shifting* foram aplicadas em todas as *benchmarks*, tal como sugerido originalmente pelos autores das arquiteturas em análise (Krizhevsky et.al., 2012; He *et al.*, 2015a; Simonyan e Zisserman, 2015; Huang *et al.*, 2018). Não foi considerada a utilização de *augmentations* que alteram o significado das imagens, como por exemplo: alteração do brilho ou das cores dos pixéis, visto que a cor é uma característica diferenciadora para a correta classificação das amostras (Apêndices A, B e C). O *dataset Skin Mnist* é mais complexo do que as restantes *benchmarks*, sendo assim foram introduzidas duas *augmentations* adicionais, rotação e *zoom* das imagens, com o intuito de fornecer aos modelos um "maior leque" de amostras de treino, o que, por sua vez, promove a melhoria dos seus desempenhos.

A norma de regularização *L2* penalizou os pesos de todas as camadas convolucionais e também das *Fully-Connected Layers* (quando aplicável). O contributo da norma é de 1^{-6}.

De modo, a reduzir o tempo exigido no processo de otimização das arquiteturas, foi considerado o uso da técnica *Early Stopping*. Se o erro de validação não diminuir ao fim de 3 *epochs* (*Colorectal Histopathology*), ou 2 *epochs* (*Breast Histopathology* e *Skin Mnist*), então dá-se por concluído o processo de treino da partícula em avaliação. O conjunto de pesos que apresentou menor erro, ao longo do processo de treino da partícula, é utilizado para aferir o seu desempenho (no *dataset* de teste).

A constante *global learning rate* (seção 4.4.2.5) foi inicializada em 0.001, e caso não haja uma redução do erro de treino ou de validação ao fim de uma *epoch*, então a constante é reduzida, considerando a sua multiplicação por 0.7.

O otimizador *Adam* foi usado em detrimento do algoritmo SGD, dada a maior rapidez de convergência que proporciona, sendo este um fator importante, dado o baixo número de *epochs* utilizadas na otimização das arquiteturas (Kingma e Ba, 2017).

Foram utilizadas as funções de custo *Binary Cross Entropy* (*Breast Histopathology*) e *Multi-Class Cross Entropy* (*Skin Mnist* e *Colorectal Histopathology*).

Os pesos das redes foram inicializados com recurso à técnica *He* uniforme. As arquiteturas *AlexNet* e *VGGNet* não utilizam por defeito esta técnica, consideram, antes, uma distribuição *gaussiana* com desvio padrão 0.01. Contudo, foi sugerido por Simonyan e Zisserman (2015), após a submissão do artigo referente à rede *VGGNet*, que a *performance* da rede pode ser melhorada, através do uso de técnicas de inicialização aleatória dos pesos da rede, como por exemplo, a técnica *He*, que foi utilizada e demonstrou eficácia nas redes *ResNet* e *DenseNet*.

6.5.2.2 Modificações aplicadas à rede *AlexNet*

De modo, a viabilizar o processo de otimização da arquitetura *AlexNet* foi necessário reformular algumas das suas propriedades técnicas.

A rede *AlexNet* estabelece uma redução inicial muito drástica das dimensões das amostras (*stride* da 1ª convolução muito grande), dificultando a sua utilização em problemas que reúnam amostras com menores dimensões, já que muita informação é despendida, resultando normalmente em modelos com baixas *performances* (Zeiler e Fergus, 2014). De modo, a atenuar esta limitação foi considerada a redução do *stride* da 1ª convolução, de 4 passos para 2 (*Skin Mnist* e *Colorectal Histopathology*), e para 1 (*Breast Histopathology*).

As dimensões do *kernel* da 1ª convolução foram reduzidas de 11*11 para 5*5. Por sua vez, o *kernel* da 2ª convolução foi também reduzido de 5*5 para 3*3. As dimensões dos *kernels* foram reduzidas, uma vez que o reconhecimento dos diferentes tipos de tecidos/lesões requer a extração de detalhes precisos. A identificação destas características minuciosas é mais eficiente, quando se recorre ao uso de *kernel's* com baixas dimensões (Pereira *et al.*, 2016).

Em conformidade com a definição original dos autores, a rede é constituída por dois tipos de blocos: *single convolution blocks* e *stacked convolution blocks*. Sendo inicialmente composta por *single blocks* e posteriormente precedida por *stacked blocks*, não existindo alternância entre diferentes blocos. A proposta de otimização desta rede, visa promover a exploração de estruturas *AlexNet* mais escaláveis, isto é, com profundidades variáveis, não estando limitadas a um número fixo de camadas (durante a otimização, serão exploradas redes *AlexNet* compostas no mínimo por um bloco *single* e por um bloco *stacked*).

De modo, a reduzir o poder computacional que pode advir da utilização de vários *stacked convolution blocks*, foi definido que um bloco *stacked* é composto, apenas, por duas camadas de convolução sobrepostas, ao invés das três utilizadas por *Krizhevsky*.

A técnica de regularização *Batch Normalization* foi utilizada em detrimento da técnica de generalização *Local Response Normalization*, uma vez que Wu e He (2018) comprovaram que a utilização de *Batch Normalization* propicia melhores desempenhos.

A técnica de regularização *Dropout* foi utilizada na regularização das *Fully-Connected Layers* e nas camadas convolucionais. O número de neurónios a otimizar (seção 6.5.4) para cada *Fully-Connected Layer*, é manifestamente inferior ao número de neurónios utilizados pela arquitetura original de *Krizhevsky*, dessa forma foi reduzido o valor da constante $dropout_{rate}$ de 0.5 para 0.2.

A possibilidade de criação de redes *AlexNet* mais profundas, manifestou a necessidade de inclusão de *dropout* após cada bloco de convolução (*single* ou *stacked*), com vista à redução da variância dos modelos (atenuação de *overfitting*), sendo considerando um $dropout_{rate}$ de 0.2. Esta modificação foi sustentada por vários estudos, que foram desenvolvidos e que demonstraram a efetividade do uso da técnica *Dropout*, na regularização das camadas convolucionais (Park e Kwak, 2017; Cai *et al.*, 2020). A utilização de *Dropout* reduz o número de neurónios que não são ativados em cada camada ("mortos" – função de ativação *ReLU*) e que não contribuem para a melhoria da generalização da rede, o que possibilita a aprendizagem de características mais informativas das amostras (Park e Kwak, 2017). Os estudos demonstraram melhorias no desempenho dos modelos, quando considerado o uso da técnica *Dropout* na regularização das camadas convolucionais.

6.5.2.3 Modificações aplicadas à rede *VGGNet*

A estrutura da rede *VGGNet* é mais simples de otimizar do que à rede *AlexNet*, visto que reúne um fluxo de operações mais uniforme. Sendo assim, o número de reformulações efetuadas à rede foi diminuto.

Cada modelo terá de ser constituído no mínimo, por dois *stacked convolution blocks*. Sendo que, e em concordância com o que fora determinado para a rede *AlexNet*, cada *stacked block* agrega duas camadas convolucionais sobrepostas (de modo a reduzir exigências computacionais).

A arquitetura "original" considera que cada *stacked block* extrai o dobro das *feature maps* obtidas no *stacked block* anterior. De modo, a evitar a criação de modelos muitos complexos, estabeleceu-se que o aumento do número de *feature maps* a extrair num bloco, é determinado através da adição de um número fixo, ao número de *feature maps* extraídas por cada convolução do bloco anterior (dimensão *growth rate* – seção 6.5.3). Ou seja, cada convolução presente num *stacked block* extrai um número adicional de *feature maps* e, não o dobro das *feature maps* criadas por uma convolução do bloco anterior.

As técnicas *Batch Normalization* e *Dropout* foram aplicadas em conformidade, com o que fora descrito e determinado para a rede *AlexNet*.

6.5.2.4 Modificações aplicadas à rede *ResNet*

Foi estabelecido que cada conjunto de *residual blocks* é composto por um número fixo de *residual blocks*.

Definiu-se ainda, que um conjunto de *residual blocks* agrega no mínimo, um *residual block* e no máximo três. Ou seja, os modelos gerados ao longo do processo de otimização terão sempre dimensões inferiores à rede *Resnet-34*.

A estrutura dos *residual blocks* considerada é condizente com os blocos utilizados pelas arquiteturas menos densas (*ResNet*-18 e 34 – Figura 17), isto é, sem recurso ao conceito de *bottleneck*. A arquitetura original determina que os *residual blocks* inseridos num determinado conjunto, extraem o dobro das *feature maps* criadas no conjunto anterior. Mas, de modo a evitar a criação de soluções muito complexas, foi definido que cada convolução inserida num *residual block*, extrai um número adicional de *feature maps* e, não o dobro das *feature maps* extraídas pelas convoluções dos *residual blocks* do conjunto anterior.

As dimensões do *kernel* da 1ª convolução foram reduzidas de 7*7 para 5*5, de modo a permitir a extração de *features* mais complexas.

Foi ainda alterado o *stride* da 1ª convolução, de 2 passos para 1, apenas na resolução do problema *Breast Histopathology*. Esta alteração evita a redução excessiva das dimensões iniciais das amostras (as amostras do problema *Breast Histopathology* apresentam dimensões iniciais muito reduzidas, resultando numa elevada perda de informação).

Não foi considerada a utilização da técnica *dropout*, à semelhança da proposta "original".

6.5.2.5 Modificações aplicadas à rede *DenseNet*

Cada *dense block* é formado por um número fixo de *composite blocks*, sendo o número de *composite blocks* variável entre 2 e 6 (inclusivé). Não foi estipulado um valor superior a 6, de modo a reduzir o poder computacional exigido e o consequente tempo de otimização.

A 1ª camada convolucional da arquitetura partilha as mesmas características e funções da 1ª convolução da rede *ResNet*, ou seja, foram reduzidas as dimensões do *kernel* de 7*7 para 5*5 e o *stride* foi, de igual forma, reduzido na resposta ao problema *Breast Histopathology*.

A técnica *Dropout* não foi aplicada. Segundo os autores, a utilização de *data augmentation* é suficiente para atenuar *overfitting*.

6.5.3 Representação das soluções

O espaço de procura das partículas é distribuído ao longo de várias dimensões. Cada dimensão corresponde a uma variável a ser otimizada, neste caso os principais hiperparâmetros das arquiteturas em análise (e ainda a otimização do hiperparâmetro *batch size*). Sinteticamente, a posição de uma partícula é refletida através de uma matriz (tamanho da matriz = número de dimensões a otimizar), onde cada elemento da matriz representa a posição da partícula numa determinada dimensão do problema (valor contínuo).

Diferentes arquiteturas promovem a otimização de diferentes dimensões, uma vez que a estrutura e as propriedades técnicas das arquiteturas variam entre si.

Dadas as limitações computacionais existentes e o tempo útil disponível foi necessário identificar um número razoável de variáveis a otimizar para cada arquitetura, garantindo assim que o algoritmo apresenta viabilidade para apresentar "bons resultados" em tempo útil aceitável.

As tabelas esquematizadas seguidamente descrevem as dimensões a otimizar por arquitetura.

Tabela 3 - Dimensões da Arquitetura *AlexNet*

Dimensões	Descrição
Nº de *single convolution blocks*	Bloco composto por uma única camada convolucional (seguido por bloco de *pooling*);
Nº de *stacked convolution blocks*	Bloco que sobrepõe duas camadas convolucionais;
Nº inicial de filtros	Número de filtros do primeiro *single block* (ou da sua primeira camada convolucional);
Growth Rate	Número de *feature maps* a incrementar entre blocos (este incremento é replicado em cada convolução existente no bloco);
Nº de *Fully-Connected layers*	Número de *Fully-Connected layers* (a camada de *output* não faz parte);
Nº de neurónios	Número de neurónios de cada *Fully-Connected layer* (valor fixo e igual entre camadas);
Batch Size	Número de amostras a propagar em cada passo de treino da rede.

Tabela 4 - Dimensões da Arquitetura *VGGNet*

Dimensões	Descrição
Nº de *stacked convolution blocks*	Bloco que sobrepõe duas camadas convolucionais;
Nº inicial de filtros	Número de filtros utilizados por cada camada convolucional do 1º *stacked block*;

Growth Rate	Número de *feature maps* a incrementar entre *stacked blocks* (este incremento é replicado em cada convolução do bloco);
Nº de *Fully-Connected layers*	Número de *Fully-Connected layers* (a camada de *output* não faz parte);
Nº de neurónios	Número de neurónios de cada *Fully-Connected layer* (valor fixo e igual entre camadas);
Batch Size	Número de amostras a propagar em cada passo de treino da rede.

Tabela 5 - Dimensões da Arquitetura *ResNet*

Dimensões	Descrição
Nº de conjuntos de *residual blocks*	Cada conjunto de *residual blocks* é responsável por efetuar *downsampling* do *input* (exceção do 1º conjunto) e por aumentar o número de *feature maps* a extrair. Cada conjunto agrega um número fixo de *residual blocks*;
Nº de *residual blocks*	Número de *residual blocks* presentes em cada conjunto de *residual blocks* (valor fixo);
Nº inicial de filtros	Número de filtros da 1ª camada convolucional;
Growth Rate	Número de *feature maps* a incrementar entre conjuntos de *residual blocks* (este incremento é replicado em cada convolução dos *residual blocks*);
Batch Size	Número de amostras a propagar em cada passo de treino da rede.

Tabela 6 - Dimensões da Arquitetura *DenseNet*

Dimensões	Descrição
Nº de *dense blocks*	Número de *dense blocks* – blocos densos que estabelecem conexão total entre *composite blocks*;
Nº de *composite blocks*	Número de *composite blocks* presentes em cada *dense block* (valor fixo);
Nº inicial de filtros	Número de filtros da 1ª camada convolucional;
Growth Rate	Número de *feature maps* adicionadas por *composite block*;
Compression Rate	Percentagem de redução, da profundidade resultante de um *dense block*;
Batch Size	Número de amostras a propagar em cada passo de treino da rede.

6.5.4 Limites das dimensões

Para cada dimensão apresentada na seção anterior é necessário associar dois limites: limite inferior e limite superior. Os limites restringem a gama de valores em que as partículas se irão movimentar, ou seja, o conjunto de valores válidos de cada hiperparâmetro a otimizar.

Na definição dos limites é necessário ter em consideração que um espaço de procura extenso requer elevado esforço computacional e tempo de otimização. Dessa forma, foi prestada especial atenção à definição dos limites de algumas dimensões, de modo a garantir que o PSO consegue convergir em tempo útil (aceitável) para soluções "adequadas" e sem exceder as capacidades computacionais em uso.

Tal como descrito previamente, os limites de cada dimensão estão dependentes do problema em estudo, uma vez que cada *benchmark* apresenta particularidades distintas, tais como: número de amostras disponíveis, dimensões das amostras ou complexidade do problema. Sendo assim, os limites definidos para cada dimensão foram fixados, de acordo com as características das arquiteturas a otimizar e de cada problema em análise.

A *benchmark Breast Histopathology* reúne um número significativamente superior de amostras, comparativamente às outras duas *benchmarks*, o que, geralmente, requer a criação de redes CNN mais complexas (maior número de parâmetros de treino) (Sangaiah, 2019). Tendo em conta esta característica, foi definido para a *benchmark Breast Histopathology* que as dimensões: número inicial de filtros e *growth rate* teriam um limite superior mais abrangente face aos restantes problemas, de modo a possibilitar a criação de soluções mais complexas (exceção da rede *DenseNet*, de modo a evitar o aumento exagerado da complexidade dos modelos).

Foram ainda definidos limites diferenciados para a dimensão *batch size*, já que a sua definição depende das características do problema, nomeadamente da sua complexidade e do seu número de amostras. É necessário ter em consideração que problemas com elevado volume de dados, utilizam um *batch size* superior, comparativamente a problemas com menor volume de dados, de modo a evitar que o tempo de treino dos modelos aumente incomportavelmente (o elevado número de amostras por si só, já implica um elevado tempo de treino, caso o *batch size* seja reduzido, então o processo de treino torna-se ainda mais demorado – maior número de atualizações realizadas) (Goodfellow et.al, 2016). Sendo assim, os limites desta dimensão foram definidos, em conformidade com o número de amostras de cada problema, isto é, limites mais elevados para o problema com o maior número de amostras, *Breast Histopathology*, e limites mais reduzidos para as restantes *benchmarks*, *Skin Mnist* e *Colorectal Histopathology*.

Os limites referentes às dimensões que controlam a profundidade das arquiteturas *AlexNet*[8] e *VGGNet*[9] foram delineados, de acordo com as dimensões das amostras de cada problema e considerando o número de *downsampling's* permitidos[10] (a definição dos limites dependeu

[8] Número de *single convolution blocks* e número de *stacked convolution blocks*;
[9] Número de *stacked convolution blocks*;
[10] Por exemplo, considerando que as dimensões das amostras de um problema são de 4*4 e recorrendo ao uso da rede *VGGNet*, o número de *dowsampling's* permitidos é de 2. A 1ª camada de *pooling* reduz as dimensões (*stride*

ainda das modificações aplicadas a ambas as redes – seções 6.5.2.2 e 6.5.2.3). Para a dimensão "número de *Fully-connected layers"* foram estabelecidos limites iguais e reduzidos, entre os diferentes problemas e as duas arquiteturas, de modo a evitar a criação de modelos extremamente complexos. O controlo da complexidade das duas arquiteturas é, também, controlado pelos limites da dimensão "número de neurónios", sendo definido, para esta dimensão, um limite superior reduzido e igual para os três problemas, de modo a evitar a criação de soluções complexas (o uso de um elevado conjunto de neurónios, em cada camada, dá origem a um elevado conjunto de parâmetros de treino, já que os neurónios de uma camada estão conectados a todos os neurónios da camada seguinte).

Para os três problemas, foram estabelecidos limites semelhantes para as dimensões responsáveis por controlar a profundidade das redes *ResNet*[11] e *DenseNet*[12]. A definição destes limites está diretamente relacionada com as características dos problemas (dimensões das amostras, número de amostras ou complexidade do problema) e com as modificações aplicadas em ambas as redes (seções 6.5.2.4 e 6.5.2.5). Para além disso, o limite superior das dimensões: "número de *residual blocks*" e "número de *composite blocks*" é reduzido, visto que valores mais elevados implicariam a criação de arquiteturas muito profundas e complexas, algo que não é suportável. Para o problema *Skin Mnist* foi definido que o limite superior da dimensão "número de *dense blocks*" seria mais amplo, uma vez que dada a elevada complexidade do problema, pode, eventualmente, ser vantajosa a criação de modelos mais profundos ou complexos (comparativamente às outras duas *benchmarks*). Para a rede *ResNet*, não foi estabelecida nenhuma diferença, entre problemas, para os limites das duas dimensões indicadas em rodapé, visto que, para os três problemas, foram estabelecidos os limites superiores máximos (das duas dimensões), viáveis e compatíveis com as limitações computacionais e temporais existentes.

De acordo com a estratégia de conversão da posição das soluções, apresentada na seção 6.5.1, o valor contínuo de cada dimensão é convertido para um valor inteiro (técnica *floor*, com exceção da dimensão *compression rate*), ou seja, se os limites de uma determinada dimensão variarem entre [1 - 3.99], então uma solução pode adotar três valores possíveis: 1, 2 ou 3.

As tabelas apresentadas posteriormente expõem os limites definidos para cada dimensão e em concordância com a arquitetura e problema a otimizar.

Tabela 7 - Limites das dimensões da Arquitetura *AlexNet* para cada *benchmark*

Dimensões	Limites – *Breast Histopathology*	Limites – *Skin Mnist*	Limites – *Colorectal Histopathology*
Nº de *single convolution blocks*	[1 – 3.99]	[1 – 3.99]	[1 – 4.99]
Nº de *stacked convolution blocks*	[1 – 3.99]	[1 – 3.99]	[1 – 3.99]

de 2) de 4*4 para 2*2 e a 2ª reduz de 2*2 para 1*1, não sendo possível realizar mais nenhum *downsampling*. De acordo, com este exemplo e com a estrutura original da rede *VGGNet* seria permitido o uso de 2 *stacked blocks* (intercalados pelas camadas de *pooling*).

[11] Número de conjuntos de *residual blocks* e número de *residual blocks* (por conjunto);

[12] Número de *dense blocks* e número de *composite blocks* (por *dense block*);

Nº inicial de filtros	[4 - 128]	[4 - 96]	[4 - 96]
Growth Rate	[0 – 64]	[0 – 48]	[0 – 48]
Nº de *Fully-Connected layers*	[1 – 2.99]	[1 – 2.99]	[1 – 2.99]
Nº de neurónios	[8 – 96]	[14 – 96]	[16 – 96]
Batch Size	[32 – 128]	[12 – 64]	[6 – 48]

Tabela 8 - Limites das dimensões da Arquitetura *VGGNet* para cada *benchmark*

Dimensões	Limites – *Breast Histopathology*	Limites – *Skin Mnist*	Limites – *Colorectal Histopathology*
Nº de *stacked convolution blocks*	[2 – 6.99]	[2 – 7.99]	[2 – 7.99]
Nº inicial de filtros	[4 - 128]	[4 - 96]	[4 - 96]
Growth Rate	[0 – 64]	[0 – 48]	[0 – 48]
Nº de *Fully-Connected layers*	[1 – 2.99]	[1 – 2.99]	[1 – 2.99]
Nº de neurónios	[8 – 72]	[14 – 72]	[16 – 72]
Batch Size	[64 – 128]	[12 – 64]	[6 – 48]

Tabela 9 - Limites das dimensões da Arquitetura *ResNet* para cada *benchmark*

Dimensões	Limites – *Breast Histopathology*	Limites – *Skin Mnist*	Limites – *Colorectal Histopathology*
Nº de conjuntos de *residual blocks*	[1 – 5.99]	[1 – 5.99]	[1 – 5.99]
Nº de *residual blocks*	[1 – 3.99]	[1 – 3.99]	[1 – 3.99]
Nº inicial de filtros	[4 - 128]	[4 - 96]	[4 - 96]
Growth Rate	[0 – 64]	[0 – 48]	[0 – 48]
Batch Size	[64 – 128]	[12 – 64]	[6 – 48]

Tabela 10 - Limites das dimensões da Arquitetura *DenseNet* para cada *benchmark*

Dimensões	Limites – *Breast Histopathology*	Limites – *Skin Mnist*	Limites – *Colorectal Histopathology*
Nº de *dense blocks*	[1 – 4.99]	[1 – 5.99]	[1 – 4.99]

Nº de *composite blocks*	[2 - 6.99]	[2 - 6.99]	[2 - 6.99]
Nº inicial de filtros	[4 - 128]	[4 - 96]	[4 - 96]
Growth Rate	[2 – 32]	[2 – 32]	[2 – 32]
Compression Rate	[0.1 – 1.0]	[0.1 – 1.0]	[0.1 – 1.0]
Batch Size	[64 – 128]	[12 – 64]	[6 – 48]

6.5.5 Funções Objetivo

As funções objetivo definidas estabelecem um *trade-off* entre a maximização do desempenho das soluções (avaliação do desempenho dos modelos criados, e correspondentes à posição das partículas ao longo de todas as dimensões) e a minimização da sua complexidade (número de parâmetros de treino do modelo).

O desempenho das partículas foi avaliado com recurso a três métricas: *macro average precision*, *macro average recall* e *macro average F1Score*. Estas métricas (*macro average*) são particularmente importantes na avaliação do desempenho dos modelos (Manning et.al., 2008), uma vez que são "insensíveis" à desproporção da distribuição das classes, ou seja, tratam todas as classes de igual forma (Sokolova e Lapalme, 2009). A sua utilização atenua a tendenciosidade que pode surgir na análise do desempenho dos modelos. Geralmente, em problemas não balanceados, os modelos apresentam tendência para generalizar melhor as amostras referentes às classes mais balanceadas, o que resulta normalmente em desempenhos muito díspares, entre as diferentes classes (métricas avaliadas). A avaliação do desempenho dos modelos, considerando a distribuição atual (não balanceada) das classes, resulta em análises muito tendenciosas, já que os bons desempenhos das classes mais balanceadas "camuflam" os baixos resultados das classes menos balanceadas. Como, as métricas *macro average* consideram que todas as classes têm a mesma distribuição, então esta tendenciosidade na análise dos resultados, não ocorre.

Nos *datasets* em análise existe um maior interesse médico na previsão correta das classes menos balanceadas. Isto é, para o *dataset Breast Histopathology* existe interesse na correta previsão de tecidos com *IDC*, enquanto que para o *Skin Mnist*, pretende-se maximizar o número de acertos das classes: *Melanoma* ou *Basal Cell Carcinoma*. A incorreta classificação destas amostras dificulta o processo de diagnóstico dos pacientes e coloca o seu estado de saúde em risco. De realçar que, existe interesse na correta classificação de todas as amostras, mas dado o maior risco que origina a incorreta classificação das amostras das classes menos balanceadas, foi prestado maior cuidado na sua correta classificação.

As equações 37, 38, 39, 40 e 41 traduzem, respetivamente, o cálculo das métricas: precisão, *recall*, *macro average precision, macro average recall* e *macro average F1Score.*

A métrica precisão quantifica a percentagem de previsões corretas associadas a uma classe, considerando todas as amostras que foram classificadas como pertencentes a essa classe. Por sua vez, o *recall* estima a percentagem de previsões corretas para uma determinada classe, tendo em consideração as amostras que, na realidade, pertencem a essa classe. A Figura 30 clarifica o significado de TP ("verdadeiros positivos"), FP ("falsos positivos"), TN ("verdadeiros negativos") e FN ("falsos negativos").

$$precisão = \frac{TP}{TP + FP} \tag{37}$$

$$recall = \frac{TP}{TP + FN} \tag{38}$$

$$precision_{macroAVG} = \frac{\sum_{i=1}^{numberClasses} precisão_i}{numberClasses} \tag{39}$$

$$recall_{macroAVG} = \frac{\sum_{i=1}^{numberClasses} recall_i}{numberClasses} \tag{40}$$

$$F1Score_{macroAVG} = 2 * \frac{precisão_{macroAVG} * recall_{macroAVG}}{precisão_{macroAVG} + recall_{macroAVG}} \tag{41}$$

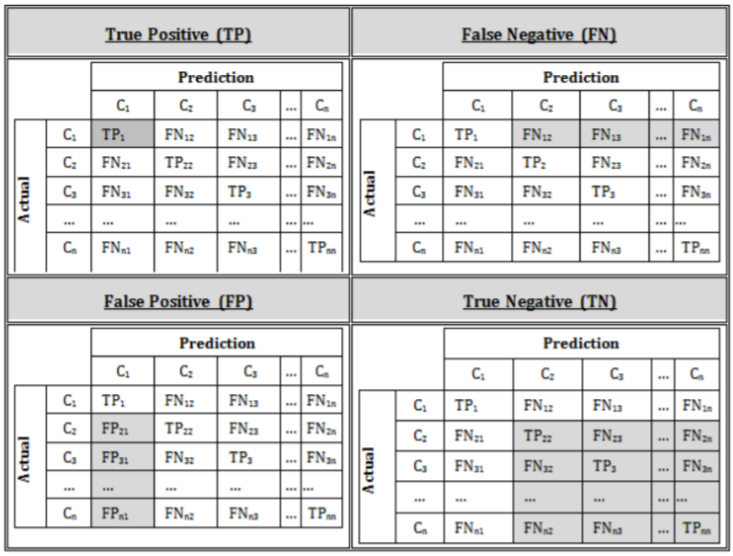

Figura 30 – TP, FN, FP e TN – Fonte:
https://shodhganga.inflibnet.ac.in/bitstream/10603/138610/13/13_chapter%205.pdf

A complexidade de uma partícula é avaliada, através da contagem dos parâmetros de treino do modelo criado e correspondente à posição da partícula no espaço, sendo que, quanto maior o número de parâmetros mais complexo é o modelo. O número de parâmetros de treino dos modelos é obtido com suporte à biblioteca *Keras*.

As equações 42 e 43 ilustram as funções objetivo (minimização) definidas para a consequente avaliação da *fitness* das soluções. A equação 42 foi utilizada na resolução dos problemas *Breast Histopathology* e *Skin* Mnist, já a equação 43 foi considerada no problema *Colorectal Histopathology*. Ambas as equações correspondem a uma reformulação das equações definidas por Vieira et.al (2013).

$$loss = 10^{-9} * trainableParams + 4.0 * (1.0 - F1Score_{macroAVG}) + 3.0 * (1.0 - recall_{macroAVG}) + 2.0 * (1.0 - precision_{macroAVG}) \quad (42)$$

$$loss = 10^{-9} * trainableParams + 6.0 * (1.0 - F1Score_{macroAVG}) \quad (43)$$

Na equação 42 foi definido um peso distinto entre *recall* e precisão. Tal como explicado anteriormente, num problema médico é crucial reduzir o número de falsos negativos (TN), de modo a garantir o bem-estar dos pacientes, ou seja, evitar que os modelos diagnostiquem lesões benignas, quando na realidade o paciente apresenta lesão maligna, daí a maior ponderação dada à métrica *recall*. Na equação 42 foi ainda incluída uma terceira métrica $F1Score_{macroAVG}$. Esta métrica foi incluída, porque ambos os *dataset's* não são balanceados, levando a que possam surgir discrepâncias significativas entre as métricas precisão e *recall* (por classes), o que é um sinal de baixa qualidade dos modelos. A utilização desta métrica afere o equilíbrio entre a $precision_{macroAVG}$ e o $recall_{macroAVG}$ (média harmónica) das diferentes classes, ou seja, quanto maior a exatidão da métrica $F1Score_{macroAVG}$, mais robusto e melhor *performance* apresenta o modelo.

Para o problema *Colorectal Histopathology* (equação 43) recorreu-se unicamente à utilização da métrica $F1Score_{macroAVG}$, não sendo, assim, estabelecidos pesos distintos para as métricas precisão e *recall*. Como, o *recall* já é significativamente elevado, não se pretende otimizar ainda mais o *recall* "às custas" da precisão, uma vez que, quanto menor for a precisão, maiores serão os custos monetários desperdiçados (hospital realiza um maior número de diagnósticos, acompanhamento dos pacientes, *etc*). Sendo assim, foi definida a mesma importância para ambas as métricas (média harmónica).

O número de parâmetros de treino é multiplicado por um valor extremamente baixo, de modo a garantir que as melhores soluções, correspondem sempre, às partículas que reúnem melhor desempenho, independentemente da sua complexidade. Ou seja, a complexidade é avaliada apenas para efeitos de desempate (elevada proximidade entre o desempenho das partículas). Os limites definidos para as dimensões a otimizar (seção 6.5.4) asseguram que a complexidade das soluções nunca ultrapassa os 10 000 000 milhões de parâmetros de treino, ou seja, a multiplicação entre $10^{-9} * trainableParams$ será sempre inferior a 0.01, garantindo que a complexidade das soluções apenas terá influência, quando as soluções apresentarem desempenhos extremamente semelhantes entre si, uma vez que os pesos atribuídos às métricas de desempenho apresentam uma influência muito superior no custo (*fitness*) das partículas.

Os pesos atribuídos às métricas de desempenho asseguram que é conferida maior, ou menor, importância a determinadas métricas (maior peso, maior importância) e garantem que a complexidade das soluções apenas terá influência, em casos de extrema semelhança no desempenho das métricas avaliadas. Os valores definidos para cada métrica (equações 42 e 43) certificam que uma solução menos complexa (menor número de parâmetros de treino) e com menor desempenho, só apresentará melhor *fitness* comparativamente a outra solução, caso a

diferença de desempenho entre as soluções, seja igual ou inferior a 0.01%[13] (considerando para a equação 42, o desempenho médio das três métricas avaliadas, e para a equação 43 o resultado da métrica $F1Score_{macroAVG}$). Mediante a necessidade de aumentar ou diminuir a importância do "fator complexidade", os pesos das métricas de desempenho podem ser diminuídos ou aumentados, respetivamente.

6.5.6 Topologia e Hiperparâmetros do PSO

Tal como indicado previamente, foi considerado o uso da topologia em estrela.

Foi utilizado um número de partículas e de iterações reduzido, respetivamente 20 e 10, de modo a garantir a obtenção dos resultados em tempo útil aceitável. Ambos os valores foram designados, em concordância com os valores propostos e utilizados por outros autores em tarefas semelhantes, nomeadamente: Fernandes e Junior (2019), Wang et.al., (2019) e Serizawa e Fujita (2020).

A convergência do PSO depende da correta definição da inércia, constante cognitiva e social. Segundo Van Den Bergh (2002) existe a garantia de convergência das partículas, caso seja satisfeita a condição traduzida abaixo.

$$1 > w > \frac{1}{2}(c1 + c2) - 1 \geq 0 \qquad (44)$$

Bergh (2002) identificou que a utilização de $w = 0.7$ e $c1 = c2 = 1.4$ permite uma rápida convergência do algoritmo para soluções adequadas, demonstrando um decréscimo efémero das oscilações iniciais das partículas. Tendo em conta as limitações existentes e a necessidade de rápida convergência das partículas foi adotada a proposta de *Bergh*.

[13] Considerando duas soluções com uma diferença de desempenho de 0,01% (média aritmética das métricas avaliadas):

- **A**: complexidade: 1 000 000 de parâmetros e desempenho: $F1Score_{macroAVG}$ = 90,01%, $Recall_{macroAVG}$ = 90,02% e $Precision_{macroAVG}$ = 90,00%; (**maior desempenho e maior complexidade**);
- **B**: complexidade: 300 000 parâmetros e desempenho: $F1Score_{macroAVG}$ = 90,00%, $Recall_{macroAVG}$ = 90,05% e $Precision_{macroAVG}$ = 89,95%; (**menor desempenho e menor complexidade**);

Aplicação da Equação 42:

- **Solução A**: 10^{-9} * 1 000 000 + (1-0,9001) * 4 + (1-0,9002) * 3 + (1-0,9) * 2 = 0,90
- **Solução B**: 10^{-9} * 300 000 + (1-0,9) * 4 + (1-0,9005) * 3 + (1-0,8995) * 2 = 0,8998 (**Melhor solução - menor *fitness***);

Aplicação da Equação 43:

- **Solução A**: 10^{-9} * 1 000 000 + (1-0,9001) * 6 = 0,6004
- **Solução B**: 10^{-9} * 300 000 + (1-0,9) * 6 = 0,6003 (**Melhor solução - menor *fitness***);

Tal como referido anteriormente, uma solução com menor complexidade só apresenta melhor (menor) *fitness* comparativamente a outra solução, caso a diferença entre os seus desempenhos seja muito próxima, isto é, igual ou menor do que 0,01%. De notar que, se a diferença da complexidade, entre as duas soluções, fosse ainda maior do que a exemplificada neste exemplo, então, a solução com menor desempenho (solução B) poderia apresentar melhor *fitness* do que a solução A (maior desempenho), caso a diferença dos seus desempenhos fosse um pouco superior a 0,01%, mas nunca superior a 0,1%. Assim, ambas as equações, garantem que as melhores soluções serão sempre as que reúnem melhor *performance*, sendo que a complexidade apenas terá impacto, em casos de extrema proximidade no desempenho das soluções.

6.5.7 Tratamento de soluções inválidas

Ao longo das iterações, as partículas podem, ocasionalmente, se movimentar para espaços que excedem os limites definidos.

Como o número de iterações e de partículas é reduzido, não é viável consentir que as partículas possam circular em espaços inválidos, uma vez que pode existir um elevado "desperdício" de iterações, até que a partícula se torne novamente válida (regresse a espaços válidos).

A posição inválida das partículas é corrigida através do uso da técnica *Shrink* (contextualizada na seção 3.2.1.3), ou seja, as partículas "param" no limite da dimensão excedida. Por sua vez, o vetor de velocidade é invertido nas dimensões excedidas, reduzindo assim a probabilidade das partículas excederem novamente os limites do problema (técnica *Invert* – seção 3.2.1.3, fator de inversão = 0.7).

6.5.8 Técnica *ensemble*

Após a conclusão do processo de otimização, foi considerado o uso de uma simples técnica *Ensemble*. O método consiste na combinação das distribuições probabilísticas obtidas pelas quatro arquiteturas em estudo (média aritmética) (Ju et.al., 2017). Esta abordagem cria uma previsão mais realista e com menor variância (Ju et.al., 2017). A Figura 31 exemplifica a abordagem descrita.

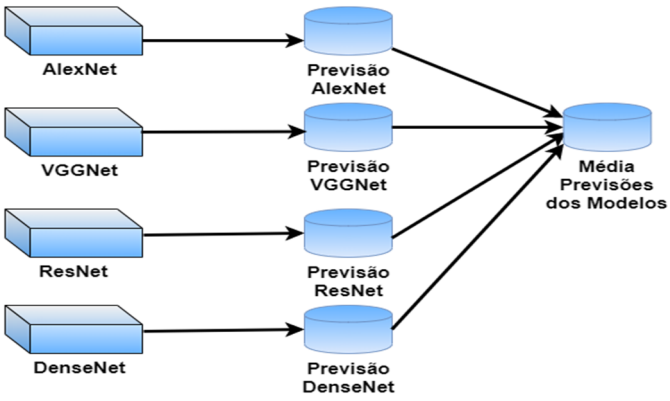

Figura 31 - Técnica *Ensemble* proposta

6.5.9 Reprodutibilidade

Para cada um dos três problemas em estudo, *Breast Histopathology*[14], *Skin Mnist*[15] e *Colorectal Histopathology*[16], foi criado um projeto independente. Estes projetos derivam de um repositório global[17], que define as classes e as funções necessárias à implementação da metodologia proposta.

As principais dependências utilizadas foram:

- *Keras* com *backend Tensorflow*: O *Tensorflow* é uma biblioteca de *Deep Learning* poderosa e desenvolvida nativamente em C++. Dá suporte à realização de um vasto conjunto de tarefas, nomeadamente: reconhecimento e classificação de imagens, processamento de linguagem natural, classificação de séries temporais, *etc*. O *Keras* atua como um *wrapper* do *Tensorflow*, fornecendo uma API intuitiva de acesso à sua *backend* (Chollet, 2015);
- *Scikit-Learn*: É uma biblioteca de *Machine Learning* (ML) desenvolvida em *Python*. Fornece um conjunto de ferramentas que dão suporte à análise dos dados de um problema (separar, pré-processar ou seleção) e à criação de modelos de ML;
- *Pyswarms*: É uma biblioteca desenvolvida em *Python* que providencia acesso a diferentes topologias do algoritmo PSO. A sua API, alto nível, promove uma fácil customização do algoritmo, nomeadamente na definição da função objetivo, fluxo de otimização ou tratamento de limites (Miranda, 2018);
- *Numpy*: É uma livraria de computação científica, desenvolvida em *Python*. A sua utilização fornece acesso a matrizes multidimensionais poderosas e eficientes, dispondo ainda de um conjunto de funções sofisticadas que facilitam o manuseamento destas matrizes;
- *h5py*: É uma biblioteca desenvolvida em *Python* que fornece uma interface alto-nível para manipulação de ficheiros *HDF5* (binários). Estes ficheiros agregam um enorme conjunto de dados numéricos. Os modelos desenvolvidos na biblioteca *Keras* são exportados em formato *HDF5*;

O projeto foi realizado em ambiente *Windows* e com recurso ao seguinte *hardware*: *Intel i5-9600K* (6 núcleos), 32*Gb* de *RAM* e *Nvidia RTX 2060 Super* (8*Gb*).

[14] https://github.com/bundasmanu/breast_histopathology
[15] https://github.com/bundasmanu/skin_mnist
[16] https://github.com/bundasmanu/Colorectal_Histopathology
[17] https://github.com/bundasmanu/Builder_Optimizer_Archicteture_AnyDataset

CAPÍTULO 7
ANÁLISE DE RESULTADOS

Este capítulo expõe e contextualiza os resultados obtidos da aplicação da metodologia descrita no capítulo 6. Foram ainda comparados os resultados obtidos com os alcançados por outros autores.

O Apêndice E complementa este capítulo, ilustrando através de gráficos e para cada arquitetura otimizada, a variação da melhor *fitness* ao longo das iterações.

7.1 Metodologia de avaliação das soluções *gbest*

O processo de otimização proposto identificou para cada arquitetura otimizada, os valores dos hiperparâmetros (hiperparâmetros otimizados em cada arquitetura) que promoveram a melhor *fitness* (menor *fitness*) ao longo das iterações.

A posição *gbest* (posição da solução com menor *fitness* identificada pelo enxame) resultante da otimização de cada arquitetura, foi convertida para valores inteiros, considerando a abordagem descrita na seção 6.5.1, ou seja, o valor contínuo de cada dimensão foi convertido para o, menor, valor inteiro mais próximo.

Seguidamente, procedeu-se à avaliação destas soluções, isto é, da posição da solução *gbest* resultante da otimização de cada arquitetura. Cada solução é treinada e testada várias vezes, através da criação do modelo (arquitetura CNN) correspondente à sua posição, ou seja, os valores dos hiperparâmetros (otimizados) do modelo correspondem aos valores convertidos da posição da solução *gbest*, ao longo das dimensões otimizadas.

É indispensável o treino e teste destas soluções em mais do que uma ocasião, uma vez que o processo de treino das redes CNN envolve "alguma" aleatoriedade. Certas técnicas de regularização utilizadas promovem a introdução de mecanismos aleatórios no treino da rede, nomeadamente: técnicas de inicialização dos pesos da rede (técnica *He*) e a técnica *Dropout*. Por sua vez, a biblioteca *Keras* não dá suporte à criação de resultados reprodutíveis[18][19], quando se utiliza a GPU no treino das redes. Ou seja, mesmo que seja utilizado o mesmo *"random seed"* na divisão dos dados do problema e na aplicação das técnicas de regularização, os resultados acabam por diferir sempre, ainda que ligeiramente.

Dessa forma, e de modo a reduzir uma eventual tendenciosidade que possa ocorrer nos resultados, foram testadas estas soluções várias vezes. Contudo, e dado o tempo útil disponível

[18] https://github.com/tensorflow/tensorflow/issues/12871
[19] https://docs.nvidia.com/deeplearning/cudnn/developer-guide/index.html#reproducibility

procedeu-se unicamente ao treino e teste destas soluções, em cinco ocasiões para o *dataset Colorectal Histopathology* e três vezes para os *datasets Skin Mnist* e *Breast Histopathology*.

Para as *benchmarks Colorectal Histopathology* e *Skin Mnist*, estas soluções foram treinadas recorrendo ao mesmo número de *epochs* (20 e 16, respetivamente) que foram utilizadas durante o processo de otimização (Seção 6.5.1). Estes valores mantiveram-se inalterados, uma vez que os modelos, ao fim do número de *epochs* definidas, já estagnaram. Por sua vez, para o *dataset Breast Histopathology*, foi necessário proceder ao aumento do número de *epochs* (40 *epochs* para a rede *DenseNet* e 30 *epochs* para as restantes arquiteturas), visto que os modelos ao fim das 15 *epochs* utilizadas durante o processo de otimização, ainda não estagnaram completamente, existindo assim margem para melhoria dos seus desempenhos.

O treino e teste destas soluções realizou-se, de acordo com o processo de divisão, dos dados de cada *benchmark*, descrito na seção 6.2 e utilizado previamente na otimização das arquiteturas.

Os resultados enumerados seguidamente correspondem à média do conjunto de testes realizados.

7.2 Resultados – *Colorectal Histopathology*

As Tabelas 11, 12, 13 e 14 enumeram respetivamente para as quatro arquiteturas em estudo, a posição convertida da solução *gbest*, ao longo das dimensões otimizadas.

A Tabela 15 esquematiza os resultados, referentes à avaliação destas soluções. Tal como mencionado anteriormente, estes resultados derivam da avaliação das soluções em cinco momentos distintos. O resultado final de cada métrica, corresponde ao valor médio das cinco avaliações realizadas. Esta tabela menciona ainda para cada arquitetura otimizada: o custo da solução *gbest* (menor *fitness* identificada pelo enxame, calculada com recurso à equação 43), o tempo gasto na sua otimização e ainda o número de parâmetros de treino da solução *gbest*.

A métrica *accuracy* é calculada com recurso à equação 45.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{45}$$

Tabela 11 - Posição da solução *gbest* - *AlexNet*

Hiperparâmetros	**Nº de *single convolution blocks***	**Nº de *stacked convolution blocks***	**Nº inicial de filtros**	***Growth Rate***	**Nº de *Fully-Connected-Layers***	**Nº de Neurónios**	***Batch Size***
AlexNet	1	2	70	40	2	74	6

Tabela 12 - Posição da solução *gbest* - *VGGNet*

Hiperparâmetros	Nº de *stacked convolution blocks*	Nº inicial de filtros	*Growth Rate*	Nº de *Fully-Connected-Layers*	Nº de Neurónios	*Batch Size*
VGGNet	4	70	19	2	71	6

Tabela 13 - Posição da solução *gbest* - *ResNet*

Hiperparâmetros	Nº de conjuntos de *residual blocks*	Nº de *residual blocks*	Nº inicial de filtros	*Growth Rate*	*Batch Size*
ResNet	4	2	61	13	6

Tabela 14 - Posição da solução *gbest* - *DenseNet*

Hiperparâmetros	Nº de *Dense Blocks*	Nº de *composite blocks*	Nº inicial de filtros	*Growth Rate*	*Compression Rate*	*Batch Size*
DenseNet	3	6	95	30	0.887	6

Tabela 15 - Descrição dos resultados obtidos

Arquiteturas	Custo Obtido	Nº de parâmetros de Treino	Duração	Resultados (*Accuracy*/*Macro*-F1Score)
AlexNet	0.28	1, 650, 000	23 horas	94.3%, 94.2%
VGGNet	0.30	1, 346, 672	27 horas	94,6%, 94.5%
ResNet	0.22	966, 800	26 horas	95.7%, 95.5%
DenseNet	0.18	1,516,405	21 horas	96.1%, 96.0%
***Ensemble – 4* Arquiteturas**	-	-	-	95.6%, 95.5%
Ensemble – ResNet + DenseNet	-	-	-	96.6%, 96.6%

As soluções *gbest*, referentes às quatro arquiteturas otimizadas, apresentaram uma complexidade semelhante. Todas elas dispõem de um elevado conjunto de parâmetros de treino, tendo em consideração o baixo número de amostras do problema. Ainda assim, este valor está distante do número de parâmetros de treino das arquiteturas definidas originalmente pelos seus

autores[20], demonstrando a "leveza" que a definição manual dos hiperparâmetros oferece, face à utilização de modelos pré-configurados.

Tal como expectável, as soluções *gbest* das quatro arquiteturas fixaram-se no limite inferior definido para a dimensão *"batch size"*. A utilização de um *batch size* reduzido, promove a atualização mais frequente dos pesos da rede por *epoch* e, a par, do baixo número de *epochs* utilizadas durante o treino das partículas, as soluções que reúnam um *batch size* reduzido tendem a apresentar melhores desempenhos, já que os parâmetros de treino da rede são atualizados mais frequentemente, possibilitando uma convergência mais robusta e rápida da rede (um *batch size* elevado requer o uso de um maior número de *epochs*, já que o modelo tende a convergir mais lentamente) (Li *et al.*, 2014; Keskar *et al.*, 2017). Em alternativa, *batches sizes* elevados apresentam maior velocidade de computação (uma *epoch* é concluída mais depressa).

A complexidade das soluções *gbest* das arquiteturas *AlexNet* e *VGGNet* revelou-se extremamente semelhante entre si, sendo a profundidade (número de camadas convolucionais utilizadas), a única diferença assinalável entre estas. A rede *VGGNet* utilizou um maior número de convoluções face à rede *AlexNet*, por sua vez, a rede *AlexNet* compensou a menor profundidade com a adição de um maior número de *feature maps* (a extrair) em cada bloco (valor superior do hiperparâmetro *growth rate*). Apesar da rede *AlexNet* ter apresentado um menor custo durante a otimização, esta rede apresentou um menor desempenho face à rede *VGGNet*. Este facto foi confirmado, após a consequente avaliação "pós-otimização" de ambas as soluções (Tabela 15), sendo que, a rede *VGGNet* apresentou um melhor desempenho médio para as métricas avaliadas (avaliação em cinco momentos distintos). Confirmando assim, a importância da avaliação das soluções em mais do que um momento, já que a biblioteca *Keras* impede a reprodutibilidade dos resultados.

A solução da arquitetura *ResNet*, curiosamente, apresentou a mesma morfologia da rede *ResNet*-18 (Figura 18), ou seja, utilizou de igual forma o mesmo número de conjuntos de *residual blocks* e de *residual blocks* por conjunto. O valor do hiperparâmetro *growth rate* é reduzido, a rede não manifesta assim a necessidade de promover um aumento drástico do número de *feature maps* a extrair, entre conjuntos de *residual blocks* (baixo *growth rate* promove baixa complexidade). Os resultados da rede, tal como expectável, foram superiores aos alcançados pelas arquiteturas *AlexNet* e *VGGNet*, dada a sua maior robustez e eficiência.

A rede *DenseNet* foi, de entre todas as arquiteturas em análise, a que revelou melhor *performance*. Os hiperparâmetros *growth rate* e *compression rate* aproximaram-se ambos do limite superior, referente às suas dimensões. A rede tende assim a apresentar melhor desempenho, quando:

- Não existe um decréscimo acentuado da profundidade do *output* resultante de um *dense block* (*compression rate* próximo de 1). Ou seja, a informação recolhida num *dense block* (praticamente toda) é preservada e dissipada ao longo de toda a rede;

[20] *AlexNet*: 60, 97 milhões de parâmetros (Hasanpour *et al.*, 2016)
VGGNet-16: 138, 36 milhões de parâmetros (Hasanpour *et al.*, 2016)
ResNet-34: 21,5 milhões de parâmetros (Leong *et al.*, 2020)
DenseNet-121: 7,2 milhões de parâmetros (Leong *et al.*, 2020)

- Os *composite blocks* contribuem com um número considerável de "nova informação" (30 *feature maps*). A imposição de um baixo limite superior para a dimensão "número de *composite blocks*" e a utilização de *composite blocks* em forma de *bottleneck*, pode ter contribuído, para a necessidade de aumento do contributo de cada *composite block*, uma vez que ambos os fatores limitam a complexidade das soluções.

A *fitness* das soluções *gbest* apresenta uma ligeira tendenciosidade, face aos resultados obtidos após a otimização. Isto é, existe uma diferença entre a *fitness* de cada solução e a média dos resultados pós-otimização, em cerca de [0.5%-1.0%]. Sendo mais expressiva a variação nas arquiteturas *AlexNet* e *DenseNet*.

Esta flutuação resulta do facto da biblioteca *Keras*[21], ao contrário de outras livrarias, não permitir que alguns dos algoritmos utilizados (e disponibilizados pela biblioteca *cuDNN*), internamente pela GPU, possam apresentar um comportamento determinista, como por exemplo, o algoritmo referente ao cálculo dos gradientes. Esta limitação da biblioteca *Keras* inviabiliza a reprodutibilidade dos resultados entre diferentes execuções, gerando assim ligeiras diferenças nos resultados obtidos entre execuções.

Foram criados dois modelos *ensemble* na tentativa de melhorar a *performance* individual das arquiteturas. O 1º modelo consiste na agregação conjunta das distribuições probabilísticas das quatro arquiteturas (média). Dada a diferença de *performances* entre as arquiteturas, era expectável que os resultados obtidos não fossem superiores aos alcançados pela rede *DenseNet* (arquitetura com melhor *performance* individual), tal como confirmado. Dessa forma, foi criado um segundo modelo *ensemble* agregando, apenas, as distribuições probabilísticas referentes às duas arquiteturas com melhor *performance*: *ResNet* e *DenseNet*. Esta combinação permitiu a melhoria dos resultados, em cerca de 0.5%.

7.2.1 Comparação com outras abordagens

A Tabela 16 esquematiza os resultados que foram obtidos por outros autores no estudo da *benchmark Colorectal Histopathology*. Os resultados incluídos na tabela são refentes à métrica *accuracy*.

Tabela 16 - Resultados obtidos por outras abordagens

Autores	*AlexNet*	*VGGNet*	*ResNet*	*DenseNet*	Outras arquiteturas/métodos
Kather *et al.*, 2016	-	-	-	-	87.4[22]
Maguolo et.al., 2019	-	95.5	94.9	-	-

[21] Outras bibliotecas como *Theano*, ou *PyTorch* permitem ao utilizador, optar por algoritmos deterministas ou estocásticos, no treino das redes

[22] Utilizou o algoritmo *Support Vector Machines*, considerando a função *Radial Basis Function*, como *kernel*.

Rodrigo, 2018	-	92	-	-	93.84[23]
Lai e Chang 2019	-	-	94.4	-	-
Alinsaif e Lang, 2020	-	-	94.14	95.02	93.78[24]
Bianconi et.al., 2017	83.3	86.0	89.6	-	93.4[25]
Rączkowski *et al.*, 2019	-	-	-	-	92.4[26]

A metodologia proposta promoveu resultados encorajadores, sendo superiores a praticamente todas as abordagens em comparação, inclusive, quando comparada a *performance* das redes em estudo, com outras arquiteturas CNN, nomeadamente: *Inception-v3* ou *MobileNet*. *Maguolo* obteve melhor *performance* na utilização da rede *VGGNet* (+0.9%).

7.3 Resultados - *Skin Mnist*

As Tabelas 17, 18, 19 e 20 ilustram, respetivamente, para as quatro arquiteturas em estudo, a posição convertida da solução *gbest*, ao longo das dimensões otimizadas.

A Tabela 21 enumera os resultados obtidos da avaliação destas soluções. Os valores referentes às métricas indicadas na tabela correspondem ao valor médio obtido, após treino e teste das soluções em três momentos distintos. Os restantes atributos da tabela são: o custo da solução *gbest* (menor *fitness* identificada pelo enxame, calculada com recurso à equação 42), o tempo gasto na sua otimização e ainda o número de parâmetros de treino da solução *gbest*.

As soluções foram avaliadas com recurso às seguintes métricas: *Macro Average F1Score* (MAFS), *Macro Average Recall* (MAR) e *Accuracy* (ACC).

Tabela 17 – Posição da solução *gbest* - *AlexNet*

Hiperparâmetros	**Nº de *single convolution blocks***	**Nº de *stacked convolution blocks***	**Nº inicial de filtros**	***Growth Rate***	**Nº de *Fully-Connected-Layers***	**Nº de Neurónios**	***Batch Size***
AlexNet	3	1	73	23	1	65	12

[23] Utilizou a arquitetura *Inception-v3*.
[24] Utiliza a arquitetura *MobileNet*.
[25] Utilizou a técnica *Improved Opponent Color Local Binary Patterns*;
[26] Utilizou o algoritmo: *Bayesian Convolutional Neural Network* (ARA-CNN), desenvolvido pelo autor.

Tabela 18 – Posição da solução *gbest* - *VGGNet*

Hiperparâmetros	Nº de *stacked convolution blocks*	Nº inicial de filtros	*Growth Rate*	Nº de *Fully-Connected-Layers*	Nº de Neurónios	*Batch Size*
VGGNet	4	71	18	1	61	12

Tabela 19 – Posição da solução *gbest* - *ResNet*

Hiperparâmetros	Nº de conjuntos de *residual blocks*	Nº de *residual blocks*	Nº inicial de filtros	*Growth Rate*	*Batch Size*
ResNet	4	2	56	42	12

Tabela 20 - Posição da solução *gbest* - *DenseNet*

Hiperparâmetros	Nº de *Dense Blocks*	Nº de *composite blocks*	Nº inicial de filtros	*Growth Rate*	*Compression Rate*	*Batch Size*
DenseNet	4	5	59	11	1.0	21

Tabela 21 - Descrição dos resultados obtidos

Arquiteturas	Custo Obtido	Nº de parâmetros de Treino	Duração	Resultados (MAFS/MAR/ACC)
AlexNet	2.85	675, 714	50 horas	65.4%, 63.5%, 81.1%
VGGNet	3.11	1,115,379	42 horas	64.8%, 62.3%, 80.8%
ResNet	2.82	3,439,765	38 horas	66.5%, 64.2%, 81.3%
DenseNet	2.77	339,108	39 horas	67.6%, 65.4%, 81.6%
***Ensemble – 4* Arquiteturas**	-	-	-	68,5 %, 65.2%, 83. 0%
Ensemble – AlexNet + ResNet + DenseNet	-	-	-	69.2%, 66.5%, 83.1%

A complexidade das soluções *gbest*, das quatro arquiteturas, apresentam uma significativa variação entre si, em contraste com o que fora evidenciado anteriormente para a *benchmark Colorectal Histopathology*. Uma possível explicação para este facto pode estar relacionada com a função objetivo utilizada na avaliação das partículas (equação 42). Uma partícula com menor complexidade (parâmetros de treino) só apresenta melhor *fitness* do que as restantes partículas, caso o seu desempenho seja superior ou extremamente próximo (diferença de desempenhos inferior ou igual a 0,01% - seção 6.5.5) das demais partículas. Ou seja, durante a otimização das arquiteturas *VGGNet* e *ResNet*, as partículas não conseguiram identificar soluções menos complexas e com desempenhos superiores ou iguais (extremamente próximos) aos obtidos.

Os valores do hiperparâmetro *batch size* aproximaram-se novamente do limite inferior definido para a sua dimensão. Este acontecimento já era previsível dadas as razões descritas na seção 7.2. Ainda assim, e de acordo com o *batch size* da arquitetura *DenseNet*, é possível constatar que o desempenho dos modelos, neste problema em particular, não está tão dependente da utilização de um *batch size* reduzido, já que a rede *DenseNet* foi a que revelou melhor desempenho e consequentemente apresentou um *batch size* superior às restantes arquiteturas. A técnica *Random Oversampling* pode ter contribuído para a menor sensibilidade dos modelos ao ajuste do *batch size*, já que a sua utilização possibilita a geração de *batches* com distribuições mais equilibradas das amostras das diferentes classes, o que, por sua vez, garante que os gradientes se tornam menos tendenciosos e mais informativos e, consequentemente, os modelos tornam-se mais resistentes à variação dos valores do *batch size* (Johnson e Khoshgoftaar, 2019).

As soluções *gbest* das arquiteturas *AlexNet* e *VGGNet* apresentaram complexidades muito distintas. A rede *AlexNet* é menos complexa, devido ao uso maioritário de *single convolution blocks*. Os *single convolution blocks* são menos exigentes computacionalmente do que os *stacked convolution blocks*, uma vez que não existe sobreposição de camadas convolucionais, reduzindo drasticamente o número de parâmetros de treino da arquitetura. O desempenho das duas redes é semelhante, contudo a rede *AlexNet* demonstrou maior robustez e melhor desempenho, quer em ambiente de otimização, quer em pós otimização, como é possível constatar através da *fitness* das duas soluções e dos resultados (pós-otimização) das métricas MAFS e MAR.

A solução da arquitetura *ResNet* apresentou, de novo, a mesma morfologia da rede *ResNet-18* (Figura 18). O hiperparâmetro *growth rate* aproximou-se do limite superior definido para a sua dimensão, ou seja, a rede tende a revelar melhor *performance*, quando existe um aumento significativo do número de *feature maps* a extrair, entre conjuntos de *residual blocks* (aumenta expressivamente a complexidade da rede). A rede obteve resultados superiores, ainda que ligeiramente, face às arquiteturas *AlexNet* e *VGGNet*.

A arquitetura *DenseNet* aproximou-se da estrutura "original" definida pelos seus autores, sendo composta pelo mesmo número de *dense blocks* e cada *composite block* contribui com um número reduzido de *feature maps* (baixo valor do hiperparâmetro *growth rate*). O hiperparâmetro *compression rate* fixou-se no limite superior definido para a sua dimensão, ou seja, a rede tende a apresentar melhor *performance*, quando a profundidade do *output* de um *dense block* permanece inalterada (*compression rate* igual a 1 - *DenseNet-B*). Apesar de não

existir uma "compressão" da profundidade do *output* dos *dense blocks*, a complexidade da rede manteve-se reduzida, devido, sobretudo, ao baixo número de *feature maps* introduzidas por cada *composite block*. A rede obteve, assim, vantagem da comunicação direta e da reutilização de *features* entre *composite blocks*, resultando numa rede com complexidade inferior e desempenho superior face às restantes arquiteturas.

Na Tabela 21 foram ainda incluídos os resultados de dois modelos *ensemble*. O 1º modelo corresponde à média das distribuições probabilísticas das quatro arquiteturas. Os resultados obtidos apresentam uma ligeira melhoria, face ao desempenho individual das arquiteturas. O 2º modelo resulta da exploração de vários *ensembles*, considerando a combinação das distribuições probabilísticas de 2 e de 3 modelos em simultâneo, sendo que o melhor desempenho obtido, diz respeito à média aritmética das distribuições das seguintes arquiteturas: *AlexNet*, *ResNet* e *DenseNet*.

7.3.1 Comparação com outras abordagens

A Tabela 22 compara os resultados obtidos com abordagens de outros autores.

Contrariamente à *benchmark Colorectal Histopathology*, os resultados obtidos estão distantes dos valores atingidos por outros autores, sendo várias as razões que estão implicitamente relacionadas com esta diferença de resultados, nomeadamente:

- Redução muito expressiva da dimensão original das imagens. As imagens foram redimensionadas de 600*450 pixéis para 128*128 pixéis. Alguns dos estudos incluídos na tabela consideraram como dimensões de *input* 224*224 ou 299*299. *Gessert* por exemplo, não reduziu as dimensões das imagens. A redução excessiva das dimensões diminui a qualidade das imagens, implicando a perda de um elevado conjunto de informação, o que, por sua vez, reduz a probabilidade de serem identificadas características relevantes nos dados. Contudo, a utilização de dimensões superiores às utilizadas era insuportável, dadas as limitações computacionais e de tempo existentes;
- A não utilização de conjuntos de dados adicionais. O *Skin Mnist* fez parte de um desafio promovido, em 2018, pela instituição *International Skin Imaging Collaboration* (ISIC). Na época, os participantes tinham acesso a dois conjuntos de dados complementares: validação[27] e teste. Contudo, os utilizadores apenas tinham acesso às imagens, e não aos seus *targets* (em ambos os *dataset's*). Os utilizadores tinham de enviar as previsões para a plataforma[28] que ostentava o "desafio", sendo que a instituição era responsável por exibir os resultados (obtidos no *dataset* de teste) dos concorrentes na plataforma. A utilização adicional destes dois conjuntos de dados, permite que o conjunto constituído pelas 10015 imagens (Anexo B – este conjunto foi utilizado para treino+validação+teste), seja utilizado unicamente para o treino da rede, aumentando

[27] Conjunto composto apenas por 153 imagens, e usado para obtenção de *feedback* rápido dos algoritmos desenvolvidos, através do envio das distribuições probabílisticas para a plataforma.
[28] https://challenge2018.isic-archive.com

assim o desempenho dos modelos, já que existe um maior número de dados disponíveis para aprendizagem. *Gessert* por exemplo, recorreu ainda à utilização de dados externos à *benchmark Skin Mnist*, na tentativa de melhorar a *performance* obtida.

- A não segmentação das imagens. Algumas das abordagens em comparação segmentarem previamente as imagens, extraindo a forma e as características relevantes de cada lesão. A 1ª tarefa do desafio ISIC era precisamente segmentar as imagens, daí a sua utilização por algumas das propostas incluídas na tabela 22. Esta tarefa não foi realizada, uma vez que "foge" ao âmbito do projeto;

Ainda assim, as soluções identificadas pelo PSO revelaram resultados adequados face às limitações existentes. Os resultados obtidos são superiores, quando comparados com abordagens que não utilizam dados externos, não segmentam as imagens ou reduzem excessivamente as dimensões das amostras.

É expectável que com o aumento do volume de dados e das dimensões das amostras, os resultados possam melhorar significativamente.

Tabela 22 – Resultados obtidos por outras abordagens

Autores	***AlexNet***	***VGGNet***	***ResNet***	***DenseNet***	**Outras arquiteturas**	***Ensemble***
Gessert *et al.*, 2018	-	-	MAR: 77.9	MAR: 78.9	MAR: 81.7[29]	MAR: 85.1
Tripathy *et al.*, 2020	-	-	-	-	MAFS: 51.0[30]	-
Shahata et.al., 2020	-	-	-	-	MAFS: 58.0[31]	-
Milton, 2019	-	-	-	-	MAR: 76.0[32]	MAR: 73.0
More *et al.*, 2020	-	-	-	-	MAFS: 47.0[33]	-
Barata e Santiago, 2020	-	MAFS: 75.0	-	-	-	-
Zhuang et.al., 2020	-	-	ACC: 84.0	-	ACC: 88.0[34]	ACC: 90.5
Pal et.al., 2018	-	-	MAR: 77.3	MAR: 69.8	MAR: 59.7[35]	MAR: 77.5

[29] Utilização da Arquitetura *SENet154*.
[30] Utilizou a arquitetura *MobileNet*.
[31] Definiu a sua própria arquitetura.
[32] Utilizou a arquitetura *PNASNet-5-Large*.
[33] Definiu a sua própria arquitetura.
[34] Utilizou a arquitetura *SENet154*.
[35] Utilizou a arquitetura *MobileNet*.

7.4 Resultados *Breast Histopathology*

As Tabelas 23, 24, 25 e 26 expõem, respetivamente, para as quatro arquiteturas em estudo, a posição convertida da solução *gbest*, ao longo das dimensões otimizadas.

A Tabela 27 esquematiza os resultados obtidos e referentes à avaliação destas soluções. Os resultados de cada métrica avaliada correspondem ao valor médio obtido, após treino e teste das soluções em três momentos distintos. Os restantes atributos da tabela são: o custo da solução *gbest* (menor *fitness* identificada pelo enxame, calculada com recurso à equação 42), o tempo gasto na sua otimização e ainda o número de parâmetros de treino da solução *gbest*.

As soluções foram avaliadas com recurso às seguintes métricas: *Macro Average F1-Score* (MAFS), *Macro Average Recall* (MAR) e *Accuracy* (ACC).

Tabela 23 - Posição da solução *gbest* - *AlexNet*

Hiperparâmetros	**Nº de *single convolution blocks***	**Nº de *stacked convolution blocks***	**Nº inicial de filtros**	***Growth Rate***	**Nº de *Fully-Connected-Layers***	**Nº de Neurónios**	***Batch Size***
AlexNet	1	3	90	40	2	49	32

Tabela 24 - Posição da solução *gbest* - *VGGNet*

Hiperparâmetros	**Nº de *stacked convolution blocks***	**Nº inicial de filtros**	***Growth Rate***	**Nº de *Fully-Connected-Layers***	**Nº de Neurónios**	***Batch Size***
VGGNet	3	89	39	2	45	64

Tabela 25 - Posição da solução *gbest* - *ResNet*

Hiperparâmetros	**Nº de conjuntos de *residual blocks***	**Nº de *residual blocks***	**Nº inicial de filtros**	***Growth Rate***	***Batch Size***
ResNet	3	3	104	21	64

Tabela 26 - Posição da solução *gbest* - *DenseNet*

Hiperparâmetros	**Nº de *Dense Blocks***	**Nº de *composite blocks***	**Nº inicial de filtros**	***Growth Rate***	***Compression Rate***	***Batch Size***
DenseNet	4	6	39	18	0.9	65

Tabela 27 - Descrição dos resultados obtidos

Arquiteturas	Custo Obtido	Nº de parâmetros de Treino	Duração	Resultados (MAFS/MAR/ACC)
AlexNet	0.913	1, 269, 615	5 dias e 4 horas	90.2%, 89.7%, 92.1%
VGGNet	0.918	1, 139, 219	4 dias e 4 horas	90.5%, 89.6%, 92.4%
ResNet	0.944	2, 586, 006	7 dias e 16 horas	90.4%, 90.2%, 92.2%
DenseNet	1.013	835, 895	6 dias e 15 horas	89.8%, 89.4%, 91.7%
***Ensemble* – 4 Arquiteturas**	-	-	-	91.1%, 90.7%, 92.9%

Tal como fora evidenciado para as *benchmarks* anteriores, os valores do hiperparâmetro *batch size* aproximaram-se novamente do limite inferior definido para a sua dimensão (as razões desta tendência, já foram descritas previamente). Foi considerado um limite inferior distinto para as diferentes arquiteturas, uma vez que, dado o elevado número de amostras existentes, a utilização de um *batch size* reduzido exige elevado tempo de execução. Sendo assim, para a rede *AlexNet* foi definido o limite inferior = 32, enquanto que para as restantes arquiteturas foi aumentado o seu valor para 64 (seção 6.5.4). Não foi definido nenhum critério para a escolha da arquitetura que teria o limite (inferior) mais baixo. O principal objetivo da variação do limite inferior desta dimensão, compreende a análise de impacto que o hiperparâmetro *batch size* tem, na resolução de problemas com um maior volume de dados, como é o caso do *dataset Breast Histopathology*.

Analisando a *fitness* das soluções *gbest*, referentes às quatro arquiteturas, é possível concluir que todas as soluções apresentaram um custo muito semelhante entre si (com exceção da rede *DenseNet*). Ou seja, apesar da rede *AlexNet* efetuar o dobro das atualizações aos parâmetros da rede numa *epoch*, a sua convergência e o seu desempenho (ao fim das 15 *epochs*) foram idênticos às restantes arquiteturas. Em conclusão, a variação ponderada do hiperparâmetro *batch size* não promoveu alterações acentuadas na *performance* dos modelos, neste problema em particular e possivelmente em outros problemas compostos por um elevado número de amostras (e com uma desproporção de amostras, semelhante, entre as classes). Contudo, é necessário ter em conta que a diferença dos *batch sizes* analisados não é severa (32 para a rede *AlexNet* e 64 para as restantes arquiteturas), caso fosse superior a tendência seria muito provavelmente alterada, já que o número de atualizações efetuadas aos pesos da rede seria ainda mais díspar, e possivelmente essa alteração já teria impacto no desempenho dos modelos.

A utilização da técnica *random undersampling* pode ter contribuído para a redução do impacto do *batch size* no desempenho e convergência dos modelos. Uma das limitações do *Breast Histopathology* prende-se com a excessiva presença de amostras da classe "*No IDC*" ao longo das *batches*, ou seja, a baixa frequência de amostras da classe "*With IDC"* torna os gradientes

pouco informativos e tendenciosos, o que promove modelos com baixa *performance* e com menor rapidez de convergência. A técnica *random undersampling* aumenta o equilíbrio da distribuição das duas classes, permitindo que cada *batch* contenha uma distribuição mais equilibrada de amostras de ambas as classes, reduzindo assim o impacto que a variação do *batch size* tem na *performance* dos modelos (os modelos tornam-se mais resistentes ao uso de diferentes valores do *batch size*, sem que isso afete o seu desempenho). Esta abordagem garante que os pesos da rede são atualizados, em conformidade com a maior probabilidade de frequência da classe menos balanceada ao longo das *batches*, melhorando assim a capacidade de generalização dos modelos, especialmente, para a classe com menor distribuição de amostras, *"With IDC"* (Johnson e Khoshgoftaar, 2019).

As Tabelas 23 e 24 demonstram a extrema semelhança entre a complexidade das soluções das arquiteturas *AlexNet* e *VGGNet*, sendo a profundidade a única diferença assinalável entre estas. A maior profundidade da rede *AlexNet* permite-lhe a extração de *features* mais complexas. Este fator pode estar diretamente relacionado com a menor sensibilidade da rede *VGGNet* na correta classificação da classe menos balanceada. A superioridade das métricas *accuracy* e *macro average F1Score*, derivam, sobretudo, da ligeira tendenciosidade que a rede *VGGNet* revela na previsão da classe mais balanceada ("*No IDC*"). A maior densidade (maior número de camadas convolucionais e maior número de reduções realizadas às dimensões iniciais das amostras) da rede *AlexNet* permite-lhe a distinção mais precisa e com menor variação dos dois tecidos do problema, já que a rede extrai características mais informativas e com maior detalhe dos dados (Zeiler e Fergus, 2014). A rede *AlexNet* demonstrou maior desempenho na previsão da classe menos balanceada, visível através da métrica MAR.

A estrutura da solução da arquitetura *ResNet*, ao contrário do sucedido para as *benchmarks* anteriores, não segue a morfologia da rede *ResNet-18*, sendo composta por três conjuntos de *residual blocks*, que, por sua vez, agregam três *residual blocks*. As baixas dimensões das amostras do problema (50*50) podem ter contribuído para a utilização de um menor número de conjuntos de *residual blocks*, dada a menor necessidade de redução das dimensões iniciais das amostras. Por sua vez, o aumento da complexidade da rede é compensado pela utilização de um maior número de *residual blocks*.

A otimização da rede *ResNet* ocorreu durante um longo período. Os limites abrangentes de algumas dimensões (número inicial de filtros, *growth rate* e número de *residual blocks*[36]) e o elevado número de dados de treino foram as causas deste atraso na conclusão do algoritmo. Limites amplos proporcionam a criação de soluções complexas e profundas e a par do elevado número de amostras de treino provocam um aumento do tempo despendido no treino das soluções (dilação na extração das *features*, na atualização dos parâmetros da rede e em operações *sum element wise*). A convergência prematura do algoritmo para uma solução profunda e complexa (Apêndice E – as partículas convergiram prematuramente para a solução apresentada na Tabela 25 e abordada aqui) promoveu, também, o significativo aumento do

[36] Não é um limite propriamente amplo, mas a não utilização de *residual blocks* em forma de *bottleneck*, faz com que três *residual blocks* em cada conjunto, possibilitem (em concordância com outros parâmetros: *growth rate* e número inicial de filtros) a criação de redes manifestamente complexas e profundas, o que, a par de um elevado número de amostras provoca um aumento drástico do tempo de otimização das soluções.

tempo de otimização. Em problemas que reúnam um número significativo de amostras deve ser considerada a estrutura do *residual block* em forma de *bottleneck* (*ResNet*-50/101/152, Figura 19), já que reduz a complexidade da rede e requer menor tempo de treino.

Em contraste com o que fora observado previamente, a rede *DenseNet* revelou-se menos eficaz na resolução deste problema, em comparação com as restantes arquiteturas em estudo. A menor eficácia da rede deriva, sobretudo, da provável "queda" precoce das partículas para um ótimo local. O limite superior estabelecido para a dimensão "número de *composite blocks*" (Tabela 10) pode, também, ter limitado a *performance* do PSO, visto que esta dimensão restringe a profundidade e a complexidade das soluções (a par de outras dimensões, como: "número de *dense blocks*", *growth* e *compression rate*). Dado o elevado número de amostras do problema, seria adequado que o limite superior da dimensão "número de *composite blocks*" fosse mais elevado, já que permitiria a exploração de soluções com maior profundidade e/ou complexidade e, eventualmente, permitiria a identificação de soluções com melhores desempenhos. Este facto é sustentado, em harmonia com a proposta original de *Huang*, que demonstrou que a arquitetura *DenseNet*-B(C) tende a revelar maior eficácia, na resolução de problemas com elevado número de dados (*ImageNet Cifar*), à medida que a profundidade da rede aumenta. A capacidade de generalização da rede pode, eventualmente, ser melhorada com o uso de *data augmentation* mais severa (maior número de alterações). Contudo, importa referir que, quer o aumento do limite superior da dimensão "número de *composite blocks"*, quer o uso de *data augmentation* mais severa, provocam o aumento do tempo de treino da rede, mas é expectável a melhoria do seu desempenho.

Não foi estabelecido um limite superior elevado para a dimensão "número de *composite blocks*", de modo a garantir que a otimização ocorria em tempo útil aceitável. O consequente aumento do limite proporcionaria um aumento drástico do tempo de execução do algoritmo, inviabilizando a obtenção e a análise dos resultados em tempo útil admissível.

O tempo de otimização da arquitetura *DenseNet* foi longo. As razões que promoveram esta demora são idênticas às causas evidenciadas na análise da rede *ResNet*: limites amplos das dimensões, elevado número de amostras e convergência das partículas para soluções profundas e complexas. O período de otimização da rede *DenseNet* foi inferior ao providenciado pela arquitetura *ResNet*, muito provavelmente, devido à menor complexidade das soluções para as quais as partículas foram convergindo ao longo da otimização. A utilização de *composite blocks* em forma de *bottleneck*, o hiperparâmetro *compression rate* < 1 e a concatenação de *features* (a concatenação é uma operação que requer baixo esforço computacional, ao contrário do somatório) promoveram, também, a redução do tempo de execução do algoritmo.

A técnica *ensemble* revelou-se novamente eficaz na redução da variância dos modelos, sendo obtidos resultados ligeiramente superiores aos alcançados, individualmente, pelas diferentes arquiteturas.

7.4.1 Comparação com outras abordagens

A Tabela 28 esquematiza os resultados alcançados, por outros autores, no estudo da *benchmark Breast Histopathology*.

Tabela 28 - Resultados obtidos por outras abordagens

Autores	*AlexNet*	*VGGNet*	*ResNet*	*DenseNet*	Outras Arquiteturas
Cruz-Roa *et al.*, 2014	-	-	-	-	F1Score: 71,8[37] MAR: 84,2
Janowczyk e Madabhushi, 2016	F1Score: 76,5 MAR: 84,7	-	-	-	-
Norris, 2019	-	-	-	-	F1Score: 82.0[38]
Alghodhaifi et.al., 2019	-	-	-	-	F1Score: 76.0[39] MAR: 87.1
Asare et.al., 2020	-	-	-	-	F1Score: 87.5[40] ACC: 89.9
Mohapatra et.al., 2019	-	-	-	-	F1Score: 77.5[41] ACC: 81.0
Romero et.al., 2019	-	-	-	-	F1Score: 89.7[42] MAR: 89.0

O comparativo confirma a primazia dos resultados obtidos, face aos valores alcançados por outros autores. A significativa superioridade das métricas MAR e MAFS, demonstra a maior sensibilidade das soluções obtidas, na correta previsão das amostras da classe menos balanceada "*With IDC*".

[37] Utilizou uma arquitetura CNN definida por si, composta por 2 únicas camadas convolucionais, kernel *8*8*.
[38] Definiu uma arquitetura própria, "apelidada" de *CancerNet*, constituída por 31 camadas.
[39] Arquitetura semelhante à rede *VGGNet*, mas com ligeiras alterações.
[40] Arquitetura definida pelo próprio.
[41] Arquitetura definida pelo próprio.
[42] Variante da arquitetura *Inception-V3*.

CAPÍTULO 8
DESENVOLVIMENTO DA PLATAFORMA *WEB*

Este capítulo apresenta a aplicação *DiagnosisViewer*. *DiagnosisViewer* é uma aplicação *web*, que facilita o processo de divulgação e o acesso a diferentes soluções (modelos convolucionais), na resposta ao diagnóstico de amostras médicas.

Seguidamente é contextualizada a aplicação desenvolvida, sendo descritas as etapas que compuseram o seu desenvolvimento, nomeadamente: as funcionalidades da aplicação, a metodologia de diagnóstico de amostras, o padrão de comunicação entre as diferentes camadas da aplicação e a base de dados utilizada.

8.1 Descrição do *DiagnosisViewer*

O *DiagnosisViewer* é uma aplicação *web* desenvolvida com recurso à *framework Django*. É uma aplicação simples, que representa um complemento a todo o trabalho desenvolvido e descrito ao longo dos capítulos anteriores.

A utilização do *DiagnosisViewer* possibilita o acesso e a divulgação de modelos convolucionais entre a comunidade (utilizadores da aplicação). Os utilizadores têm a oportunidade de partilharem os seus modelos, na resposta a diferentes *benchmarks* médicas. As *benchmarks* são também introduzidas pelos utilizadores.

O *DiagnosisViewer* possibilita a submissão de modelos CNN, criados e treinados com recurso a diferentes configurações, nomeadamente: adaptados a diferentes dimensões das amostras, a diferentes tamanhos do *dataset* (uso de um maior ou menor número de amostras), a diferentes resultados do mapeamento (para valores inteiros) das categorias referentes às classes ou a diferentes abordagens de treino (diferentes optimizadores, diferentes valores dos hiperparâmetros, *etc*).

A principal característica do *DiagnosisViewer* é a sua funcionalidade de diagnóstico interativo. Ao contrário das aplicações descritas na Seção 2.3, o *DiagnosisViewer* possibilita a "criação" de *benchmarks* (a criação de uma *benchmark* corresponde à indicação das suas principais características) e a submissão de modelos convolucionais na plataforma, sem recurso à implementação de código, sendo apenas necessário indicar um conjunto de informações sobre os modelos e as *benchmarks* (bem como a submissão do ficheiro HDF5 referente ao modelo). Esta abordagem garante que os utilizadores podem explorar, facilmente e rapidamente, o diagnóstico de amostras para qualquer *benchmark* presente na plataforma e recorrendo a qualquer modelo disposto na aplicação. A utilização da plataforma abrange todos os tipos de utilizadores e não são necessários quaisquer conhecimentos informáticos para o seu uso.

A tarefa de diagnóstico é concluída em poucos segundos, uma vez que os modelos submetidos na plataforma foram treinados previamente (antes da submissão), possibilitando assim o acesso rápido ao diagnóstico de uma amostra e não é requerida a implementação de código por parte dos utilizadores, ao contrário de outras plataformas que exigem o desenvolvimento de código.

8.2 Funcionalidades da aplicação

Tal como referido previamente, a aplicação tem um âmbito bem definido e disponibiliza um número restrito de funcionalidades, sendo o seu objetivo concreto, permitir o diagnóstico interativo de amostras, através do uso de modelos convolucionais disponibilizados na plataforma. O diagrama de casos de uso ilustrado na Figura 32 esquematiza as principais funcionalidades da aplicação.

O acesso ao *DiagnosisViewer* (e às suas funcionalidades) requer autenticação, ou seja, os utilizadores necessitam de se registar no sistema. Após o seu registo e consequente autenticação, os mesmos têm acesso a todas as funcionalidades, com exceção da tarefa de gestão de utilizadores, que é da responsabilidade do administrador. Estão ainda limitados à gestão de *dataset's* e de modelos, isto é, os utilizadores podem alterar e apagar, somente, os *datasets* e modelos que foram submetidos por si, estando assim impedidos de apagar e alterar, conteúdo disponibilizado pelos outros utilizadores (condição representada nas notas do diagrama de casos de uso). O administrador pode alterar e apagar qualquer conteúdo disponibilizado na *app*.

O caso de uso "Efetua Diagnóstico" representa a funcionalidade principal da *app*. A abordagem proposta para a implementação desta funcionalidade está descrita na Seção 8.5.

Por último, o caso de uso "Divulga Histórico" consiste na divulgação de informação "histórica" sobre a atividade dos utilizadores, nomeadamente *top* 3: *dataset's* mais utilizados pelo utilizador autenticado e *dataset's* e modelos mais utilizados pela comunidade. Esta informação é disponibilizada, em conformidade com as últimas alterações registadas na *app* (pedido, assíncrono, executado sempre que é recarregada a página principal – Figura 53).

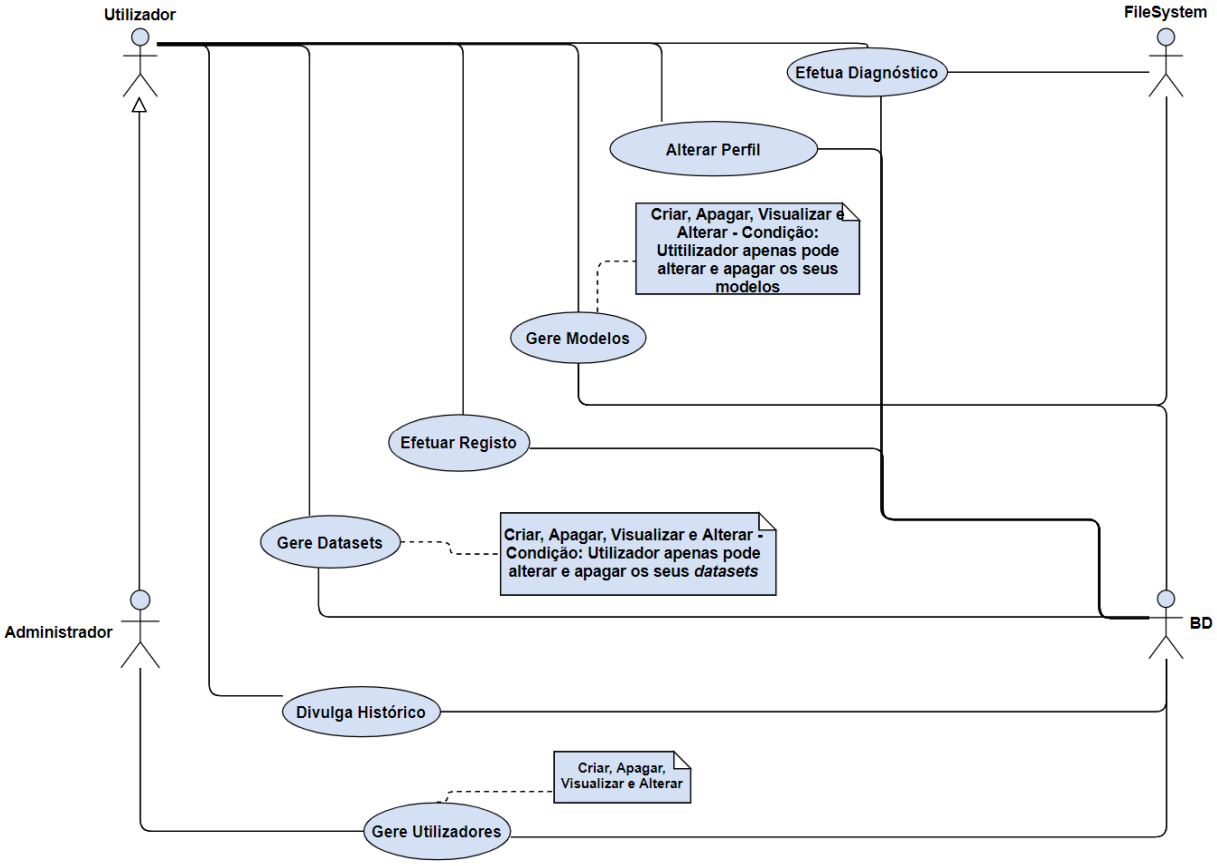

Figura 32 - Diagrama de Casos de Uso do *DiagnosisViewer*

8.3 Padrão de comunicação entre camadas

A *framework Django* utiliza, por defeito, o padrão de desenho *Model-View-Template* (MVT). Este padrão divide a arquitetura da aplicação em três camadas: *Templates*, Vistas e Modelos. *Template* corresponde à camada de interação com os utilizadores e promove a renderização dos dados através de páginas HTML (*HyperText Markup Language*). *Model* define a estrutura dos dados, em concordância com as entidades representadas numa base de dados, e a lógica de alteração e validação dos mesmos. *View* estabelece uma "ponte" entre as camadas citadas anteriormente, ou seja, é responsável por receber as ações do utilizador e definir o comportamento da aplicação, através de pedidos realizados aos modelos para consequente realização de operações de criação, remoção, alteração ou seleção de dados. Como resposta ao utilizador, a vista exibe uma nova página *web* (ou atualiza a página atual), que apresenta o resultado das alterações pedidas.

Sempre que um cliente envia um pedido HTTP (*Hypertext Transfer Protocol*), o mesmo é reencaminhado para a vista que está associada ao URL (*Uniform Resource Locator*) do pedido (responsabilidade do *Django*). Esta associação é definida recorrendo a um ficheiro *urls.py*, que estabelece para cada URL, a vista a ser chamada após o pedido do cliente. A vista é responsável, depois, por responder ao pedido.

A Figura 33 ilustra o mecanismo de separação e de comunicação entre camadas, descrito nos parágrafos anteriores e utilizado na implementação da *app*.

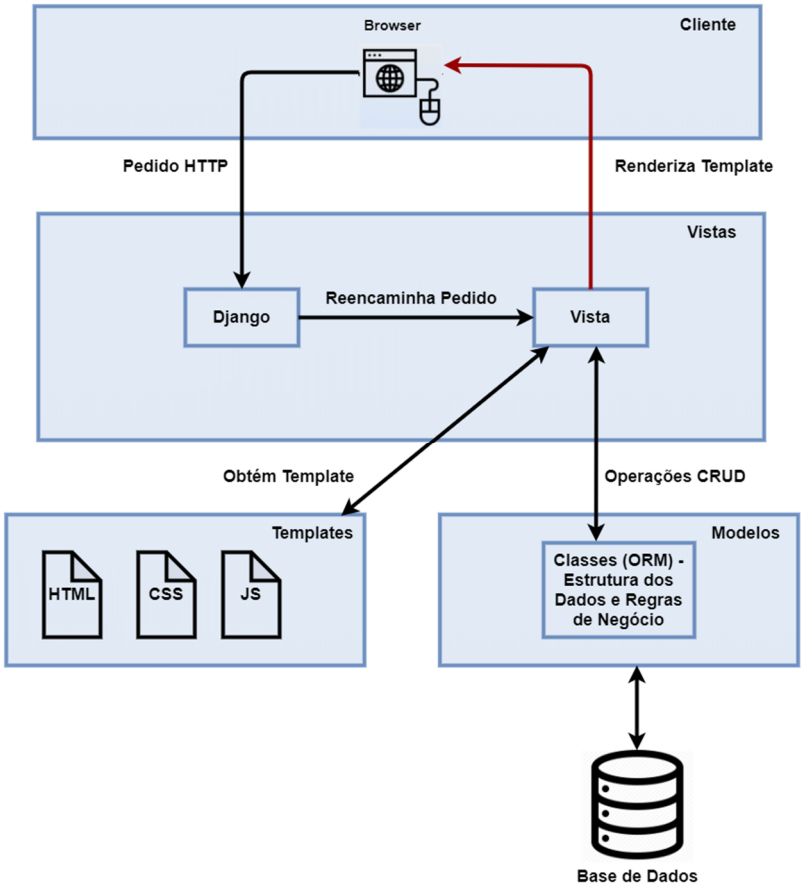

Figura 33 - Comunicação entre Camadas – MVT

8.4 Base de Dados

A Base de Dados (BD) definida é simples e está em conformidade com o âmbito restrito da aplicação. Foi utilizado o Sistema de Gestão de Base de Dados *PostgreSQL* na modelação da BD. A Figura 34 ilustra o diagrama de Entidade-Relacionamento proposto.

O número de entidades da BD é reduzido e está diretamente relacionado com a natureza explícita das funcionalidades a implementar (Seção 8.2), sendo identificadas as seguintes entidades: "*users*", "*dataset*", "*model*" e "*history*".

A entidade "users" refere todos os utilizadores registados na plataforma. "*Dataset*" descreve as várias *benchmarks* adicionadas pelos utilizadores. A tabela "*model*" apresenta uma dupla função, isto é, representa a configuração dos modelos submetidos pelos utilizadores na plataforma e armazena a localização, relativa ao sistema de ficheiros, do consequente ficheiro binário (HDF5 – modelos CNN criados com recurso à biblioteca *Keras*) submetido pelo utilizador e a utilizar no processo de diagnóstico (atributo *model_path*). Por último, a tabela "*history*" traduz o histórico das tentativas de diagnóstico realizadas pelos utilizadores e resulta do relacionamento entre as entidades citadas anteriormente.

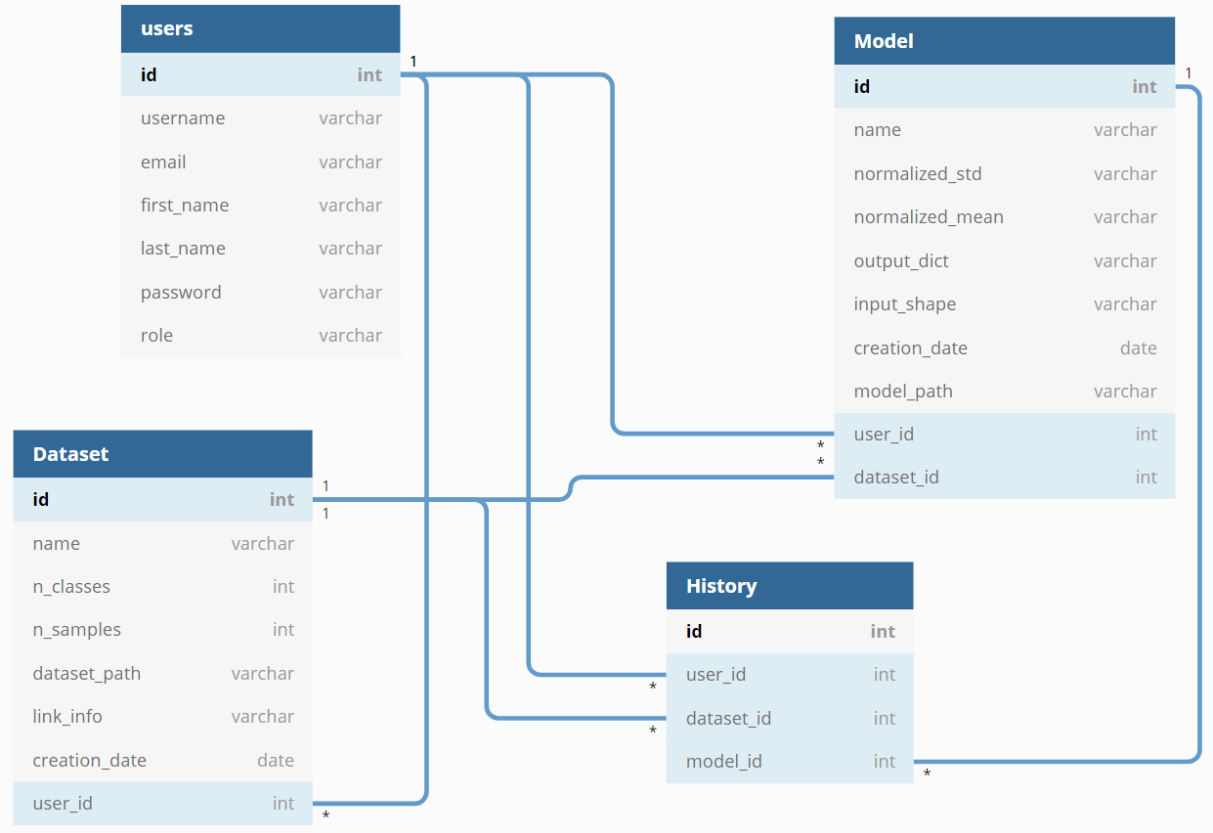

Figura 34 - Modelo Entidade Relacionamento

8.5 Descrição da funcionalidade: "Efetua Diagnóstico"

A implementação da funcionalidade "Efetua Diagnóstico" envolve alguma complexidade, sendo assim foi construído um diagrama de sequência que define, sucintamente, o fluxo de passos subjacentes ao desenvolvimento desta funcionalidade. A Figura 35 exibe o diagrama de sequência criado.

No diagrama é incluído o fluxo de interação entre as várias camadas da arquitetura (receção do pedido, tratamento e consequente envio da resposta). As operações realizadas à BD estão descritas no diagrama e são representadas através de interações com as entidades, concretas, com as quais a vista interage: "*model*" e "*history*".

A página *web* de diagnóstico (Figura 67) dispõe de um formulário, que solicita ao utilizador a informação necessária para realizar o diagnóstico de uma amostra, nomeadamente: seleção da *benchmark* condizente com a amostra a diagnosticar, escolha de um dos modelos CNN disponíveis (para a *benchmark* selecionada) e submissão da amostra a diagnosticar (formatos aceites: png, jpeg, tiff, svg, bmp, dib ou webp). Este pedido é redirecionado para a vista *PredictView*, que é responsável por dar resposta ao pedido (o *Django* internamente efetua o *redirect*). Seguidamente, são descritos os passos executados pela vista *PredictView* no tratamento do pedido e consequente envio da resposta ao utilizador.

Após a submissão do formulário, a vista toma conhecimento dos dados submetidos pelo utilizador. A imagem submetida é temporariamente armazenada em disco, de modo a permitir a leitura e a transformação da imagem para uma matriz de *unsigned bytes*, bem como a consequente aplicação das etapas de pré-processamento na amostra, sendo efetuado ainda um pedido à BD para a obtenção do modelo (objeto) selecionado pelo utilizador.

Seguidamente, a amostra é transformada numa matriz de *unsigned bytes*, sendo constituída pelos códigos das cores referentes a cada pixel da imagem (a amostra é representada, de agora em diante, por esta matriz). Procede-se, ainda, ao redimensionamento da amostra (caso haja necessidade), tendo em conta as dimensões (comprimento e largura) iniciais do modelo CNN, que foram indicadas pelo utilizador no ato de inserção do modelo na plataforma (atributo *input_shape*). Esta abordagem permite o diagnóstico de amostras com dimensões distintas, uma vez que as suas dimensões são ajustadas às dimensões iniciais do modelo selecionado.

De modo, a garantir a efetividade do processo de diagnóstico, é necessário transformar a amostra, em conformidade com a distribuição do conjunto de treino, que fora utilizado pelo modelo na sua aprendizagem. De momento, a aplicação considera que os modelos submetidos na plataforma, foram treinados com recurso a amostras, que foram transformadas recorrendo à técnica *z-score*. Na transformação da amostra são considerados os valores médios e de desvio padrão, de cada canal de profundidade, do conjunto de treino utilizado pelo modelo (a *app* suporta modelos com 1 canal de profundidade – *grayscale*, ou 3 canais – RGB). Estes valores são indicados, pelo utilizador, no ato de submissão do modelo na plataforma (atributos *normalized_mean* e *normalized_std*).

As etapas de manipulação e transformação da amostra já se encontram devidamente concluídas, posto isto, é iniciado o processo de diagnóstico. Numa primeira fase, é necessário carregar o modelo selecionado pelo utilizador do sistema de ficheiros (obter o ficheiro HDF5 correspondente ao modelo selecionado), seguindo-se a classificação da amostra submetida. Os resultados obtidos são descritos em conformidade com um dicionário, que foi definido pelo utilizador no ato de inserção do modelo na *app*. O dicionário representa o resultado, que foi considerado pelo modelo em ato de treino (antes da submissão, etapa pré *one-hot encoding*), do mapeamento das (categorias das) classes do problema para valores inteiros (atributo *output_dict*). A utilização deste dicionário é crucial, uma vez que diferentes modelos, podem considerar uma abordagem distinta, de mapeamento (para valores inteiros) das classes do problema, originando assim divergências entre a interpretação dos resultados e a sua exibição.

A amostra diagnosticada é apagada do sistema de ficheiros, após a conclusão da tarefa, garantindo, assim, que a confidencialidade dos utilizadores é respeitada. É efetuado um pedido à BD para inserção da atividade de diagnóstico na entidade "*history*". Esta abordagem proporciona a geração contínua de um histórico de diagnósticos.

A página *web* "Efetua Diagnóstico" é novamente renderizada, sendo exibido o diagnóstico obtido e referente à amostra submetida (Figura 69 – exibição das distribuições probabilísticas obtidas para cada classe).

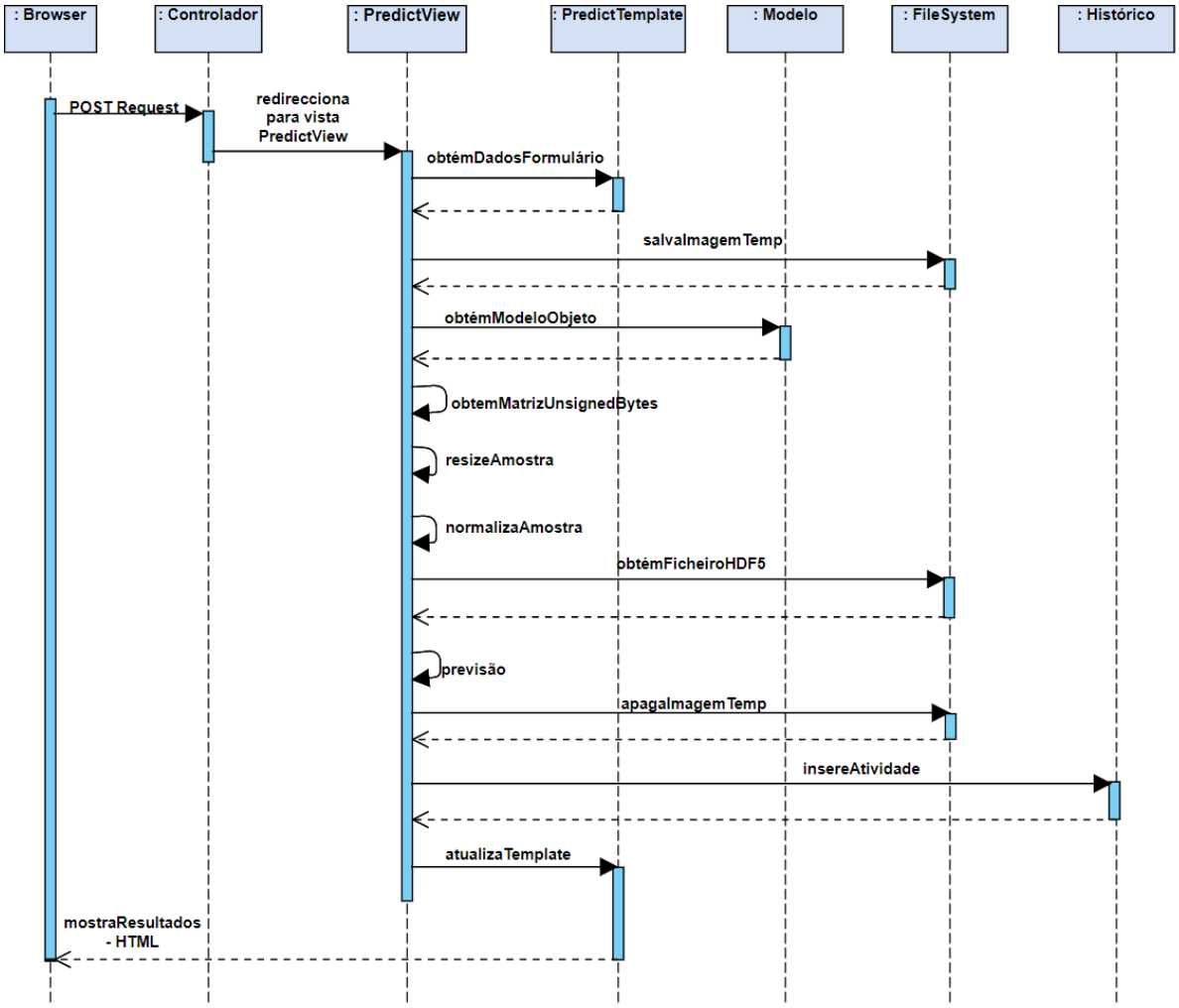

Figura 35 - Diagrama de Sequência da Funcionalidade "Efetua Diagnóstico"

8.6 Resultado da aplicação

A *app DiagnosisViewer*[43] está acessível através de um repositório no *github*.

As páginas *web* implementadas estão descritas e ilustradas ao longo do Apêndice D. A área de Administrador foi gerada automaticamente pela *framework*.

As bibliotecas utilizadas pela aplicação estão descritas no ficheiro *requirements.txt*. Durante o desenvolvimento da *app* recorreu-se à utilização de *Javascript/jQuery*, AJAX, *bootstrap* (apenas para renderização de *modal's*) e fontes externas. As dependências necessárias à sua utilização estão disponibilizadas na pasta *static*.

O *layout* das páginas *web* está em concordância com o estilo padrão utilizado pela *framework Django*. A aplicação é responsiva e adapta-se a várias resoluções e navegadores *web*.

[43] https://github.com/bundasmanu/ProjetoMestrado

CAPÍTULO 9
CONCLUSÕES E TRABALHO FUTURO

Este capítulo apresenta as principais conclusões do trabalho realizado, sendo ainda identificadas algumas abordagens a considerar futuramente, com vista à melhoria do *DiagnosisViewer*, da metodologia desenvolvida e dos resultados obtidos.

9.1 Conclusões

O principal objetivo do projeto visava a exploração do algoritmo *Particle Swarm Optimization*, na otimização dos principais hiperparâmetros de quatro conhecidas arquiteturas de redes convolucionais: *AlexNet*, *VGGNet*, *ResNet* e *DenseNet*.

A eficiência do algoritmo foi avaliada em problemas de domínio médico, precisamente, na classificação de tecidos e lesões dermatológicas associadas a diferentes tipos de cancro: mama, colorretal e pele.

As técnicas utilizadas para atenuação das limitações das *benchmarks* (baixo número de amostras, classes não balanceadas, elevada complexidade) proporcionaram uma melhoria significativa da *performance* dos modelos. Este facto é particularmente visível no *dataset Breast Histopathology*, onde a utilização das técnicas *random undersampling*, *cost sensitive learning* e *data augmentation* promoveram uma adequada generalização dos modelos, permitindo a obtenção de resultados superiores face a outras abordagens em comparação. A utilização de *Data Augmentation* possibilitou a obtenção de excelentes resultados no *dataset Colorectal Histopathology*. Relativamente à *benchmark Skin Mnist* é necessário proceder à realização de mais testes, nomeadamente, ao aumento das dimensões das amostras (aumento do número de *features*) ou à utilização de um maior número de dados, de modo a poder aferir as vantagens que a utilização das técnicas de *random oversampling, cost sensitive learning* e *data augmentation* propiciam. Ainda assim, os resultados alcançados no problema *Skin Mnist* foram superiores, quando comparados com abordagens similares à proposta definida, isto é, abordagens que não utilizam conjuntos de dados adicionais, não recorrem à segmentação das imagens e reduzem expressivamente as dimensões das amostras.

Os resultados obtidos demonstraram que as redes *AlexNet* e *VGGNet* conseguem apresentar resultados muito satisfatórios, superando inclusive arquiteturas "teoricamente mais eficientes", como por exemplo, a rede *Inception-v3* (utilizada por outros autores). As arquiteturas *ResNet* e *DenseNet* revelaram-se globalmente mais robustas, apresentando resultados ligeiramente superiores aos alcançados pelas redes *AlexNet* e *VGGNet* (exceção rede *DenseNet* – problema *Breast Histopathology*). O uso da técnica *ensemble* (proposta) permitiu a obtenção de resultados ligeiramente superiores, face aos alcançados individualmente pelas diferentes arquiteturas.

O PSO demonstrou ser um algoritmo extremamente eficaz, permitindo a obtenção de resultados superiores comparativamente a outras abordagens, em duas das três *benchmarks* analisadas. A utilização de um baixo número de iterações e de partículas não foi impedimento para a identificação de "boas" soluções, aliás, o algoritmo apresentou rápida convergência quando aplicado em diferentes contextos, nomeadamente em problemas com diferentes volumes de dados e com complexidades distintas. A topologia *gbest* provou ser um mecanismo de comunicação adequado, quando se pretende garantir eficiência e rapidez de convergência. O algoritmo adaptou-se com facilidade às restrições que lhe foram impostas: recursos computacionais e tempo disponível limitado (promoveu, mesmo com estas condicionantes, excelentes resultados).

A utilização do PSO, ou eventualmente de outro algoritmo de otimização, deve ser sempre equacionada, quando as *benchmarks* apresentam um baixo número de amostras, uma vez que o tempo despendido na otimização das arquiteturas é reduzido e o seu uso proporciona inúmeras vantagens, entre as quais: automatiza e reduz o esforço manual do cientista de dados, melhora o desempenho das arquiteturas e reduz a sua complexidade. Os benefícios citados foram comprovados através dos excelentes resultados obtidos no *dataset Colorectal Histopathology*. A sua utilização em problemas com um volume de dados superior (*Breast Histopathology*) revelou-se de igual forma eficaz, contudo os recursos computacionais e tempo de execução exigidos aumentam significativamente. Existem abordagens que reduzem o tempo de execução do algoritmo, como: a diminuição do número de *epochs* ou a definição de limites menos abrangentes para as dimensões do problema, porém a eficiência do PSO decresce.

A aplicação desenvolvida demonstrou ser eficiente e rápida na resposta ao diagnóstico de amostras, sendo exibido o diagnóstico num intervalo inferior a 10 segundos. Ao contrário das restantes aplicações, o *DiagnosisViewer* não limita o acesso dos utilizadores a um número reduzido de *benchmarks* e de modelos convolucionais, permitindo o acesso dos utilizadores a todo o conteúdo disponibilizado na plataforma, sendo que, quanto maior o conteúdo submetido, maior a oferta disponível para acesso. A plataforma minimiza as restrições de partilha de modelos CNN entre a comunidade, possibilitando a submissão de modelos com diferentes configurações de "construção e treino", nomeadamente: adaptados a diferentes resoluções das amostras, a diferentes resultados do mapeamento (para valores inteiros) das categorias referentes às classes do problema ou a diferentes abordagens de treino. O *DiagnosisViewer* disponibiliza uma *interface* intuitiva e não requer a implementação de código, quer para a submissão de modelos convolucionais, quer para a realização de diagnósticos (outras plataformas exigem a implementação de código), possibilitando, assim, um fácil acesso às suas vantagens e às suas funcionalidades.

9.2 Trabalho Futuro

A proposta inicialmente definida foi cumprida na íntegra, contudo existe ainda espaço para a melhoria da proposta implementada. Seguidamente são identificados alguns "pontos" que devem ser considerados e trabalhados com vista à melhoria da solução desenvolvida, nomeadamente:

- Deve ser aprimorado o desempenho das arquiteturas na resolução do problema *Skin Mnist*;
- As arquiteturas foram otimizadas com recurso à topologia *gbest*. Futuramente deve ser avaliada a *performance* do algoritmo, atendendo ao uso de diferentes topologias, como por exemplo: a topologia em anel;
- A variação do custo e a rapidez de convergência devem ser analisadas, considerando o uso de um maior número de iterações e de partículas;
- O PSO permite a avaliação paralela das partículas, contudo e devido às limitações existentes (único GPU disponível), foi avaliada uma única partícula em simultâneo. No futuro, importa analisar o impacto da paralelização na redução do tempo de execução do algoritmo;
- O método proposto deve ser validado num maior número de *benchmarks*, com o intuito de confirmar a sua eficácia (e do PSO) num contexto mais abrangente;
- A eficácia do PSO deve ser comprovada, considerando o aumento do número de dimensões a otimizar para cada arquitetura. As dimensões do *kernel* ou o valor do *stride* são exemplos de hiperparâmetros adicionais a otimizar. Esta "experiência" avalia o impacto que o aumento do espaço do problema pode causar, ao nível da procura (identificação de "boas" soluções) e da convergência das partículas;
- Possibilitar a submissão de modelos convolucionais, criados com recurso às principais *frameworks*: *PyTorch, Caffe*, *Microsoft Cognitive Toolkit*. De momento, é apenas permitida a submissão de modelos CNN desenvolvidos através da biblioteca *Keras*;
- Os modelos submetidos pelos utilizadores são armazenados localmente. Esta abordagem torna-se rapidamente impraticável, à medida que o número de submissões aumenta. Sendo assim, os ficheiros devem ser guardados em "locais" que possibilitem a memorização de elevada quantidade de informação, por exemplo instâncias disponibilizadas por serviços de *clouding*;
- Melhoria da *interface* gráfica da aplicação e consequente adição de um maior número de funcionalidades, como por exemplo: permitir o envio de mensagens entre utilizadores, criação de fórum de discussão, *etc*;
- Disponibilização do *DiagnosisViewer* num serviço de *hosting*.

REFERÊNCIAS

Aggarwal, C. C. (2018) *Neural Networks and Deep Learning: A Textbook*. Springer International Publishing. doi: 10.1007/978-3-319-94463-0.

Alghodhaifi, H., Alghodhaifi, A. and Alghodhaifi, M. (2019) 'Predicting Invasive Ductal Carcinoma in breast histology images using Convolutional Neural Network', in *2019 IEEE National Aerospace and Electronics Conference (NAECON). 2019 IEEE National Aerospace and Electronics Conference (NAECON)*, pp. 374–378. doi: 10.1109/NAECON46414.2019.9057822.

Alinsaif, S. and Lang, J. (2020) 'Histological Image Classification using Deep Features and Transfer Learning', in *2020 17th Conference on Computer and Robot Vision (CRV). 2020 17th Conference on Computer and Robot Vision (CRV)*, Ottawa, ON, Canada: IEEE, pp. 101–108. doi: 10.1109/CRV50864.2020.00022.

Arora, R. K. (2015) *Optimization: Algorithms and Applications*. CRC Press.

Asare, S. K., You, F. and Nartey, O. T. (2020) 'Efficient, Ultra-facile Breast Cancer Histopathological Images Classification Approach Utilizing Deep Learning Optimizers', *International Journal of Computer Applications*, 177(37), pp. 1–9.

Atienza, R. (2020) *Advanced Deep Learning with TensorFlow 2 and Keras: Apply DL, GANs, VAEs, deep RL, unsupervised learning, object detection and segmentation, and more, 2nd Edition*. Edição: 2. Packt Publishing.

Barata, C. and Santiago, C. (2020) 'How Important Is Each Dermoscopy Image?', in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Seattle, WA, USA: IEEE, pp. 3202–3210. doi: 10.1109/CVPRW50498.2020.00379.

Barrett, T. *et al.* (2005) 'NCBI GEO: mining millions of expression profiles—database and tools', *Nucleic Acids Research*, 33(suppl_1), pp. D562–D566. doi: 10.1093/nar/gki022.

Beni, G. and Wang, J. (1993) 'Swarm Intelligence in Cellular Robotic Systems', in Dario, P., Sandini, G., and Aebischer, P. (eds) *Robots and Biological Systems: Towards a New Bionics?* Berlin, Heidelberg: Springer (NATO ASI Series), pp. 703–712. doi: 10.1007/978-3-642-58069-7_38.

Bianconi, F., Bello-Cerezo, R. and Napoletano, P. (2017) 'Improved opponent color local binary patterns: an effective local image descriptor for color texture classification', *Journal of Electronic Imaging*, 27(01), p. 1. doi: 10.1117/1.JEI.27.1.011002.

Bjorck, J. *et al.* (2018) 'Understanding batch normalization', in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. Montréal, Canada: Curran Associates Inc. (NIPS'18), pp. 7705–7716.

Branco, P., Torgo, L. and Ribeiro, R. P. (2016) 'A Survey of Predictive Modeling on Imbalanced Domains', *ACM Computing Surveys*, 49(2), pp. 1–50. doi: 10.1145/2907070.

Cai, S. *et al.* (2020) 'Effective and Efficient Dropout for Deep Convolutional Neural Networks', *arXiv:1904.03392 [cs]*. Available at: http://arxiv.org/abs/1904.03392 (Accessed: 20 November 2020).

Carvalho, T. M. de *et al.* (2019) 'Development of Smartphone Apps for Skin Cancer Risk Assessment: Progress and Promise', *JMIR Dermatology*, 2(1), p. e13376. doi: 10.2196/13376.

Chellapilla, K., Puri, S. and Simard, P. (2006) 'High Performance Convolutional Neural Networks for Document Processing', in. *Tenth International Workshop on Frontiers in Handwriting Recognition*, Suvisoft. Available at: https://hal.inria.fr/inria-00112631 (Accessed: 7 August 2020).

Chollet, F. (2015) *GitHub - keras-team/keras: Deep Learning for humans*. Available at: https://github.com/keras-team/keras (Accessed: 19 June 2020).

Clark, K. *et al.* (2013) 'The Cancer Imaging Archive (TCIA): Maintaining and Operating a Public Information Repository', *Journal of Digital Imaging*, 26(6), pp. 1045–1057. doi: 10.1007/s10278-013-9622-7.

Clerc, M. (2010) *Particle Swarm Optimization*. John Wiley & Sons.

Coolen, A. C. C. (1998) 'MatAheBmeagtinicnseor'fsNGeuuirdael Ntoettwheorks', *Concepts for Neural Networks*, p. 61. doi: https://doi.org/10.1007/978-1-4471-3427-5_2.

Crawford, K. L., Neu, S. C. and Toga, A. W. (2016) 'The Image and Data Archive at the Laboratory of Neuro Imaging', *NeuroImage*, 124(0 0), pp. 1080–1083. doi: 10.1016/j.neuroimage.2015.04.067.

Cruz-Roa, A. *et al.* (2014) 'Automatic detection of invasive ductal carcinoma in whole slide images with convolutional neural networks', in Gurcan, M. N. and Madabhushi, A. (eds). *SPIE Medical Imaging*, San Diego, California, USA, p. 904103. doi: 10.1117/12.2043872.

Cui, H. *et al.* (2017) 'Parameter Selection and Performance Comparison of Particle Swarm Optimization in Sensor Networks Localization', *Sensors (Basel, Switzerland)*, 17(3). doi: 10.3390/s17030487.

Djoewahir, A., Kanya, T. and Shenglin, M. (2012) 'A Modified Particle Swarm Optimization with Nonlinear Decreasing Inertia Weight Based PID Controller for Ultrasonic Motor', *International Journal of Innovation*, 3(3), p. 4.

Eberhart, R. and Kennedy, J. (1995) 'A new optimizer using particle swarm theory', in *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science. MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, Nagoya, Japan: IEEE, pp. 39–43. doi: 10.1109/MHS.1995.494215.

Emmert, D. B. *et al.* (1994) 'The European Bioinformatics Institute (EBI) databases', *Nucleic Acids Research*, 22(17), pp. 3445–3449. doi: 10.1093/nar/22.17.3445.

Engelbrecht, A. P. (2007) *Computational Intelligence: An Introduction*. 2nd edn. Wiley Publishing.

Engelbrecht, A. P. (2013) 'Particle Swarm Optimization: Global Best or Local Best?', in *2013 BRICS Congress on Computational Intelligence and 11th Brazilian Congress on*

Computational Intelligence. 2013 BRICS Congress on Computational Intelligence & 11th Brazilian Congress on Computational Intelligence (BRICS-CCI & CBIC), Ipojuca, Brazil: IEEE, pp. 124–135. doi: 10.1109/BRICS-CCI-CBIC.2013.31.

Fernandes Junior, F. E. and Yen, G. (2019) 'Particle swarm optimization of deep neural networks architectures for image classification', *Swarm and Evolutionary Computation*, 49, pp. 62–74. doi: 10.1016/j.swevo.2019.05.010.

Frangioni, J. V. (2008) 'New Technologies for Human Cancer Imaging', *Journal of Clinical Oncology*, 26(24), pp. 4012–4021. doi: 10.1200/JCO.2007.14.3065.

FUKUSHIMA, K. (1979) 'Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position-Neocognitron-', *IEICE Technical Report, A*, 62(10), pp. 658–665.

Fukushima, K. (1980) 'Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position', *Biological Cybernetics*, 36(4), pp. 193–202. doi: 10.1007/BF00344251.

Geirhos, R. *et al.* (2018) 'Comparing deep neural networks against humans: object recognition when the signal gets weaker', *arXiv:1706.06969 [cs, q-bio, stat]*. Available at: http://arxiv.org/abs/1706.06969 (Accessed: 23 May 2020).

Gessert, N. *et al.* (2018) 'Skin Lesion Diagnosis using Ensembles, Unscaled Multi-Crop Evaluation and Loss Weighting', *arXiv:1808.01694 [cs]*. Available at: http://arxiv.org/abs/1808.01694 (Accessed: 21 August 2020).

Glorot, X. and Bengio, Y. (2010) 'Understanding the difficulty of training deep feedforward neural networks', in *In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS'10). Society for Artificial Intelligence and Statistics*, p. 8.

Goodfellow, I., Bengio, Y. and Courville, A. (2016) *Deep Learning*. The MIT Press.

Greenspan, H., van Ginneken, B. and Summers, R. M. (2016) 'Guest Editorial Deep Learning in Medical Imaging: Overview and Future Promise of an Exciting New Technique', *IEEE Transactions on Medical Imaging*, 35(5), pp. 1153–1159. doi: 10.1109/TMI.2016.2553401.

Gu, J. *et al.* (2017) 'Recent Advances in Convolutional Neural Networks', *arXiv:1512.07108 [cs]*. Available at: http://arxiv.org/abs/1512.07108 (Accessed: 12 November 2020).

Haixiang, G. *et al.* (2017) 'Learning from class-imbalanced data: Review of methods and applications', *Expert Systems with Applications*, 73, pp. 220–239. doi: 10.1016/j.eswa.2016.12.035.

Hanin, B. and Rolnick, D. (2018) 'How to Start Training: The Effect of Initialization and Architecture', *32nd Conference on Neural Information Processing Systems*, p. 11.

Hasanpour, S. H. *et al.* (2016) 'Lets keep it simple: using simple architectures to outperform deeper and more complex architectures'.

Hassanien, A. E. and Alamry, E. (2015) *Swarm Intelligence: Principles, Advances, and Applications*. USA: CRC Press, Inc.

He, K. *et al.* (2015a) 'Deep Residual Learning for Image Recognition', *arXiv:1512.03385 [cs]*. Available at: http://arxiv.org/abs/1512.03385 (Accessed: 10 July 2020).

He, K. *et al.* (2015b) 'Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification', in *2015 IEEE International Conference on Computer Vision (ICCV). 2015 IEEE International Conference on Computer Vision (ICCV)*, Santiago, Chile: IEEE, pp. 1026–1034. doi: 10.1109/ICCV.2015.123.

Heaton, J. (2012) *Introduction to the Math of Neural Networks*. Heaton Research, Inc.

Helwig, S. (2010) *Particle Swarms for Constrained Optimization*. Doctoral Thesis. Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU). Available at: https://opus4.kobv.de/opus4-fau/frontdoor/index/index/year/2010/docId/1328 (Accessed: 23 June 2020).

Hilario, A. F. *et al.* (2018) *Learning from Imbalanced Data Sets*. Springer International Publishing. doi: 10.1007/978-3-319-98074-4.

Howard, A. G. *et al.* (2017) 'MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications', *arXiv:1704.04861 [cs]*. Available at: http://arxiv.org/abs/1704.04861 (Accessed: 24 August 2020).

Huang, G. *et al.* (2018) 'Densely Connected Convolutional Networks', *arXiv:1608.06993 [cs]*. Available at: http://arxiv.org/abs/1608.06993 (Accessed: 10 July 2020).

Ioffe, S. and Szegedy, C. (2015) 'Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift', *arXiv:1502.03167 [cs]*. Available at: http://arxiv.org/abs/1502.03167 (Accessed: 10 July 2020).

Janowczyk, A. and Madabhushi, A. (2016) 'Deep learning for digital pathology image analysis: A comprehensive tutorial with selected use cases', *Journal of Pathology Informatics*, 7(1), p. 29. doi: 10.4103/2153-3539.186902.

Johnson, J. M. and Khoshgoftaar, T. M. (2019) 'Survey on deep learning with class imbalance', *Journal of Big Data*, 6(1), p. 27. doi: 10.1186/s40537-019-0192-5.

Ju, C., Bibaut, A. and van der Laan, M. J. (2017) 'The Relative Performance of Ensemble Methods with Deep Convolutional Neural Networks for Image Classification', *arXiv:1704.01664 [cs, stat]*. Available at: http://arxiv.org/abs/1704.01664 (Accessed: 8 November 2020).

Kather, J. N. *et al.* (2016) 'Multi-class texture analysis in colorectal cancer histology', *Scientific Reports*, 6(1), p. 27988. doi: 10.1038/srep27988.

Kennedy, J. (1999) 'Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance', in *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406). Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, pp. 1931-1938 Vol. 3. doi: 10.1109/CEC.1999.785509.

Kennedy, J. and Eberhart, R. C. (2001) *Swarm intelligence*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Kennedy, J. and Mendes, R. (2002) 'Population structure and particle swarm performance', in *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600). Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600)*, pp. 1671–1676 vol.2. doi: 10.1109/CEC.2002.1004493.

Keskar, N. S. *et al.* (2017) 'On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima', *arXiv:1609.04836 [cs, math]*. Available at: http://arxiv.org/abs/1609.04836 (Accessed: 24 November 2020).

Kingma, D. P. and Ba, J. (2017) 'Adam: A Method for Stochastic Optimization', *arXiv:1412.6980 [cs]*. Available at: http://arxiv.org/abs/1412.6980 (Accessed: 10 July 2020).

Kistler, M. (2013) *JMIR - The Virtual Skeleton Database: An Open Access Repository for Biomedical Research and Collaboration | Kistler | Journal of Medical Internet Research*. Available at: https://www.jmir.org/2013/11/e245 (Accessed: 19 June 2020).

Kokkinos, I. (2016) 'Pushing the Boundaries of Boundary Detection using Deep Learning', *arXiv:1511.07386 [cs]*. Available at: http://arxiv.org/abs/1511.07386 (Accessed: 23 May 2020).

Kotsiantis, S., Kanellopoulos, D. and Pintelas, P. (2006) 'Handling imbalanced datasets: A review', p. 12.

Krizhevsky, A., Sutskever, I. and Hinton, G. E. (2012) 'ImageNet classification with deep convolutional neural networks', in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*. Red Hook, NY, USA: Curran Associates Inc. (NIPS'12), pp. 1097–1105.

Lai, K.-H. and Chang, M.-C. (2019) 'Classification of Tissue Types in Histological Colorectal Cancer Images Using Residual Networks', in *2019 IEEE 8th Global Conference on Consumer Electronics (GCCE). 2019 IEEE 8th Global Conference on Consumer Electronics (GCCE)*, pp. 299–300. doi: 10.1109/GCCE46687.2019.9015250.

LeCun, Y. *et al.* (1989) 'Backpropagation applied to handwritten zip code recognition', *Neural Computation*, 1(4), pp. 541–551. doi: 10.1162/neco.1989.1.4.541.

Lecun, Y. *et al.* (1998) 'Gradient-based learning applied to document recognition', *Proceedings of the IEEE*, 86(11), pp. 2278–2324. doi: 10.1109/5.726791.

Leong, M. *et al.* (2020) 'Semi-CNN Architecture for Effective Spatio-Temporal Learning in Action Recognition', *Applied Sciences*, 10, p. 557. doi: 10.3390/app10020557.

Li, M. *et al.* (2014) 'Efficient mini-batch training for stochastic optimization', in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '14*. *the 20th ACM SIGKDD international conference*, New York, New York, USA: ACM Press, pp. 661–670. doi: 10.1145/2623330.2623612.

Lin, M., Chen, Q. and Yan, S. (2014) 'Network In Network', *arXiv:1312.4400 [cs]*. Available at: http://arxiv.org/abs/1312.4400 (Accessed: 10 July 2020).

Lorenzo, P. R. *et al.* (2017) 'Particle swarm optimization for hyper-parameter selection in deep neural networks', in *Proceedings of the Genetic and Evolutionary Computation Conference*.

GECCO '17: Genetic and Evolutionary Computation Conference, Berlin Germany: ACM, pp. 481–488. doi: 10.1145/3071178.3071208.

Lu, L. *et al.* (2020) 'Dying ReLU and Initialization: Theory and Numerical Examples', *arXiv:1903.06733 [cs, math, stat]*. Available at: http://arxiv.org/abs/1903.06733 (Accessed: 30 October 2020).

Maguolo, G., Nanni, L. and Ghidoni, S. (2019) 'Ensemble of Convolutional Neural Networks Trained with Different Activation Functions', p. 7.

Manning, C. D., Raghavan, P. and Schütze, H. (2008) *Introduction to Information Retrieval*. USA: Cambridge University Press.

Marcus, G. (2018) 'Deep Learning: A Critical Appraisal', p. 27.

Markram, H. *et al.* (2011) 'Introducing the Human Brain Project', *Procedia Computer Science*, 7, pp. 39–42. doi: 10.1016/j.procs.2011.12.015.

Marta, D. (2016) *Deep Learning Methods for ReinforcementLearning*. Instituto Superior Técnico.

Mendes, R. *et al.* (2002) 'Particle swarms for feedforward neural network training', in *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No.02CH37290). Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No.02CH37290)*, pp. 1895–1899 vol.2. doi: 10.1109/IJCNN.2002.1007808.

Milton, A. A. (2019) 'Automated Skin Lesion Classification Using Ensemble of Deep Neural Networks in ISIC 2018: Skin Lesion Analysis Towards Melanoma Detection Challenge', p. 4.

Miranda, L. J. (2018) 'PySwarms: a research toolkit for Particle Swarm Optimization in Python', *Journal of Open Source Software*, 3(21), p. 433. doi: 10.21105/joss.00433.

Mitchell, T. M. (1997) *Machine Learning*. 1st edn. USA: McGraw-Hill, Inc.

ml5-library (2020) *ml5js/ml5-library*. ml5. Available at: https://github.com/ml5js/ml5-library (Accessed: 15 October 2020).

Mohapatra, P., Panda, B. and Swain, S. (2019) 'Enhancing Histopathological Breast Cancer Image Classification using Deep Learning', *International Journal of Innovative Technology and Exploring Engineering*, 8(7), p. 9.

More, A. (2016) 'Survey of resampling techniques for improving classification performance in unbalanced datasets', *arXiv:1608.06048 [cs, stat]*. Available at: http://arxiv.org/abs/1608.06048 (Accessed: 15 July 2020).

More, J. *et al.* (2020) 'Skin Disease Classification Using Convolutional Neural Network', in. *International Research Journal ofEngineering and Technology*. Available at: https://www.irjet.net/archives/V7/i5/IRJET-V7I51478.pdf (Accessed: 23 August 2020).

Nasr, G. E., Badr, E. A. and Joun, C. (2002) 'Cross Entropy Error Function in Neural Networks: Forecasting Gasoline Demand', in *Proceedings of the Fifteenth International Florida Artificial Intelligence Research Society Conference*. AAAI Press, pp. 381–384.

Nielsen, M. A. (2015) *Neural Networks and Deep Learning*. Determination Press.

Norris, D. J. (2019) *Machine Learning with the Raspberry Pi: Experiments with Data and Computer Vision*. 1st ed. edition. Place of publication not identified: Apress.

Nwankpa, C. *et al.* (2018) 'Activation Functions: Comparison of trends in Practice and Research for Deep Learning', *arXiv:1811.03378 [cs]*. Available at: http://arxiv.org/abs/1811.03378 (Accessed: 10 July 2020).

Oldewage, E. T., Engelbrecht, A. P. and Cleghorn, C. W. (2018) 'Boundary Constraint Handling Techniques for Particle Swarm Optimization in High Dimensional Problem Spaces', in Dorigo, M. et al. (eds) *Swarm Intelligence*. Cham: Springer International Publishing (Lecture Notes in Computer Science), pp. 333–341. doi: 10.1007/978-3-030-00533-7_27.

Ortega-Navas, M. del C. (2017) 'The use of New Technologies as a Tool for the Promotion of Health Education', *Procedia - Social and Behavioral Sciences*, 237, pp. 23–29. doi: 10.1016/j.sbspro.2017.02.006.

Ouyang, W. *et al.* (2019) 'ImJoy: an open-source computational platform for the deep learning era', *Nature Methods*, 16(12), pp. 1199–1200. doi: 10.1038/s41592-019-0627-0.

Pal, A., Ray, S. and Garain, U. (2018) 'Skin disease identification from dermoscopy images using deep convolutional neural network', *arXiv:1807.09163 [cs]*. Available at: http://arxiv.org/abs/1807.09163 (Accessed: 22 August 2020).

Panigrahi, B. K., Shi, Y. and Lim, M.-H. (2011) *Handbook of Swarm Intelligence: Concepts, Principles and Applications*. 1st edn. Springer Publishing Company, Incorporated.

Park, S. and Kwak, N. (2017) 'Analysis on the Dropout Effect in Convolutional Neural Networks', in Lai, S.-H. et al. (eds) *Computer Vision – ACCV 2016*. Cham: Springer International Publishing (Lecture Notes in Computer Science), pp. 189–204. doi: 10.1007/978-3-319-54184-6_12.

Parsopoulos, K. E. and Vrahatis, M. N. (2010) *Particle Swarm Optimization and Intelligence: Advances and Applications*. IGI Global (Advances in Computational Intelligence and Robotics). doi: 10.4018/978-1-61520-666-7.

Patterson, J. and Gibson, A. (2017) *Deep Learning: A Practitioner's Approach*. 1 edition. O'Reilly Media.

Pereira, S. *et al.* (2016) 'Brain Tumor Segmentation Using Convolutional Neural Networks in MRI Images', *IEEE Transactions on Medical Imaging*, 35(5), pp. 1240–1251. doi: 10.1109/TMI.2016.2538465.

Perez, L. and Wang, J. (2017) 'The Effectiveness of Data Augmentation in Image Classification using Deep Learning', *arXiv:1712.04621 [cs]*. Available at: http://arxiv.org/abs/1712.04621 (Accessed: 10 July 2020).

Prathabrao, M., Nawawi, A. and Sidek, N. A. (2017) 'Swarm size and iteration number effects to the performance of PSO algorithm in RFID tag coverage optimization', *AIP Conference Proceedings*, 1831(1), p. 020051. doi: 10.1063/1.4981192.

Rączkowski, Ł. *et al.* (2019) 'ARA: accurate, reliable and active histopathological image classification framework with Bayesian deep learning', *Scientific Reports*, 9(1), p. 14347. doi: 10.1038/s41598-019-50587-1.

Rawat, W. and Wang, Z. (2017) 'Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review', *Neural Computation*, 29(9), pp. 2352–2449. doi: 10.1162/neco_a_00990.

Reynolds, C. W. (1987) 'Flocks, herds and schools: A distributed behavioral model', *ACM SIGGRAPH Computer Graphics*, 21(4), pp. 25–34. doi: 10.1145/37402.37406.

Riedmiller, M. and Braun, H. (1993) 'A direct adaptive method for faster backpropagation learning: the RPROP algorithm', in *IEEE International Conference on Neural Networks*. *IEEE International Conference on Neural Networks*, San Francisco, CA, USA: IEEE, pp. 586–591. doi: 10.1109/ICNN.1993.298623.

Rodrigo, M. (2018) *Evaluación de redes neuronales convolucionales para la clasificación de imágenes histológicas de cáncer colorrectal mediante transferencia de aprendizaje*. Universitat Oberta de Catalunya. Available at: http://openaccess.uoc.edu/webapps/o2/bitstream/10609/74105/6/jmarturetTFM0118memoria.pdf (Accessed: 20 August 2020).

Romero, F. P., Tang, A. and Kadoury, S. (2019) 'Multi-Level Batch Normalization In Deep Networks For Invasive Ductal Carcinoma Cell Discrimination In Histopathology Images', *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*, pp. 1092–1095. doi: 10.1109/ISBI.2019.8759410.

Rosenblatt, F. (1958) 'The perceptron: A probabilistic model for information storage and organization in the brain.', *Psychological Review*, 65(6), pp. 386–408. doi: 10.1037/h0042519.

Ruder, S. (2017) 'An overview of gradient descent optimization algorithms', *arXiv:1609.04747 [cs]*. Available at: http://arxiv.org/abs/1609.04747 (Accessed: 10 July 2020).

Sangaiah, A. K. (2019) *Deep Learning and Parallel Computing Environment for Bioengineering Systems*. Academic Press.

Saxe, A. M., McClelland, J. L. and Ganguli, S. (2014) 'Exact solutions to the nonlinear dynamics of learning in deep linear neural networks', *arXiv:1312.6120 [cond-mat, q-bio, stat]*. Available at: http://arxiv.org/abs/1312.6120 (Accessed: 10 July 2020).

Sengupta, S., Basak, S. and Peters, R. A. (2018) *Particle Swarm Optimization: A survey of Historical and Recent Developments with Hybridization Perspectives*. preprint. MATHEMATICS & COMPUTER SCIENCE. doi: 10.20944/preprints201809.0007.v1.

Serizawa, T. and Fujita, H. (2020) 'Optimization of Convolutional Neural Network Using the Linearly Decreasing Weight Particle Swarm Optimization', *arXiv:2001.05670 [cs]*. Available at: http://arxiv.org/abs/2001.05670 (Accessed: 3 August 2020).

Shahata, F., Kaur, K. and Fiaidhi, J. (2020) 'COMPUTER VISION FOR SKIN CANCER DETECTION AND DIAGNOSIS'. doi: 10.36227/techrxiv.12089010.v1.

Shi, Y. and Eberhart, R. (1998) 'A modified particle swarm optimizer', in *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress*

on Computational Intelligence (Cat. No.98TH8360). 1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence, Anchorage, AK, USA: IEEE, pp. 69–73. doi: 10.1109/ICEC.1998.699146.

Simonyan, K. and Zisserman, A. (2015) 'Very Deep Convolutional Networks for Large-Scale Image Recognition', *arXiv:1409.1556 [cs]*. Available at: http://arxiv.org/abs/1409.1556 (Accessed: 10 July 2020).

Sokolova, M. and Lapalme, G. (2009) 'A systematic analysis of performance measures for classification tasks', *Information Processing & Management*, 45(4), pp. 427–437. doi: 10.1016/j.ipm.2009.03.002.

Springenberg, J. T. *et al.* (2015) 'Striving for Simplicity: The All Convolutional Net', *arXiv:1412.6806 [cs]*. Available at: http://arxiv.org/abs/1412.6806 (Accessed: 10 July 2020).

Srivastava, N. *et al.* (2014) 'Dropout: a simple way to prevent neural networks from overfitting', *The Journal of Machine Learning Research*, 15(1), pp. 1929–1958.

StackML (2020) *stackml/stackml-js*. Available at: https://github.com/stackml/stackml-js (Accessed: 15 October 2020).

Sun, Y. *et al.* (2019) 'A Particle Swarm Optimization-based Flexible Convolutional Auto-Encoder for Image Classification', *IEEE Transactions on Neural Networks and Learning Systems*, 30(8), pp. 2295–2309. doi: 10.1109/TNNLS.2018.2881143.

Sutskever, I. *et al.* (2013) 'On the importance of initialization and momentum in deep learning', in *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*. Atlanta, GA, USA: JMLR.org (ICML'13), p. III–1139–III–1147.

Szegedy, C. *et al.* (2014) 'Going Deeper with Convolutions', *arXiv:1409.4842 [cs]*. Available at: http://arxiv.org/abs/1409.4842 (Accessed: 24 August 2020).

Tanweer, M. R., Suresh, S. and Sundararajan, N. (2016) 'Dynamic mentoring and self-regulation based particle swarm optimization algorithm for solving complex real-world optimization problems', *Information Sciences*, 326, pp. 1–24. doi: 10.1016/j.ins.2015.07.035.

Tieleman, T. and Hinton, G. (2012) 'Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude', *COURSERA: Neural networks for machine learning*, 4(2), pp. 26–31.

Tripathy, A. K. *et al.* (eds) (2020) *Advances in Distributed Computing and Machine Learning: Proceedings of ICADCML 2020*. Springer Singapore (Lecture Notes in Networks and Systems). doi: 10.1007/978-981-15-4218-3.

Tsay, J. *et al.* (2018) 'Runway: machine learning model experiment management tool', in.

Tschandl, P., Rosendahl, C. and Kittler, H. (2018) 'The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions', *Scientific Data*, 5(1), p. 180161. doi: 10.1038/sdata.2018.161.

Van Den Bergh, F. (2002) *An analysis of particle swarm optimizers*. phd. University of Pretoria.

vandenBergh, F. and Engelbrecht, A. P. (2004) 'A Cooperative Approach to Particle Swarm Optimization', *IEEE Transactions on Evolutionary Computation*, 8(3), pp. 225–239. doi: 10.1109/TEVC.2004.826069.

Vasilev, I. *et al.* (2019) *Python Deep Learning: Exploring Deep Learning Techniques and Neural Network Architectures with Pytorch, Keras, and TensorFlow, 2nd Edition.* Birmingham: Packt Publishing Ltd. Available at: https://public.ebookcentral.proquest.com/choice/publicfullrecord.aspx?p=5639029 (Accessed: 9 July 2020).

Venter, G. (2010) 'Review of optimization techniques'. Available at: https://scholar.sun.ac.za:443/handle/10019.1/14646 (Accessed: 25 October 2020).

Vieira, S. M. *et al.* (2013) 'Modified binary PSO for feature selection using SVM applied to mortality prediction of septic patients', *Applied Soft Computing*, 13(8), pp. 3494–3504. doi: 10.1016/j.asoc.2013.03.021.

Wang, B. *et al.* (2019) 'A Hybrid GA-PSO Method for Evolving Architecture and Short Connections of Deep Convolutional Neural Networks', *arXiv:1903.03893 [cs]*. Available at: http://arxiv.org/abs/1903.03893 (Accessed: 23 May 2020).

Wang, D., Tan, D. and Liu, L. (2017) 'Particle swarm optimization algorithm: an overview', *Soft Computing*. doi: 10.1007/s00500-016-2474-6.

Watts, D. J. (1999) 'Networks, Dynamics, and the Small-World Phenomenon', *American Journal of Sociology*, 105(2), pp. 493–527. doi: 10.1086/210318.

Watts, D. J. and Strogatz, S. H. (1998) 'Collective dynamics of "small-world" networks', 393, p. 3.

Weng, J., Ahuja, N. and Huang, T. S. (1992) 'Cresceptron: a self-organizing neural network which grows adaptively', in *[Proceedings 1992] IJCNN International Joint Conference on Neural Networks. [Proceedings 1992] IJCNN International Joint Conference on Neural Networks*, pp. 576–581 vol.1. doi: 10.1109/IJCNN.1992.287150.

Williams, E. *et al.* (2017) 'Image Data Resource: a bioimage data integration and publication platform', *Nature Methods*, 14(8), pp. 775–781. doi: 10.1038/nmeth.4326.

Wu, Y. and He, K. (2018) 'Group Normalization', *International Journal of Computer Vision*, 128. doi: 10.1007/s11263-019-01198-w.

Xu, B. *et al.* (2015) 'Empirical Evaluation of Rectified Activations in Convolutional Network', *arXiv:1505.00853 [cs, stat]*. Available at: http://arxiv.org/abs/1505.00853 (Accessed: 10 July 2020).

Xu, S. and Rahmat-Samii, Y. (2007) 'Boundary Conditions in Particle Swarm Optimization Revisited', *IEEE Transactions on Antennas and Propagation*, 55(3), pp. 760–765. doi: 10.1109/TAP.2007.891562.

Yamasaki, T., Honma, T. and Aizawa, K. (2017) 'Efficient Optimization of Convolutional Neural Networks Using Particle Swarm Optimization', in, pp. 70–73. doi: 10.1109/BigMM.2017.69.

Yang, X. *et al.* (2007) 'A modified particle swarm optimizer with dynamic adaptation', *Applied Mathematics and Computation*, 189(2), pp. 1205–1213. doi: 10.1016/j.amc.2006.12.045.

Ye, F. (2017) 'Particle swarm optimization-based automatic parameter selection for deep neural networks and its applications in large-scale and high-dimensional data', *PLOS ONE*. Edited by W.-B. Du, 12(12), p. e0188746. doi: 10.1371/journal.pone.0188746.

Zeiler, M. D. and Fergus, R. (2014) 'Visualizing and Understanding Convolutional Networks', in Fleet, D. et al. (eds) *Computer Vision – ECCV 2014*. Cham: Springer International Publishing (Lecture Notes in Computer Science), pp. 818–833. doi: 10.1007/978-3-319-10590-1_53.

Zhuang, D., Chen, K. and Chang, J. M. (2020) 'CS-AF: A Cost-sensitive Multi-classifier Active Fusion Framework for Skin Lesion Classification', *arXiv:2004.12064 [cs, eess]*. Available at: http://arxiv.org/abs/2004.12064 (Accessed: 22 August 2020).

APÊNDICE A

DATASET – BREAST HISTOPATHOLOGY

O *Invasive Ductal Carcinoma* é um dos tipos de cancro de mama mais frequentes. Uma abordagem usual de diagnóstico da patologia consiste na análise de tecidos mamários. Os tecidos são observados pelos médicos recorrendo a imagens que resultam de um processo de digitalização.

O IDC desenvolve-se inicialmente nos *milk duct's*. Estes caminhos são responsáveis por transportar o leite, que é produzido nos lóbulos, até ao *nipple*. À medida que o cancro se desenvolve tende a invadir outras zonas da mama, geralmente, os seus tecidos e lóbulos. A Figura 36 ilustra o interior da mama.

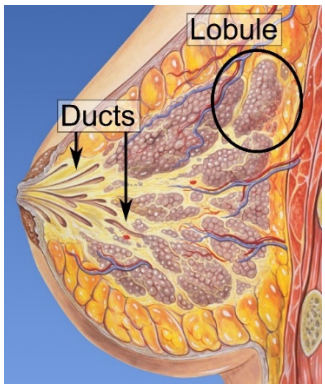

Figura 36 - *Milk ducts* e Lóbulos – Fonte: https://commons.wikimedia.org/wiki/File:Lobules_and_ducts_of_the_breast.jpg

A Figura 37 ilustra a progressão do tumor no interior dos *milk ducts*. As células cancerígenas vão gradualmente circundando toda a área do *duct* (*Myopithelium*), sendo que em última fase, as células acabam por invadir os tecidos mamários.

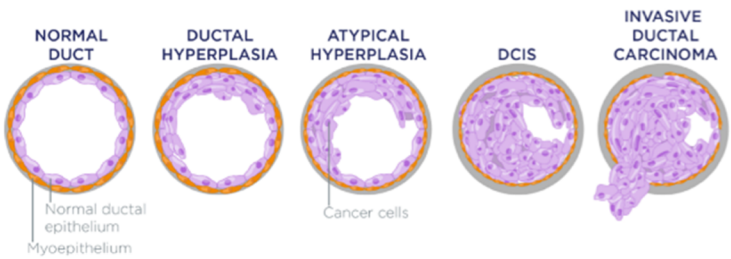

Figura 37 - Progressão do cancro mamário – Fonte: https://medium.com/minimalist-pharmacist/breast-cancer-disease-state-review-patient-case-presentation-4ac505174c4e

O *Breast Histopathology* representa num problema de classificação binário composto por 277524 amostras. As amostras correspondem a imagens digitalizadas de tecidos mamários, referentes a diversos pacientes. O objetivo do problema compreende a correta classificação de tecidos com IDC e sem IDC.

A Figura 38 ilustra a distribuição das amostras entre as duas classes. O número de tecidos sem presença de *"IDC"* (classe *"No IDC"*) é efetivamente superior ao número de tecidos com presença de *"IDC"* (classe *"With IDC"*), apoiando-se numa razão de 2.5-1 amostras, respetivamente. A classe *"No IDC"* reúne 198738 amostras, enquanto que a classe *"With IDC"* contém apenas 78786 amostras.

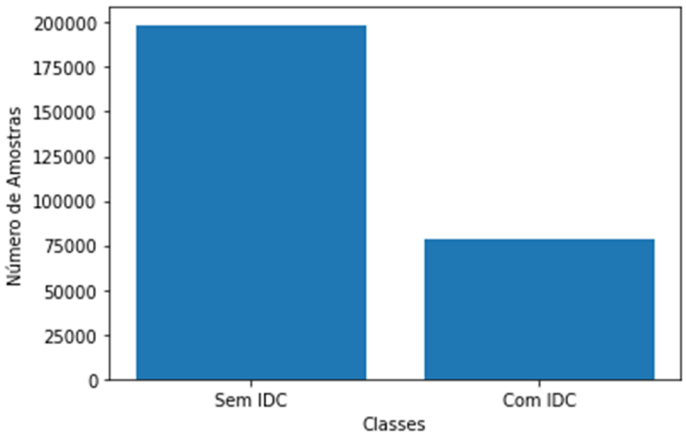

Figura 38 - Distribuição das amostras pelas duas classes

Os tecidos são disponibilizados em formato RGB e apresentam dimensões de 50*50 pixéis, respetivamente largura e comprimento.

As Figuras 39 e 40 ilustram respetivamente dois conjuntos de amostras referentes às classes *"No IDC"* e *"With IDC"*.

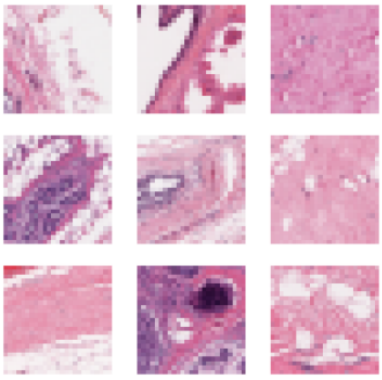

Figura 39 - Amostras classe: "*No IDC*"

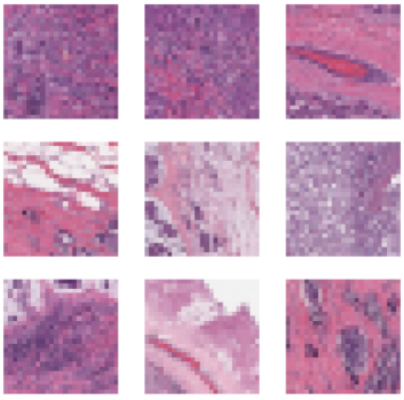

Figura 40 - Amostras classe: "*With IDC*"

À primeira vista o problema é complexo, não sendo trivial a identificação, "a olho nu", de diferenças entre as classes. O padrão de cores é provavelmente a característica mais diferenciadora entre as duas classes. As amostras com presença de *"IDC"* exibem cores mais vivas, possivelmente, devido à presença de células cancerígenas nos tecidos.

APÊNDICE B

DATASET – SKIN MNIST

O *Skin MNIST* é um problema de classificação composto por imagens dermatológicas alusivas a diferentes tipos de cancro de pele.

O *dataset* é composto por 10015 imagens que resultaram de um estudo intensivo desenvolvido por várias entidades e estendendo-se ao longo de 20 anos.

As imagens foram obtidas recorrendo ao processo de dermatoscopia. Os dermatologistas usam o dermatoscópio para visualizarem com maior rigor e segurança as lesões dos seus pacientes. A utilização deste aparelho permite a análise das lesões em escalas amplificadas.

Este *dataset* é composto por 7 diferentes tipos de cancro de pele (classes). Cada classe reúne um conjunto de amostras que podem ser subjacentes a diferentes subtipos de lesões de pele.

Os diferentes tipos de cancro de pele presentes no *dataset* são: *Actinic Keratoses (*akiec), *Basal cell carcinoma* (bcc), *Benign Keratosis* (bkl), *Dermatofibroma* (df), *Melanocytic nevi* (nv), *Melanoma* (mel) e *Vascular skin lesions* (vasc). Seguidamente são descritas as diferentes classes do problema.

A classe *Actinic Keratoses* (ou *Solar Keratoses*) descreve lesões de pele causadas pela excessiva exposição solar. A probabilidade de evolução para cancro de pele invasivo é baixa.

Basal cell carcinoma é o tipo de cancro de pele mais comum. A exposição solar excessiva e a exposição a radioterapia são as principais causas conhecidas para o seu aparecimento. Normalmente e quando detetada precocemente, não resulta em morte. Ainda assim, é necessário ter em consideração que o tumor cresce e pode tornar-se invasivo.

A lesão *Benign Keratosis* subdivide-se em três diferentes grupos: *seborrheic keratoses*, *solar lentigo* e *lichen-planus like keratoses*. Estes diferentes sub-tipos de lesão benigna apresentam semelhanças biológicas entre si e o aspeto da lesão varia consoante o local do corpo onde está localizada.

Dermatofibroma é uma lesão de pele benigna, sendo ainda desconhecidas as causas associadas ao seu aparecimento. Na maioria dos casos, não existe a necessidade de recorrer a qualquer tipo de cirurgia, quer para o seu tratamento, quer para a sua eliminação.

Melanocytic nevi (ou *nevus-cell*) é uma lesão de pele benigna que geralmente resulta da exposição solar excessiva ou está associada a fatores genéticos. Estas lesões aparecem, normalmente, nos primeiros 20 anos de vida dos pacientes. Na maioria dos casos, a probabilidade de evoluir para *Melanoma* é reduzida.

Melanoma representa uma lesão de pele maligna, sendo vários os tipos de *melanoma* conhecidos: superficial, nodular, lentigo maligna ou acral lentiginoso. A exposição solar

excessiva é a principal causa conhecida e associada ao seu aparecimento. Os índices de cura são elevados quando detetada precocemente.

Vascular skin lesion é uma lesão que habitualmente resulta de malformações adquiridas e que origina problemas nos vasos linfáticos, podendo ser lesões benignas ou malignas. As variantes da lesão incluídas no *dataset* são: *cherry angiomas*, *angiokeratomas* e *pyogenic granulomas*.

As imagens são exibidas em formato RGB e apresentam dimensões de 600*450 pixéis, respetivamente comprimento e largura.

A Figura 41 esquematiza a distribuição do número de amostras das sete classes do problema. O *dataset* não é balanceado, sendo que, aproximadamente 70% das amostras correspondem à classe *Melanocytic nevi (nv)*. As restantes classes também apresentam distribuições manifestamente distintas entre si, por exemplo, a classe *Dermatofibroma (df)* reúne apenas 10% do total de amostras da classe *Benign Keratosis (bkl)*.

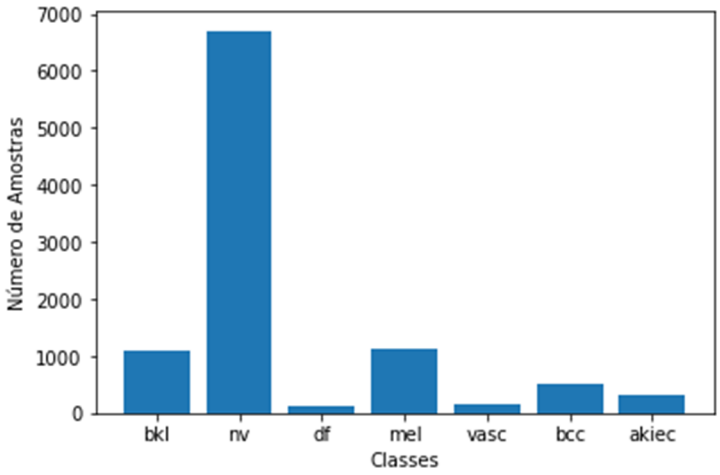

Figura 41 - Distribuição das amostras por classe

Os autores do *dataset* disponibilizaram um conjunto de informações adicionais, que possibilitam a análise mais exaustiva dos dados do problema, nomeadamente: a localização da lesão, a idade do paciente e o sexo do paciente.

A Figura 42 ilustra a quantidade de amostras que está associada a cada local do corpo.

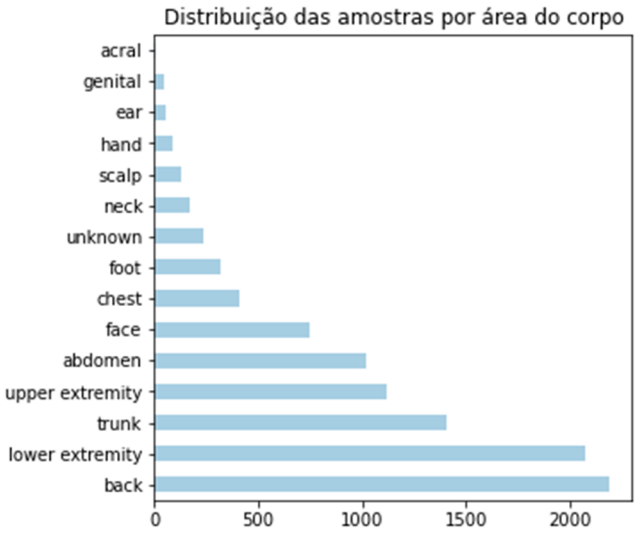

Figura 42 - Distribuição das amostras por área do corpo

A maioria das amostras está distribuída ao longo de cinco principais áreas do corpo: costas (*back*), pernas (*lower extremity*), torso (*trunk*), braços *(upper extremity*) e abdómen. Os restantes locais do corpo representam zonas menos prováveis de aparecimento de lesões de pele. Existem cerca de 350 amostras às quais não lhe fora atribuída uma área do corpo específica (*unknown*).

A Figura 43 esquematiza a distribuição da faixa etária dos pacientes por tipo de lesão. Todas as lesões revelam uma maior incidência em idades posteriores a 40 anos, ainda assim, as lesões *Melanocytic nevi* e *Vascular skin lesion* apresentam também tendência de surgimento em idades inferiores. Sendo assim, a longevidade de vida dos pacientes é um fator que pode estar associado, ao aumento da probabilidade de aparecimento de lesões de pele.

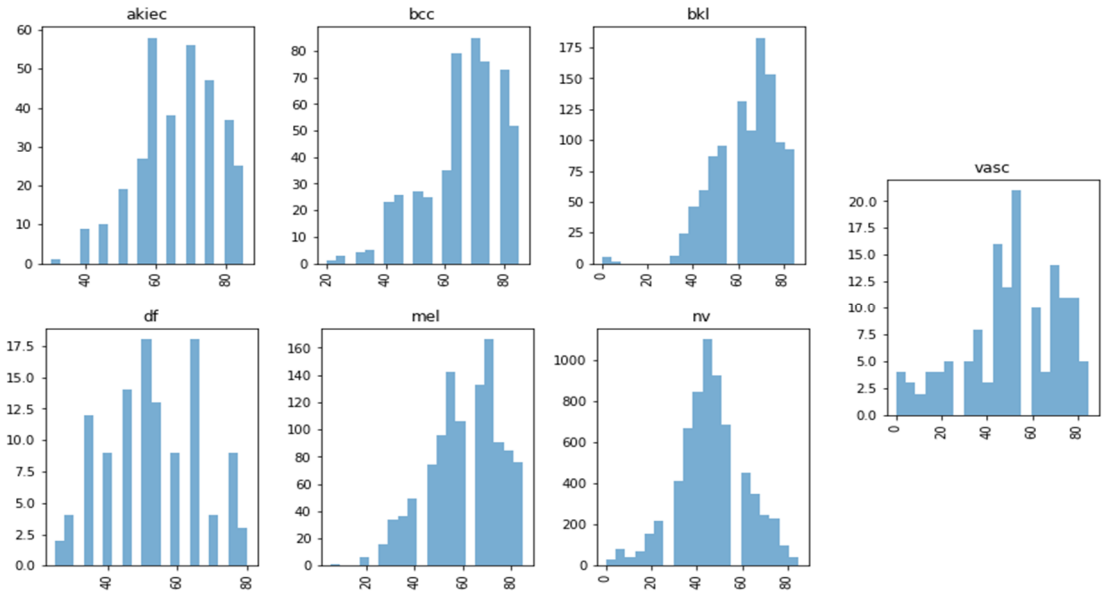

Figura 43 - Distribuição da idade dos pacientes por tipo de lesão

A Figura 44 esquematiza a distribuição do número de amostras por sexo do paciente e por tipo de lesão. À exceção da lesão *Vascular skin*, todas as restantes classes evidenciam uma maior

incidência em pacientes do sexo masculino. Um número reduzido de amostras não reúne identificador do sexo do paciente (*unknown*).

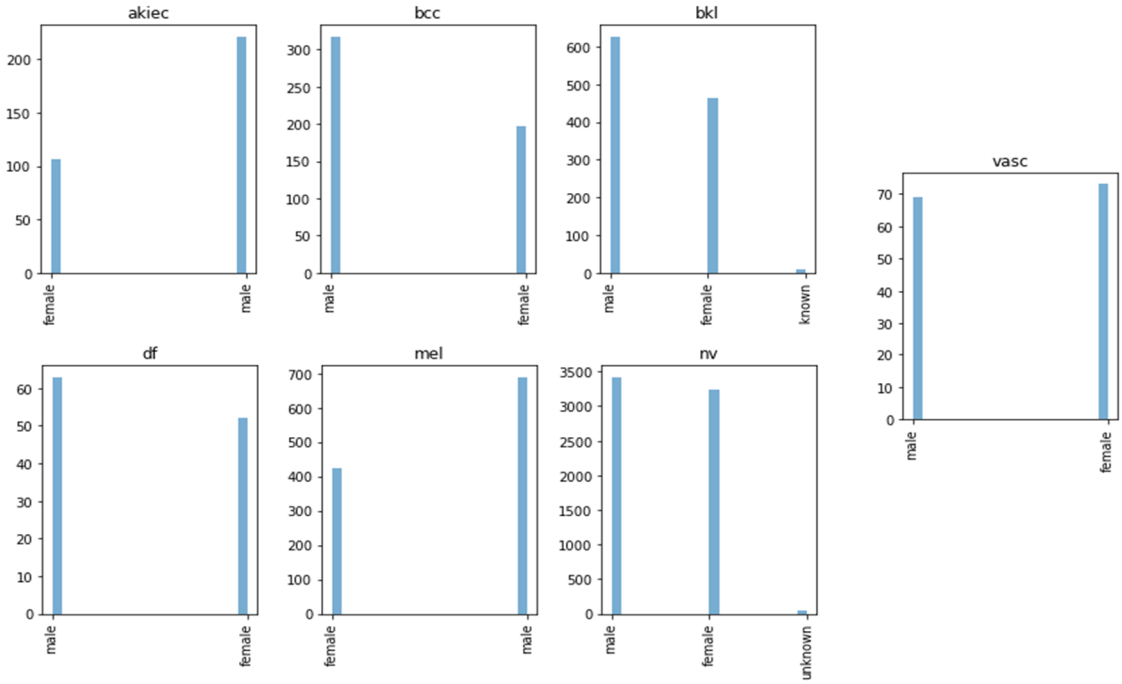

Figura 44 - Distribuição do número de amostras por sexo do paciente e por classe

A Figura 45 ilustra três amostras aleatórias de cada classe.

Aparentemente o problema é complexo, não sendo clara a identificação de particularidades nas amostras (cores ou formas) que permitam a sua classificação intuitiva.

Ainda assim, é possível constatar algumas diferenças entre as lesões, por exemplo, as amostras associadas à classe *Vascular skin lesion* apresentam forma e cor distintas das restantes lesões, isto é, revelam tons fora do "padrão" castanho e o tamanho da lesão é inferior, sendo à partida a classe mais fácil de prever. Os modelos também não deverão reunir dificuldades na classificação da lesão *Melanocytic nevi*, uma vez que esta classe reúne um elevado conjunto de amostras, o que facilita a aprendizagem das suas características.

Já as restantes classes apresentam muitas semelhanças entre si, sendo difícil identificar particularidades, visíveis a "olho nu", que propiciem a correta distinção do tipo de lesão. Dessa forma, e dado o elevado número de classes existentes, o modelo poderá revelar sérias dificuldades na correta classificação das amostras.

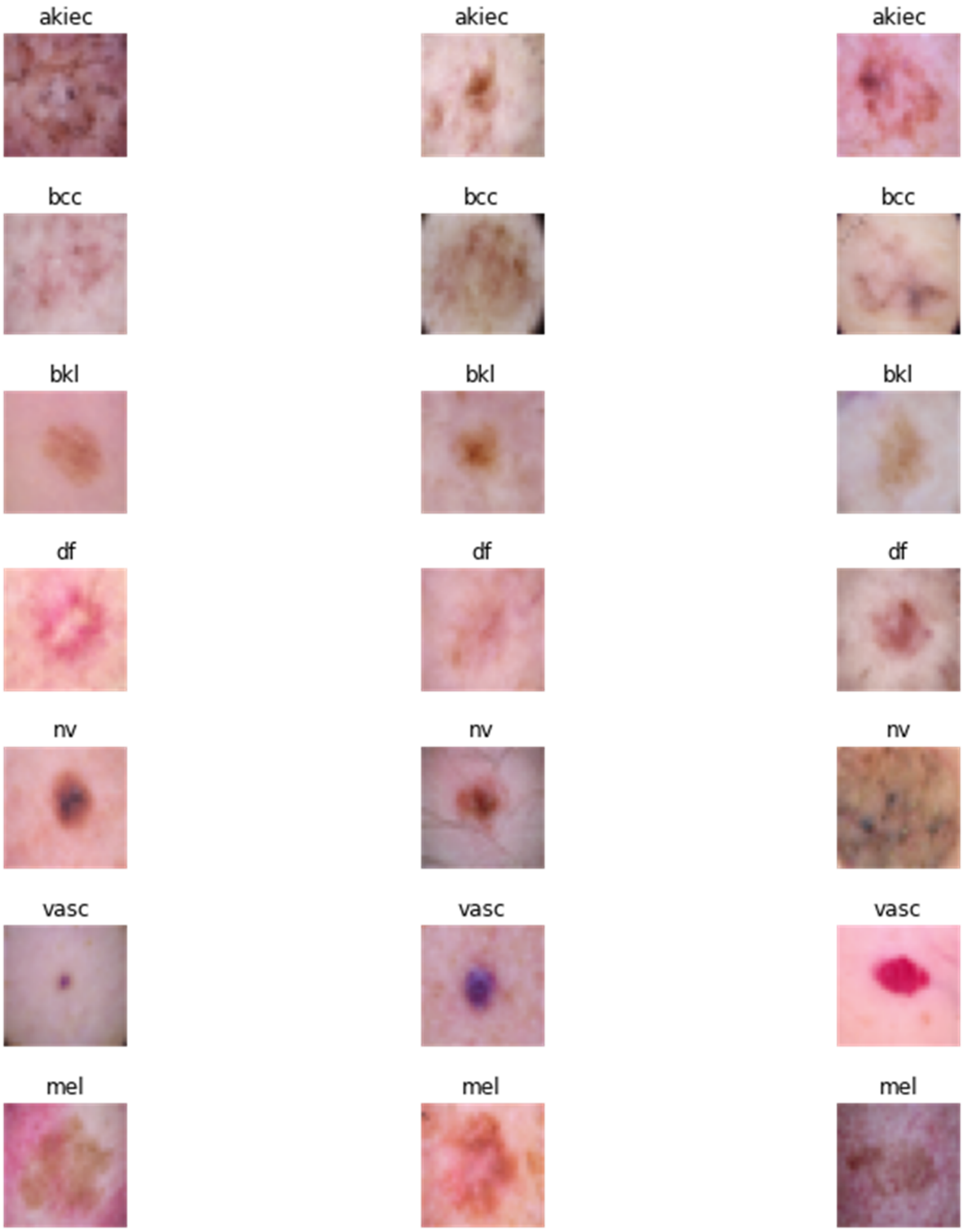

Figura 45 - Exemplificação de três amostras por classe

APÊNDICE C

DATASET – COLORECTAL HISTOPATHOLOGY

O Cancro Colorretal é um dos tipos de cancro que afeta um maior número de pessoas em todo o mundo. O seu aparecimento resulta da transformação progressiva das *epithelial* cells, presentes no intestino, em células cancerígenas.

O *Colorectal Histopathology* é um problema de classificação multi-classe composto por oito tipos de tecidos colorretais (humanos). Os tecidos foram proporcionados pela *University Medical Center Mannheim* e resultaram de um processo histopatológico, ou seja, os patologistas recorreram à observação microscópica das características biológicas dos tecidos, com o objetivo de identificar possível lesões e o seu grau de severidade.

O *dataset* reúne ao todo 5000 amostras, sendo distribuídas 625 amostras por classe. A Figura 46 ilustra a distribuição das amostras pelas oito classes do problema.

As imagens são disponibilizadas em formato RGB (três canais de profundidade) e o seu tamanho é de 150*150 pixéis (largura e comprimento).

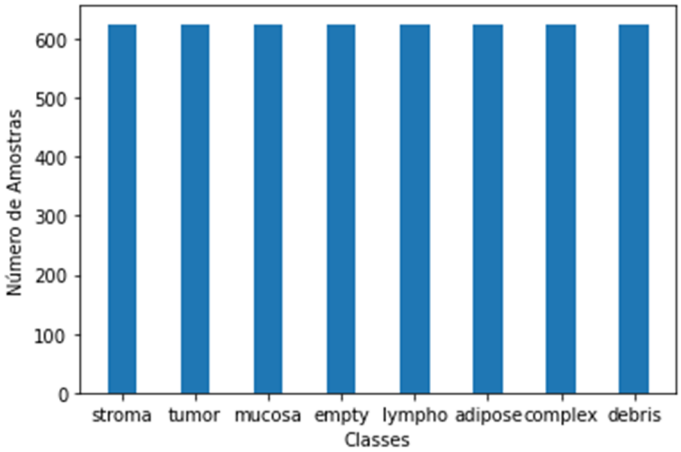

Figura 46 - Distribuição das amostras pelas oito classes

As oito classes em estudo são: *Tumour epithelium* (tumor), *Simple stroma* (stroma), *Complex stroma* (*complex*), *Immune cells* (*lymphoid follicles - lympho*), *Debris* (debris), *Normal mucosal glands* (mucosa), *Adipose tissue* (adipose) e *no tissue* (*empty*).

Tumour epithelium representa tecidos cancerígenos.

Simple stroma referencia composições homogéneas que incluem *tumour stroma, extra-tumoural stroma* e *smooth muscle*.

Complex Stroma contém células com tumor simples.

A classe *Immune Cells* inclui *immune-cell conglomerates* e *sub-mucosal lymphoid follicles*.

Debris descreve tecidos necróticos (mortos).

Normal mucosal glands auxiliam a mucosa através da segregação de *mucus* para o duodeno.

Adipose são tecidos responsáveis por armazenar energia.

A classe *No tissue*, simplesmente, não corresponde a nenhum tecido.

A Figura 47 exemplifica três amostras de cada classe.

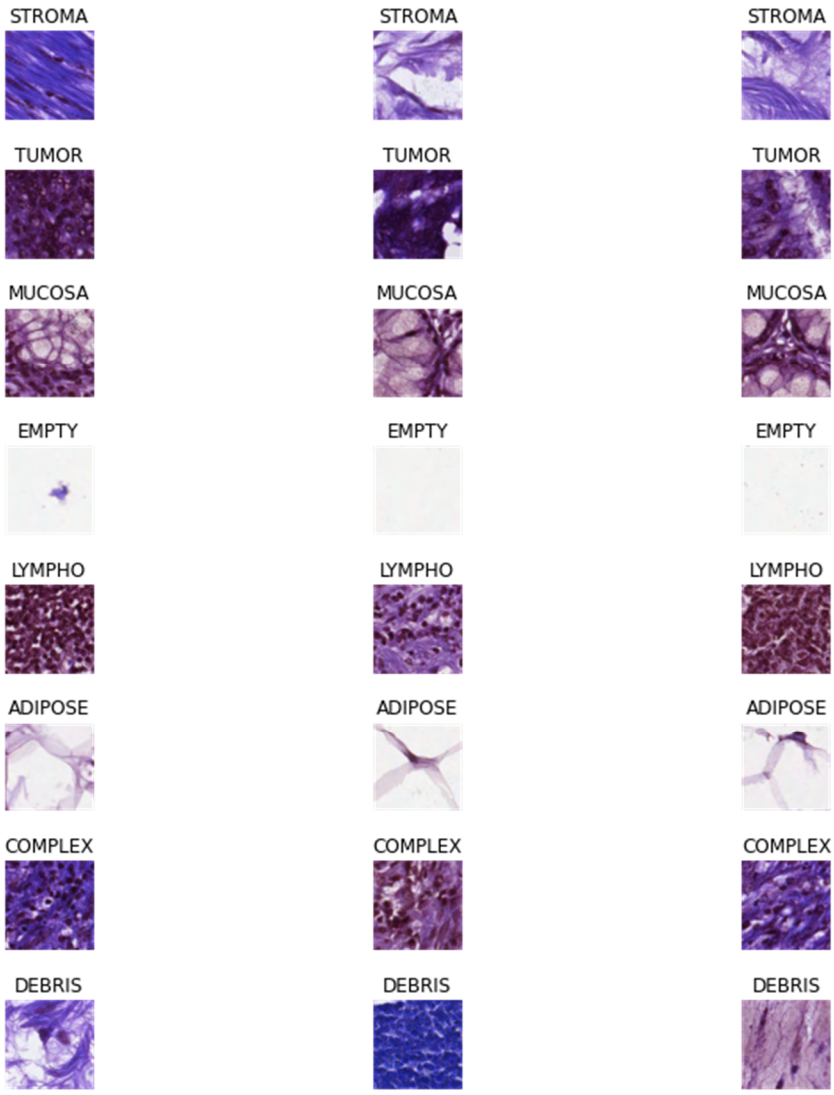

Figura 47 - Exemplificação de três amostras por classe

Observando a Figura 47, é possível constatar que existem duas classes muito distintas das restantes, *"adipose"* e *"no tissue"*. Nestas duas classes predomina a cor branca e o seu preenchimento não é similar às restantes classes.

Aparentemente as classes *Tumour epithelium* e *Normal mucosal glands* também reúnem características que permitem a sua distinção visual, nomeadamente, a cor púrpura muito "acentuada" da classe *tumor* e as formas circulares com "tons claros" presentes nos tecidos das *mucosal glands*.

As restantes classes apresentam características semelhantes entre si, sendo difícil a identificação a "olho nu" de padrões diferenciadores.

A Figura 48 ilustra a cor média, resultante da análise do espetro RGB de todas as amostras, de cada classe (para os três canais de profundidade, foram calculados os seus valores médios).

Figura 48 - Cor média por classe

Esta figura ajuda a identificar diferenças e semelhanças que existem entre as classes. Por exemplo, as classes *Stroma* e *Debris* revelam uma cor média extremamente semelhante entre si, dificultando a sua distinção. Já, as classes *Tumor, Lympho*, *Mucosa* e *Complex* apesar de apresentarem uma cor média idêntica, é mais intuitiva a sua separação.

APÊNDICE D

ILUSTRAÇÃO DO *DIAGNOSISVIEWER*

Este apêndice exibe o resultado da plataforma desenvolvida. Ocasionalmente são incluídas breves descrições textuais com o intuito de facilitar a interpretação das páginas *web* ilustradas adiante.

A Figura 49 ilustra a página inicial da aplicação.

Figura 49 - Página inicial

A Figura 50 esquematiza a página "Sobre". Esta página é subdividida em três partes distintas: informação sobre o autor do projeto, descrição do âmbito do projeto e ainda uma explicação sumária do método proposto.

Figura 50 - Página *"Sobre"*

A Figura 51 exibe a página de *Login*.

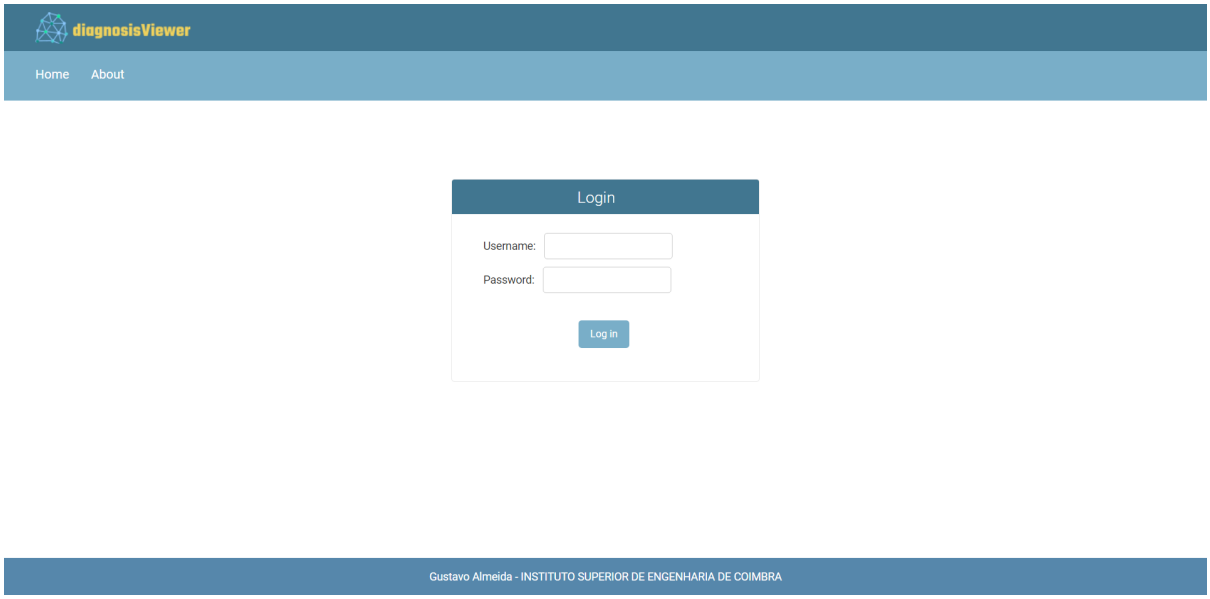

Figura 51 - Página de *Login*

A Figura 52 demonstra a página de registo.

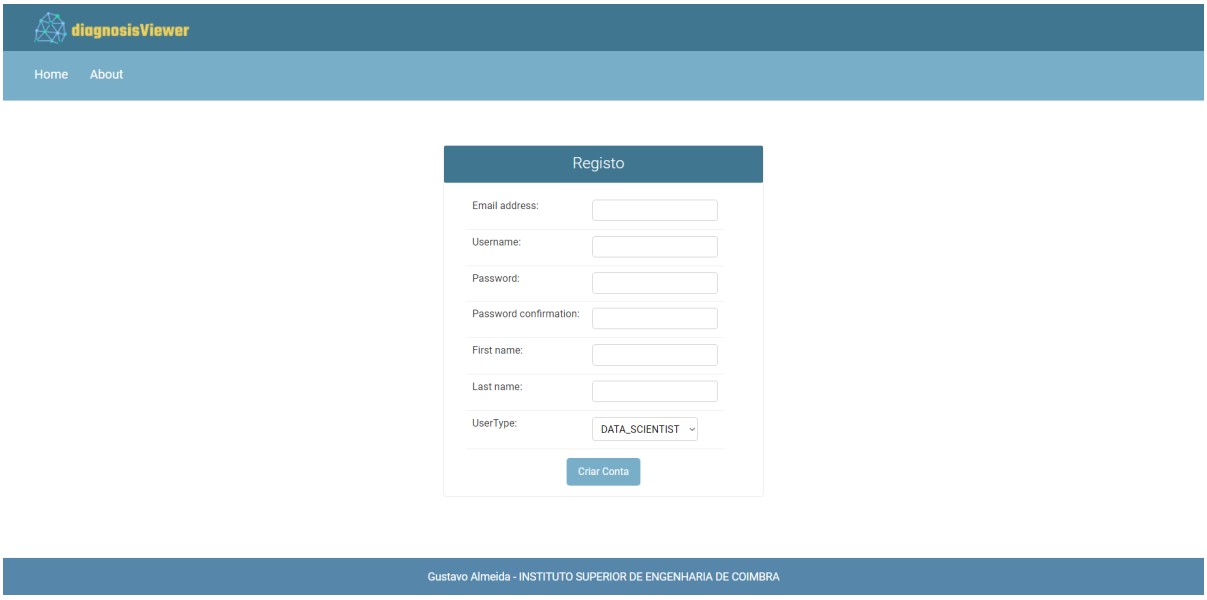

Figura 52 - Página de Registo

A Figura 53 exemplifica a página principal da plataforma, sendo exibida após a autenticação com sucesso do utilizador na *app*. Esta página proporciona acesso às funcionalidades do *DiagnosisViewer* e expõe informações sobre o histórico do utilizador e de toda a comunidade, com recurso a um *slideshow*.

Figura 53 - Página principal

A Figura 54 ilustra a página de edição do perfil do utilizador. Os campos editáveis são: *username* e seu nome (primeiro e último).

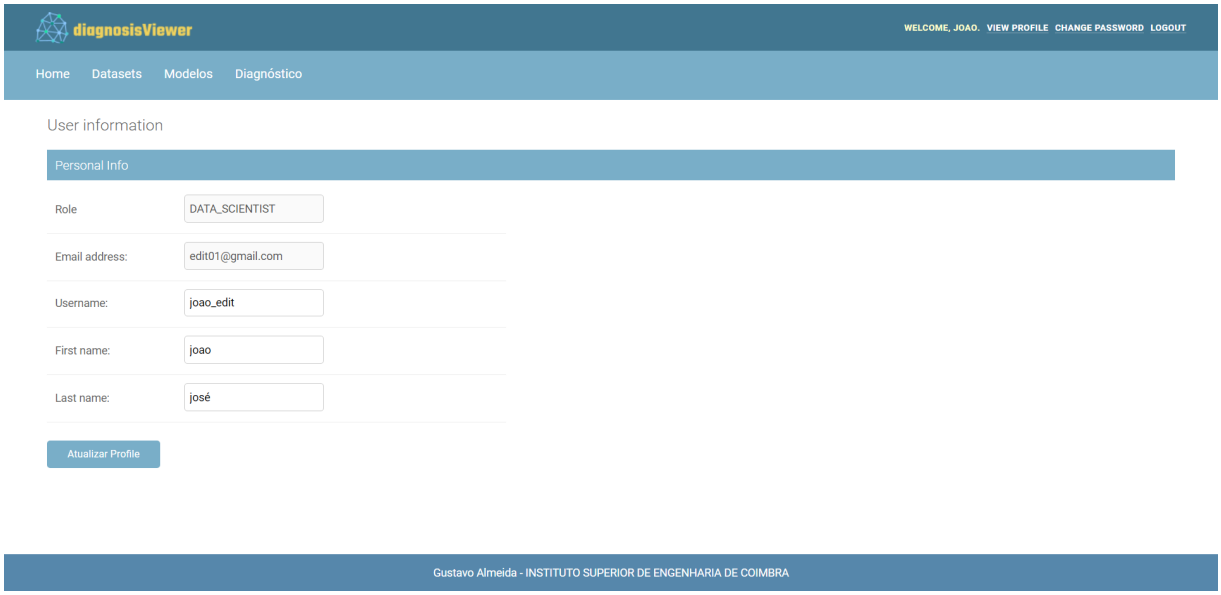

Figura 54 - Página de visualização/edição de perfil

A Figura 55 apresenta a página de alteração de *password*.

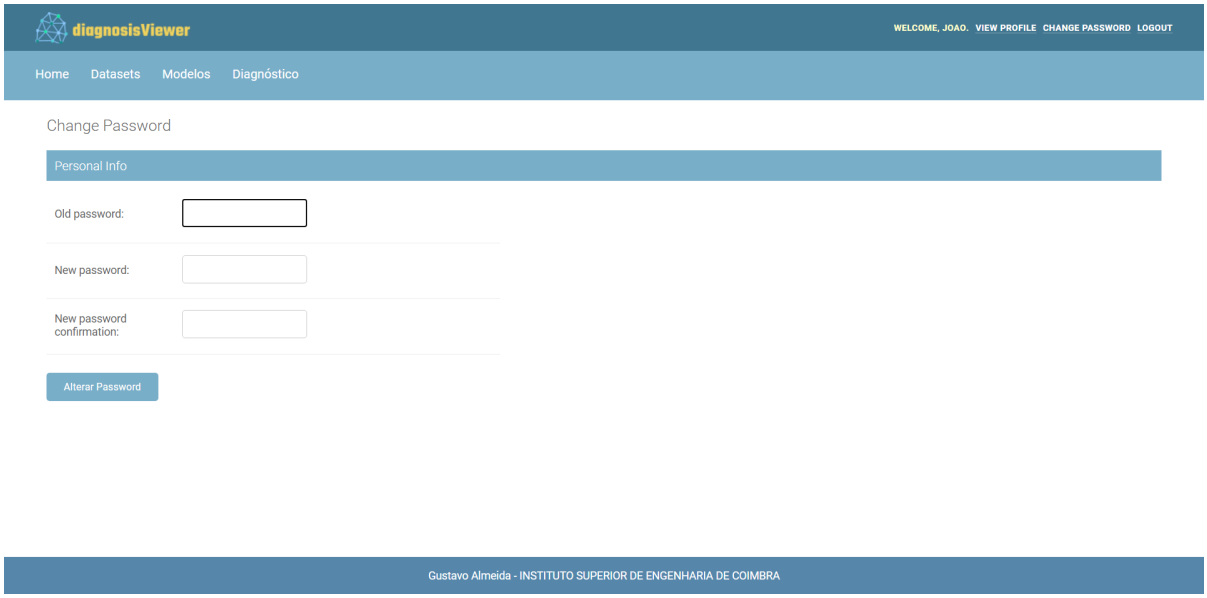

Figura 55 - Página de alteração de *password*

A Figura 56 esquematiza a listagem dos *dataset's* disponíveis na plataforma. Esta página contém um "painel de opções" que permite aos utilizadores efetuar a gestão interativa dos seus *dataset's*, ou criar um novo. A coluna "*EDIT/DROP*" especifica quais os *dataset's* que foram submetidos pelo utilizador (presença de *checkbox*), sendo que, o utilizador apenas pode apagar e alterar os *dataset's* que foram submetidos por si. A *checkbox* "Só criados por mim" filtra e lista, unicamente, os *dataset's* submetidos pelo utilizador.

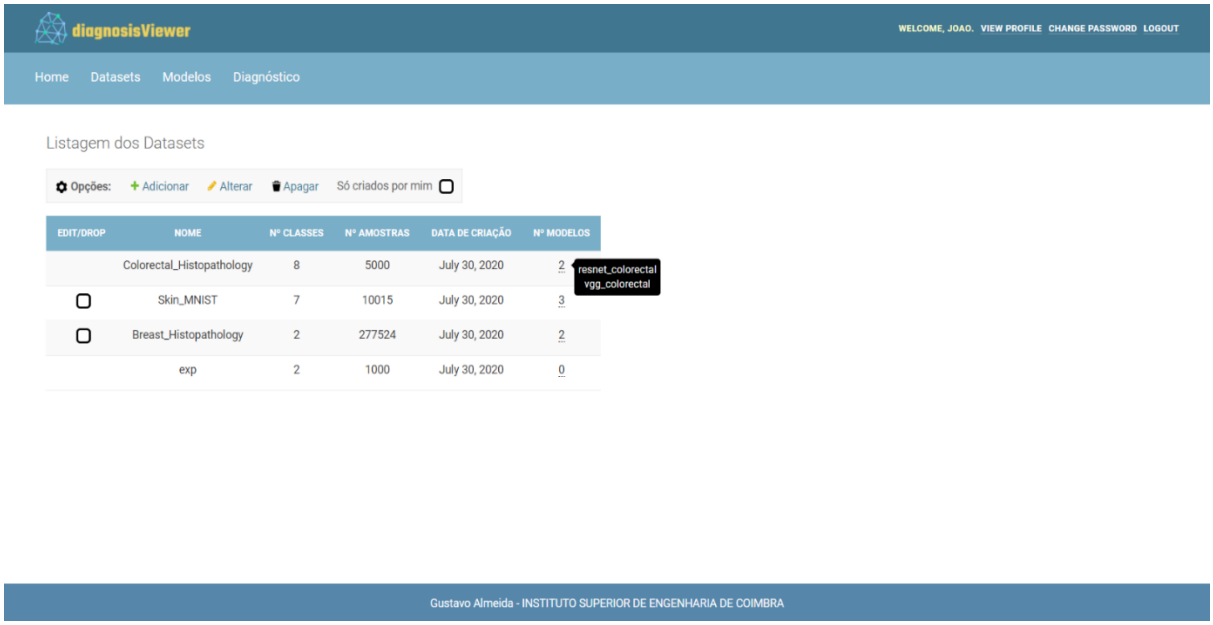

Figura 56 - Listagem dos *dataset's*

A Figura 57 exibe a página de criação de um *dataset*. Esta página é renderizada após "clique" na opção "Adicionar", presente na página de listagem de *dataset's* (Figura 56).

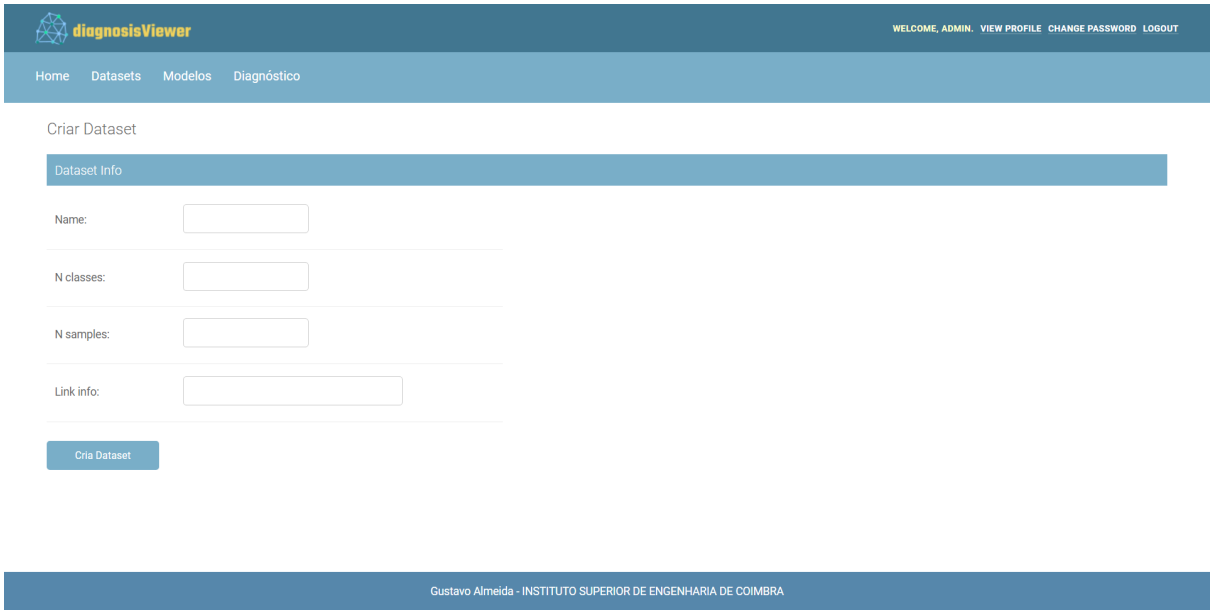

Figura 57 - Página de criação de um *dataset*

A página de edição de um *dataset* é exemplificada através da Figura 58. Esta página é renderizada após consequente seleção de um dos seus *dataset's* e posterior clique na opção "Alterar" (Figura 56). Caso, o utilizador não tenha selecionado previamente um *dataset*, é exibida a *popup* ilustrada na Figura 59.

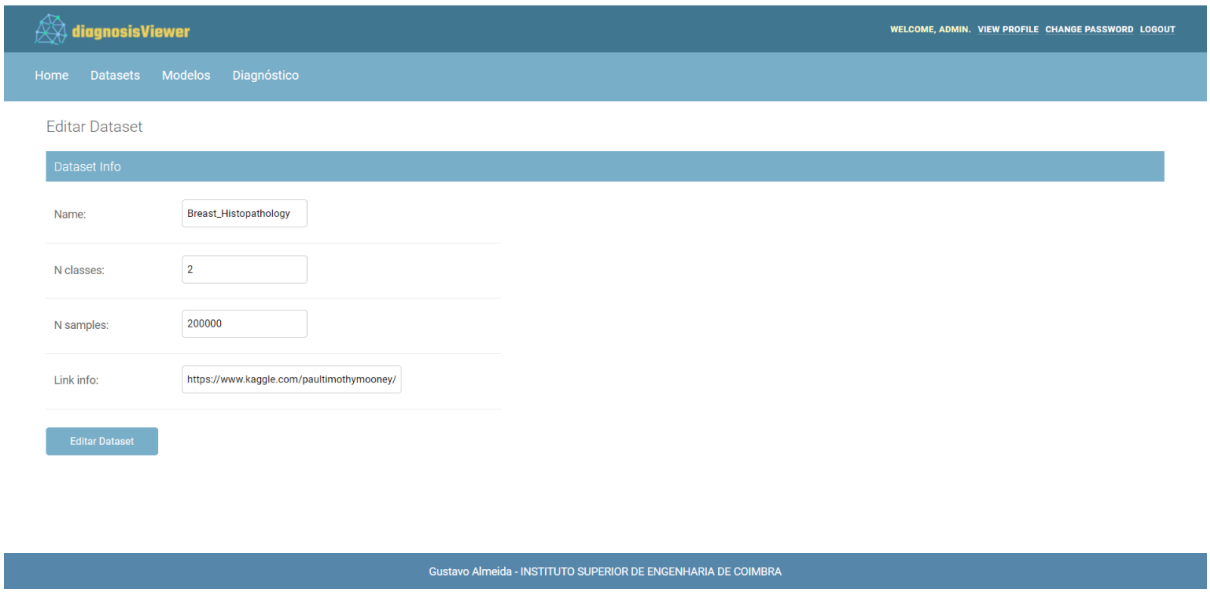

Figura 58 - Página de edição de um *dataset*

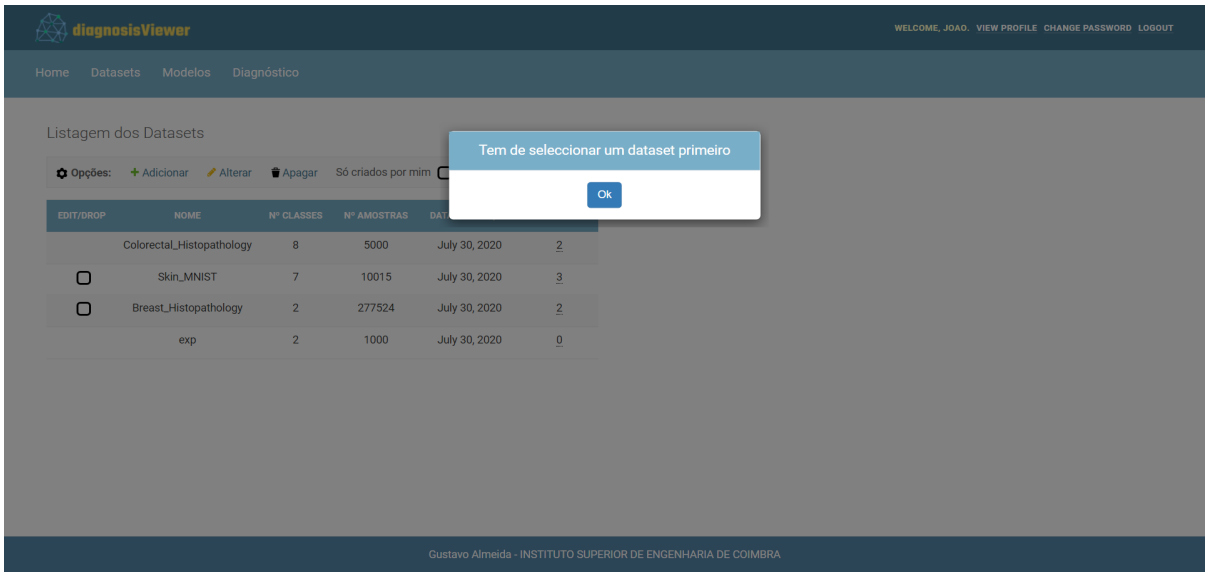

Figura 59 - Aviso, alteração/apagar *dataset*

A Figura 60 exemplifica a *popup*, de confirmação, que é exibida após clique na opção "Apagar" (Figura 56).

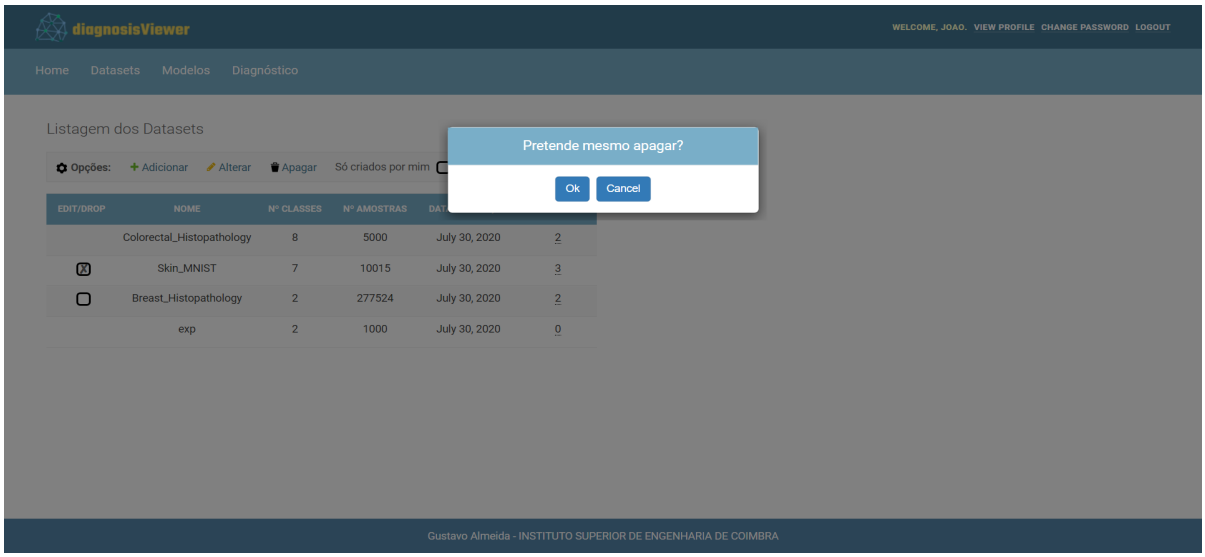

Figura 60 - *Popup* de confirmação - apagar *dataset*

A Figura 61 ilustra a listagem de todos os modelos existentes na aplicação, sendo a sua *interface* idêntica à página de listagem de *dataset's* (Figura 56). As restrições de alterar e de apagar aplicam-se de igual modo aos modelos. As respetivas páginas de adição, alteração e eliminação de modelos são demonstradas através das Figuras 62, 63 e 64. Os avisos e *popup's* de confirmação são utilizados de igual forma.

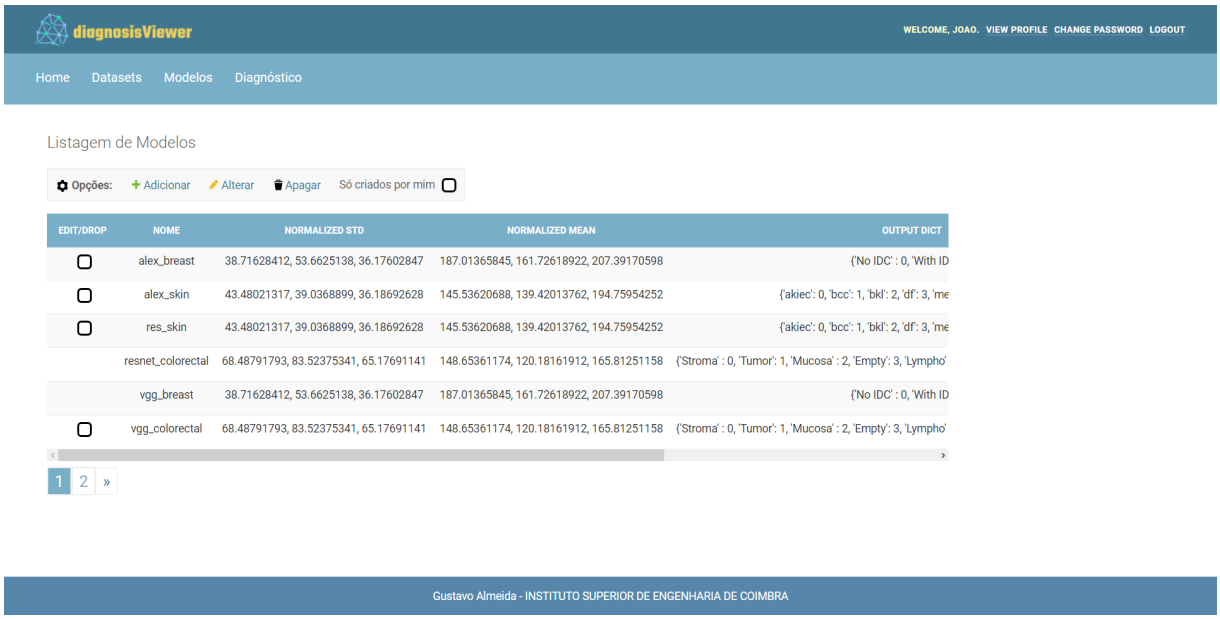

Figura 61 - Listagem dos modelos

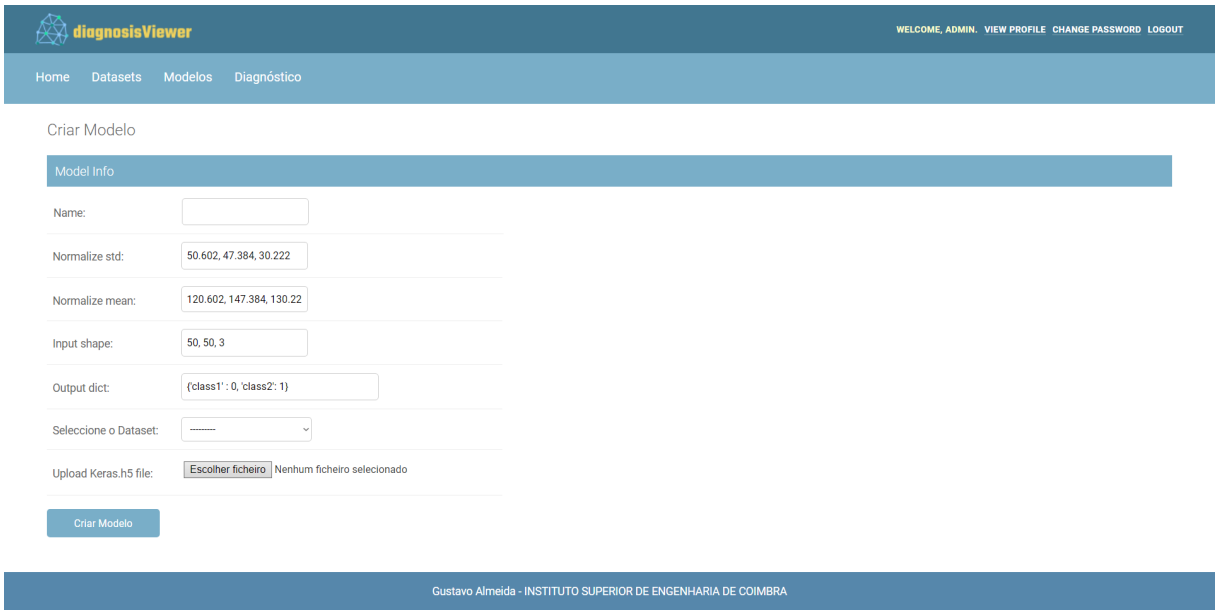

Figura 62 - Página de criação de um modelo

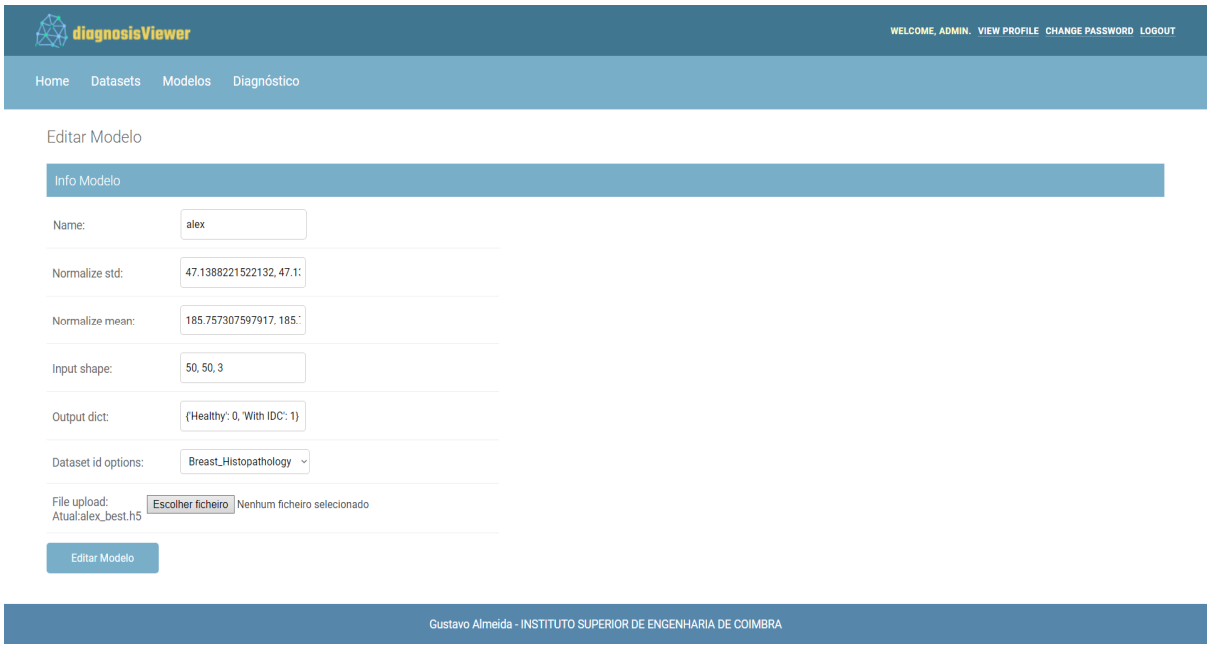

Figura 63 - Página de edição de um modelo

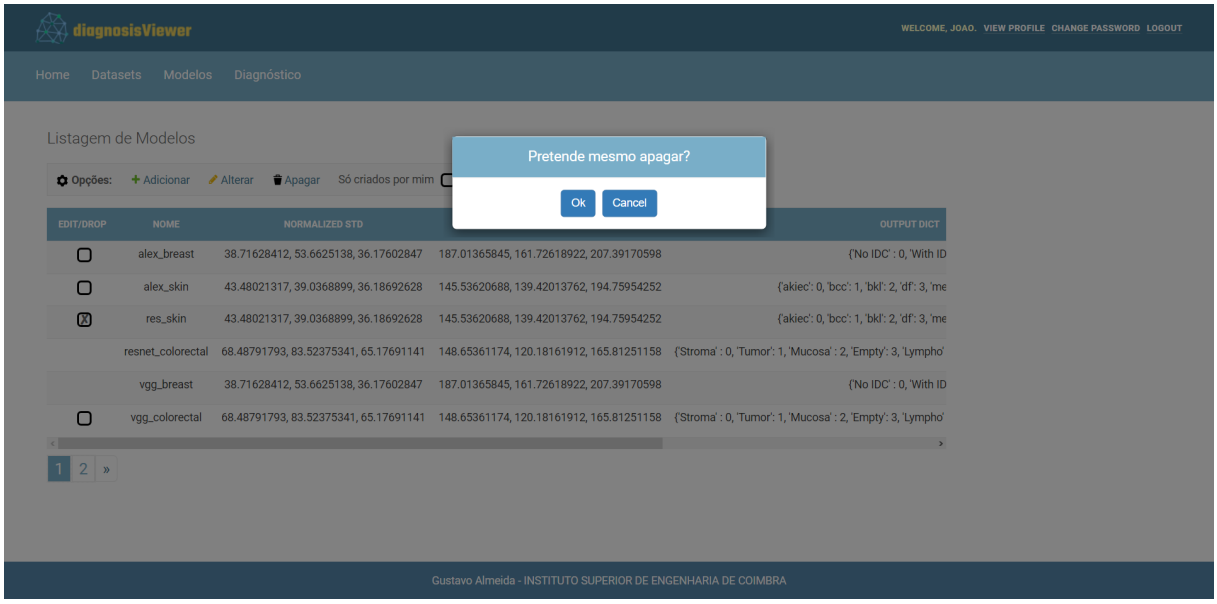

Figura 64 – Confirmação da eliminação de um modelo

As Figuras 65 e 66 demonstram duas situações distintas de resposta às ações efetuadas pelo utilizador. A Figura 65 apresenta o *feedback* demonstrado ao utilizador, após a criação com sucesso de um *dataset*. Já, a Figura 66 exibe o parecer que é indicado ao utilizador, após a "tentativa falhada" de edição de um dos seus modelos (incoerência entre o número de classes do *dataset* e o dicionário de *output* do modelo – o dicionário deve referenciar o mesmo número de classes que foram indicadas no ato de criação do *dataset*). A aplicação reúne um vasto conjunto de confirmações e de validações, que ajudam o utilizador a interpretar as respostas às suas ações.

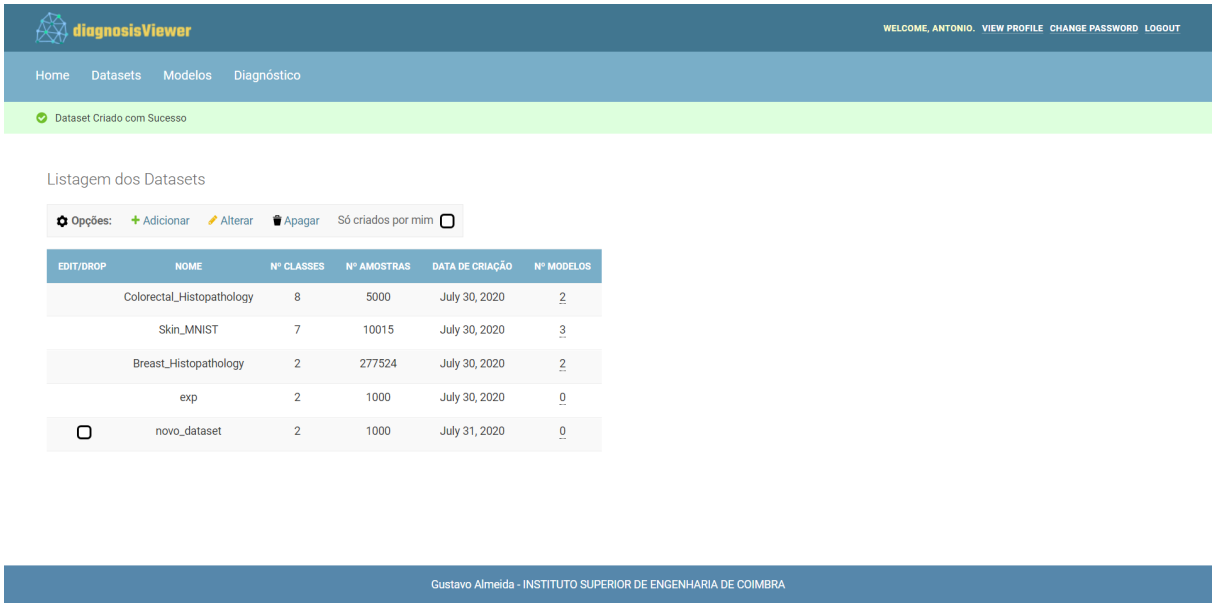

Figura 65 - Confirmação: *dataset* criado com sucesso

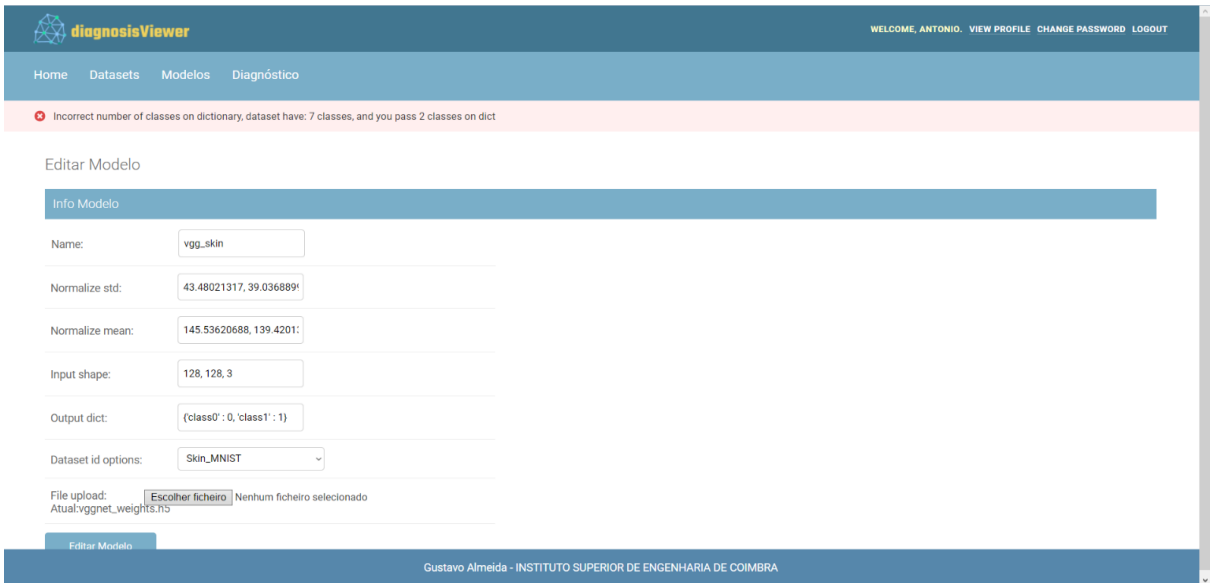

Figura 66 - Validação: incoerência entre o número de classes e o dicionário de *output*

A Figura 67 ilustra a página de diagnóstico. O utilizador para proceder à realização do diagnóstico necessita de indicar: a *benchmark*, selecionar um dos modelos disponíveis (para a *benchmark* selecionada) e efetuar *upload* da amostra em formato imagem. Após o envio do pedido (clique "Efetuar Previsão") é exibida uma página de *loading*, enquanto o utilizador aguarda pelos resultados do diagnóstico. A Figura 68 demonstra a página de *loading*.

Os resultados são depois exibidos através de uma *popup*, que enumera para cada classe, a respetiva probabilidade de representar a amostra. A Figura 69 exemplifica a *popup*. Caso surjam eventuais "problemas" durante o processo de diagnóstico, os mesmos são validados e descritos ao utilizador, de modo a que este possa proceder à sua resolução. A Figura 70 demonstra a abordagem de apresentação dos erros identificados durante a previsão.

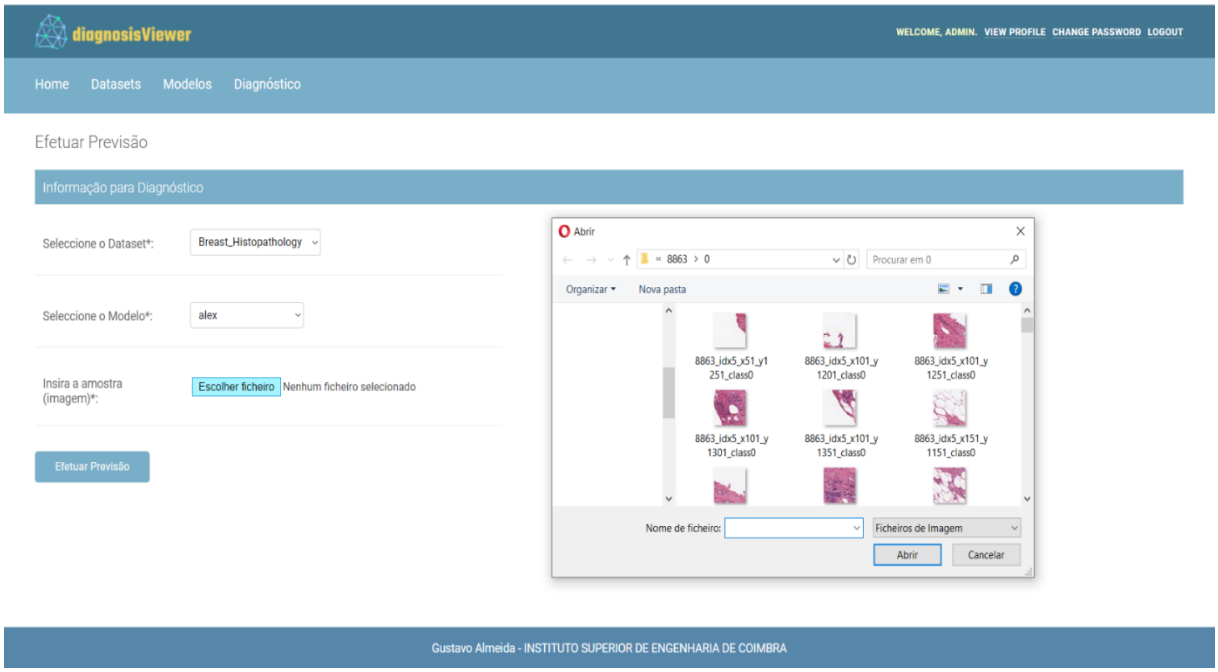

Figura 67 - Página de diagnóstico

Figura 68 - Página de *loading*

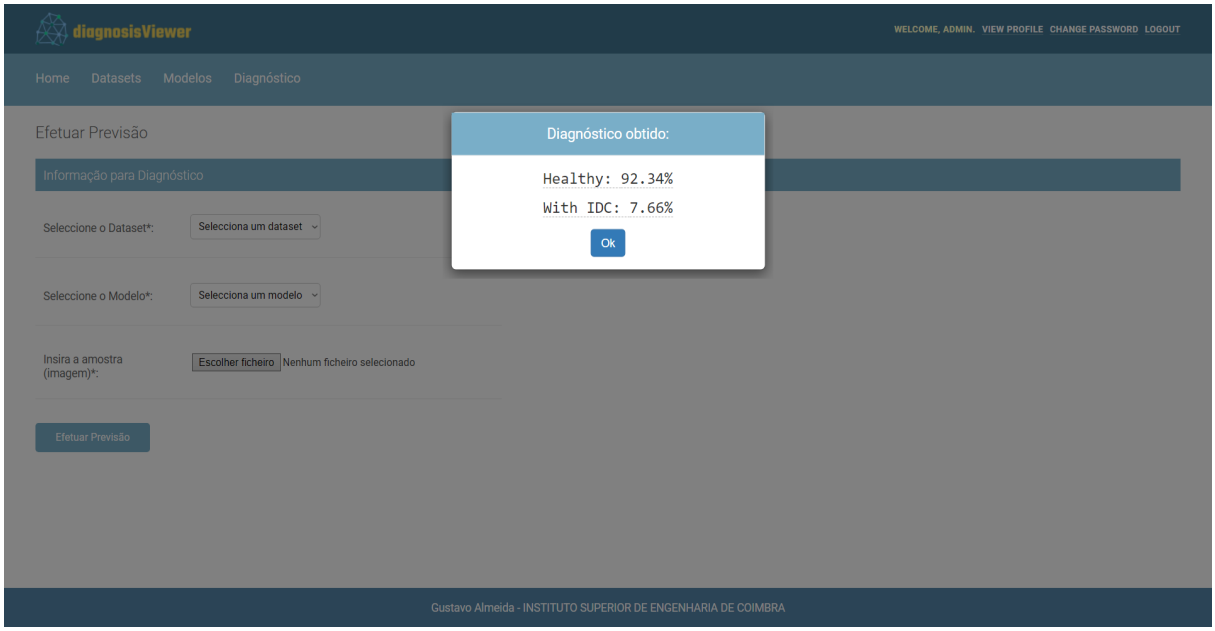

Figura 69 - *Popup* descritiva dos resultados obtidos

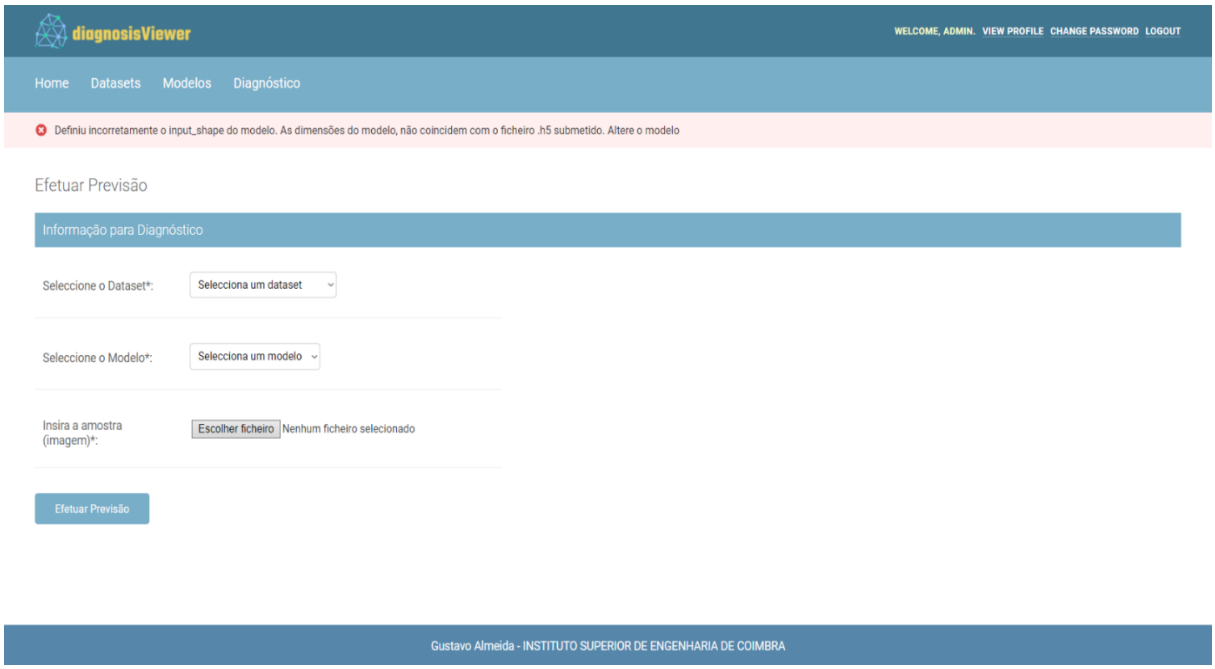

Figura 70 - Exemplo de um possível erro ocorrido na previsão

APÊNDICE E
VARIAÇÃO DA *FITNESS* - PSO

Seguidamente, para cada arquitetura otimizada e em conformidade com as três *benchmarks* avaliadas, é exposta a variação da melhor *fitness* ao longo das iterações.

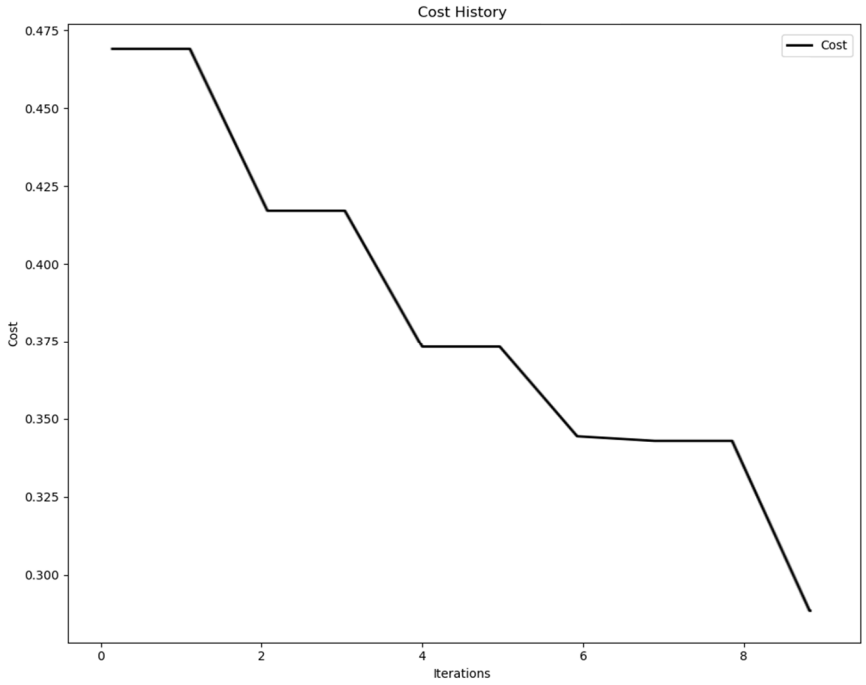

Figura 71 - Variação da melhor *fitness* - *AlexNet*, *Colorectal Histopathology*

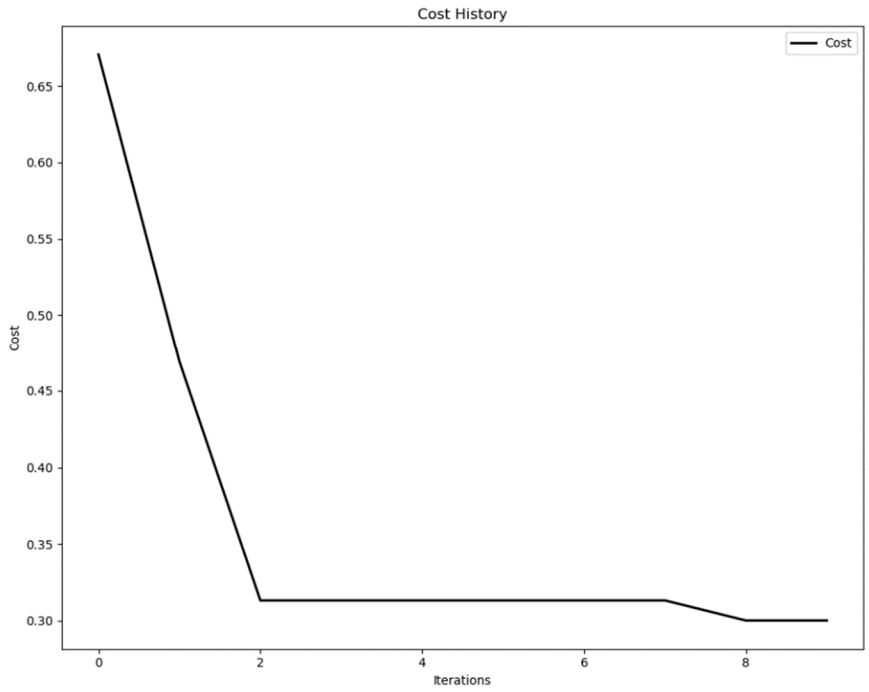

Figura 72 - Variação da melhor *fitness* - *VGGNet*, *Colorectal Histopathology*

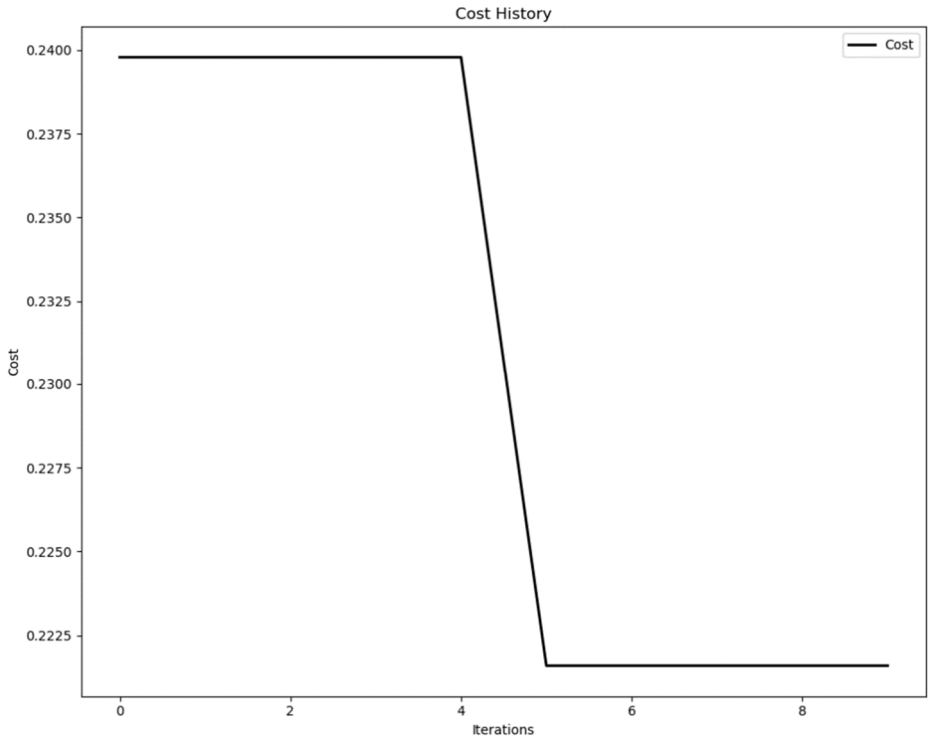

Figura 73 - Variação da melhor *fitness* - *ResNet*, *Colorectal Histopathology*

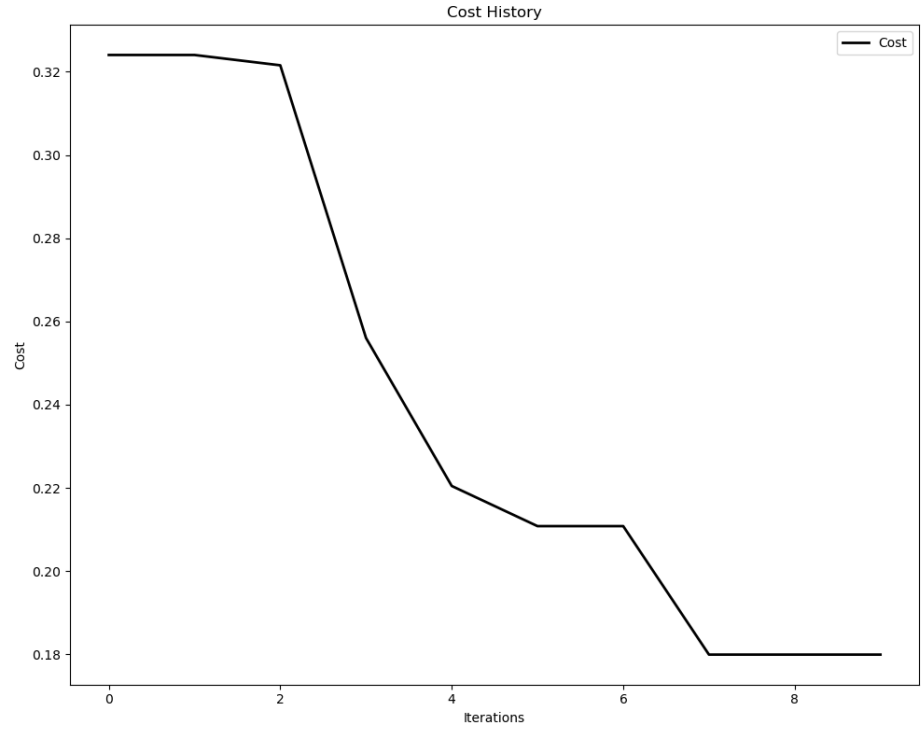

Figura 74 - Variação da melhor *fitness* - *DenseNet*, *Colorectal Histopathology*

Analisando as figuras 71, 72, 73 e 74 é possível concluir que as partículas convergiram rapidamente na otimização das arquiteturas *VGGNet* e *ResNet*. O ligeiro decréscimo da *fitness*, que ocorreu após um longo período de estagnação, na otimização de ambas as arquiteturas, deve-se provavelmente ao fator estocástico presente no processo de treino das soluções

(biblioteca *Keras* inviabiliza a reprodutibilidade dos resultados), ou ao fator *estocástico* do algoritmo PSO. A convergência das partículas na otimização da arquitetura *DenseNet* revelou-se mais tardia. Por último, na otimização da rede *AlexNet*, as partículas ao fim das 10 iterações não apresentaram indícios de convergência. Ou seja, seria necessário considerar um maior número de iterações, de modo a verificar se as partículas tenderiam a convergir, ou se existia ainda a possibilidade de redução do custo.

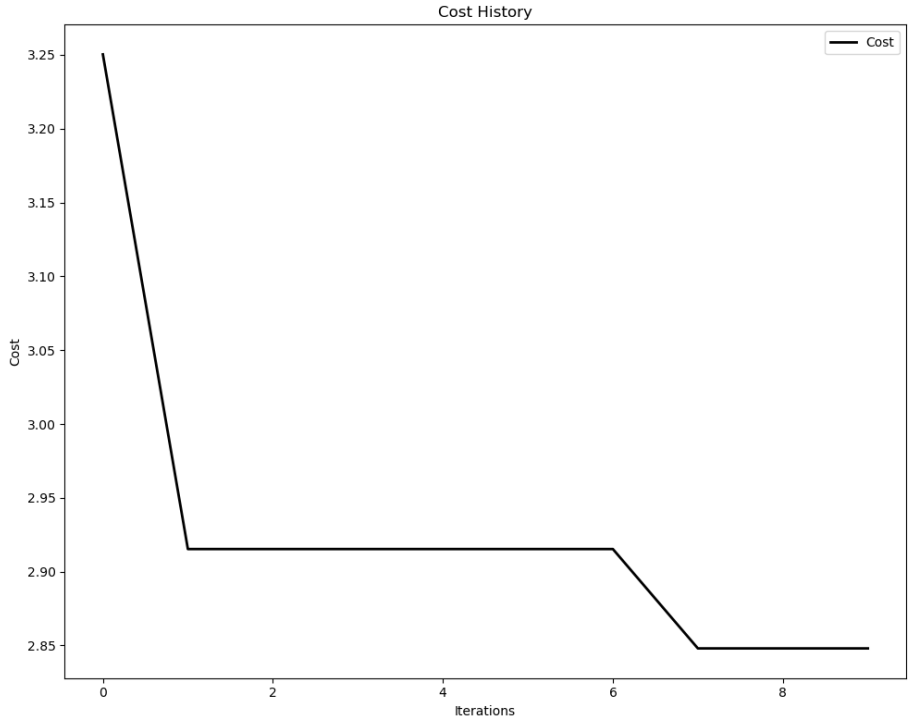

Figura 75 - Variação da melhor *fitness* - *AlexNet*, *Skin Mnist*

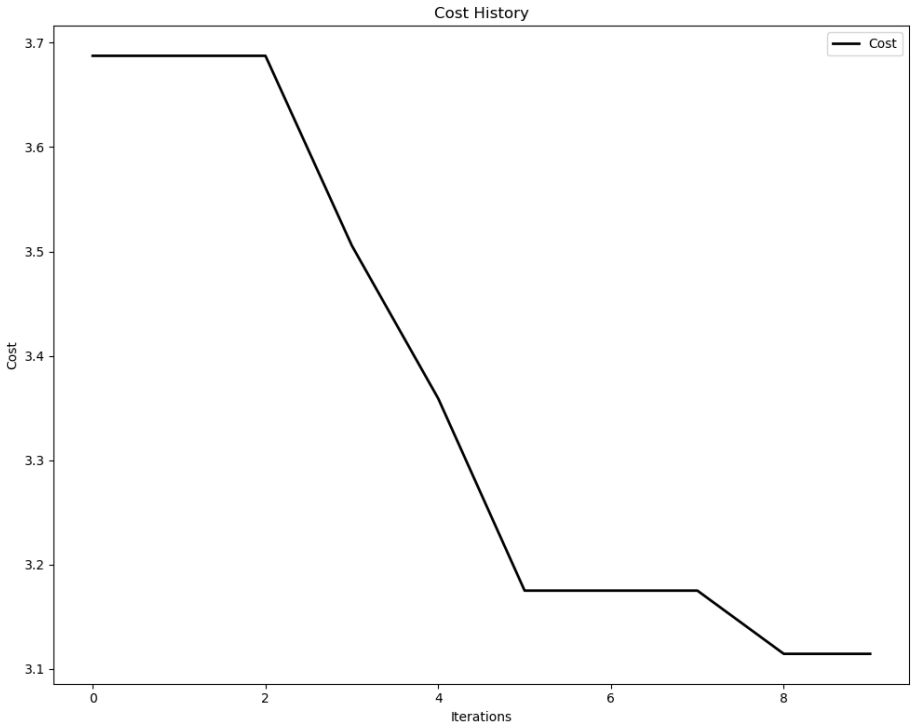

Figura 76 - Variação da melhor *fitness* - *VGGNet*, *Skin Mnist*

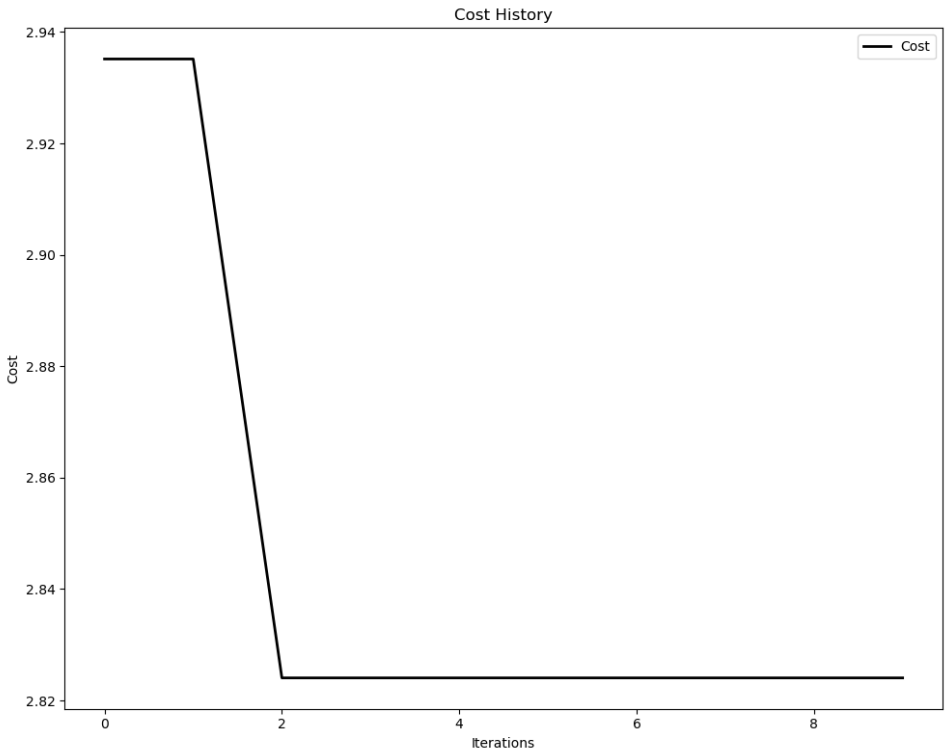

Figura 77 - Variação da melhor *fitness* - *ResNet*, *Skin Mnist*

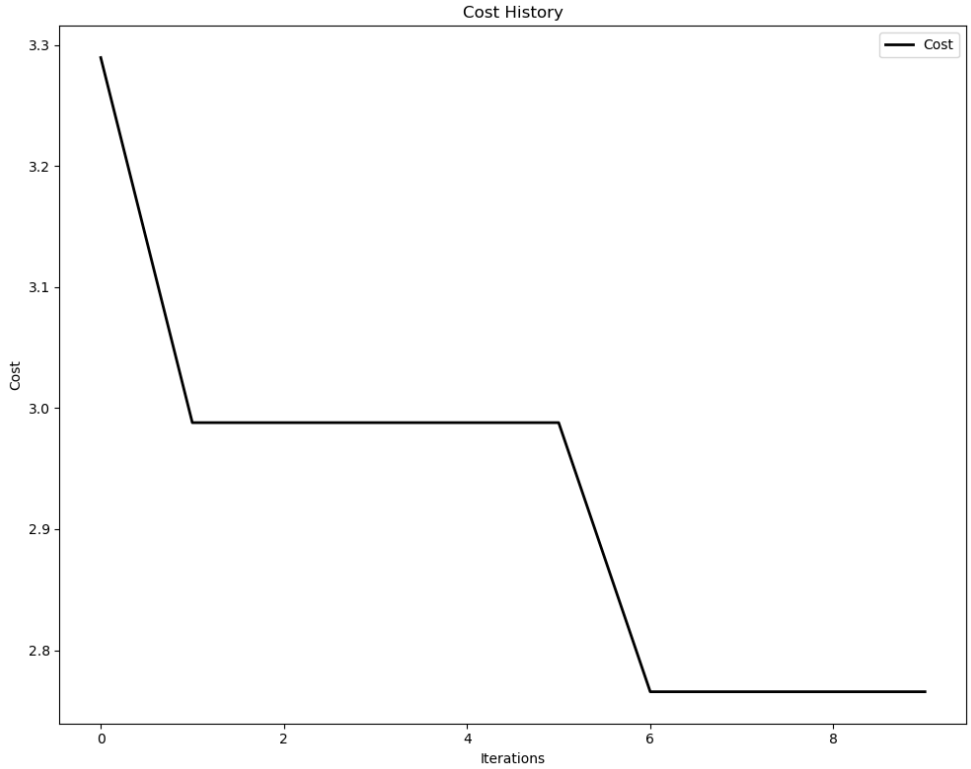

Figura 78 - Variação da melhor *fitness* - *DenseNet*, *Skin Mnist*

À exceção da rede *VGGNet*, o processo de otimização das restantes arquiteturas apresentou rápida convergência. As variações da *fitness*, evidenciadas após um longo período de estagnação, na otimização das arquiteturas *AlexNet* e *DenseNet*, devem-se muito provavelmente ao facto do PSO ser um algoritmo estocástico, possibilitando assim que algumas partículas se

possam "afastar" da solução para a qual o enxame convergiu, sendo, eventualmente, identificadas soluções com melhor *fitness*. Como descrito no Capítulo 7, o processo de treino das redes CNN não é determinista, ou seja, partículas com configurações muito semelhantes/iguais (valores dos hiperparâmetros idênticos) podem apresentar desempenhos com ligeiras flutuações. A ligeira variação da *fitness*, ocorrida após a convergência das partículas, na otimização da rede *AlexNet*, pode estar relacionada com este acontecimento.

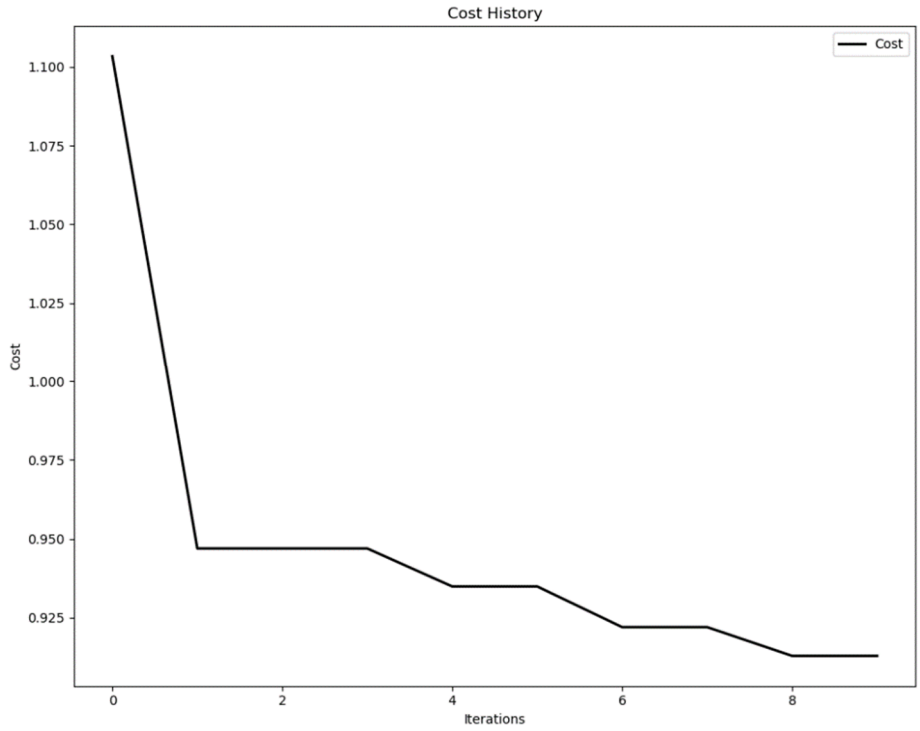

Figura 79 - Variação da melhor *fitness* - *AlexNet*, *Breast Histopathology*

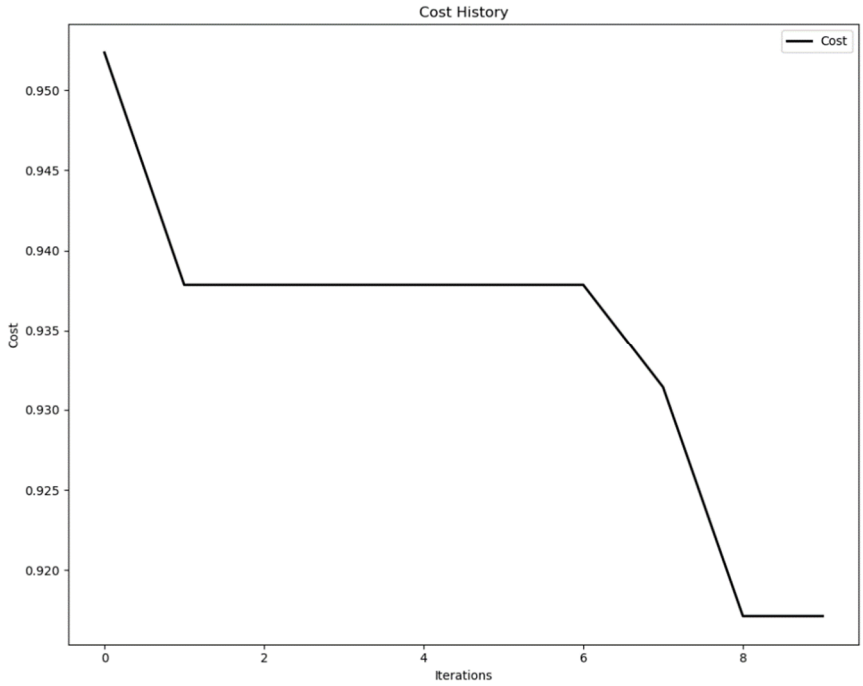

Figura 80 - Variação da melhor *fitness* - *VGGNet*, *Breast Histopathology*

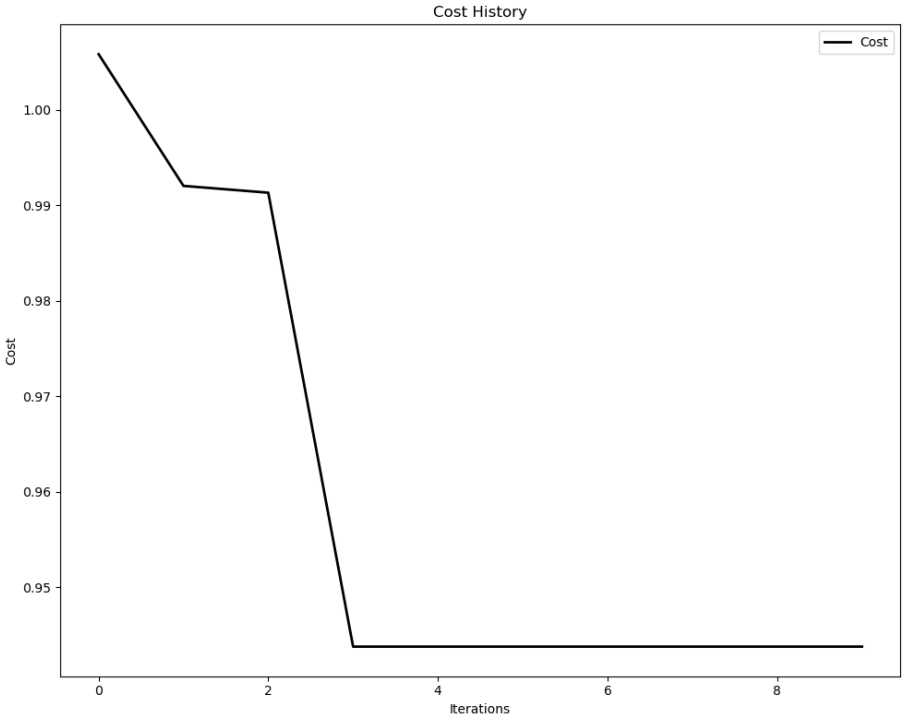

Figura 81 - Variação da melhor *fitness* - *ResNet*, *Breast Histopathology*

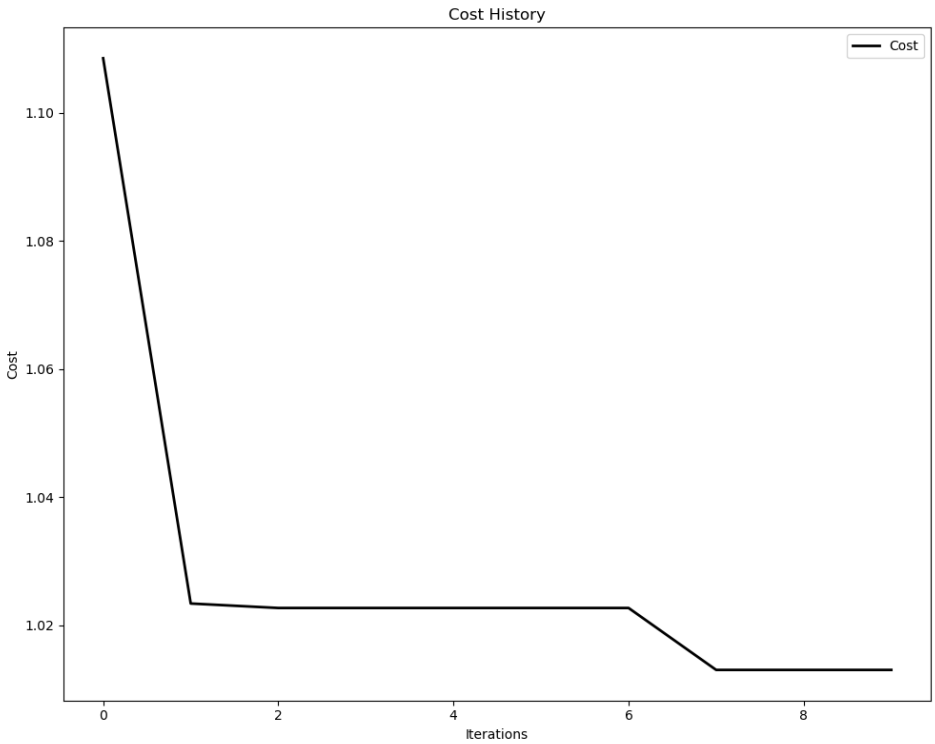

Figura 82 - Variação da melhor *fitness* - *DenseNet*, *Breast Histopathology*

O processo de otimização de todas as arquiteturas, à exceção da rede *AlexNet*, apresentou rápida convergência. O decréscimo residual do custo, evidenciado após a convergência prematura das partículas, na otimização das redes *VGGNet* e *DenseNet*, está muito provavelmente associado ao carácter estocástico do processo de treino das soluções. A componente estocástica do PSO pode também ter sido uma das causas desta variação.