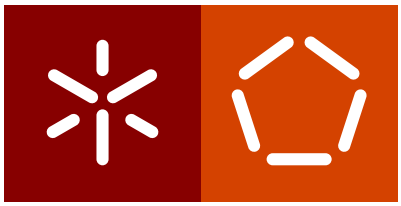
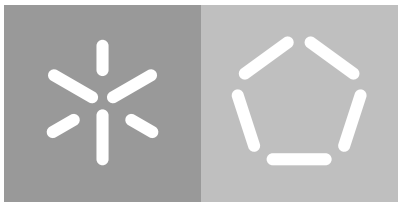


Universidade do Minho
Escola de Engenharia
Departamento de Informática

Mateus Ferreira da Silva

Diagnóstico da Doença de Alzheimer com Redes Neuronais Profundas

Julho 2022



Universidade do Minho
Escola de Engenharia
Departamento de Informática

Mateus Ferreira da Silva

Diagnóstico da Doença de Alzheimer com Redes Neuronais Profundas

Relatório de Dissertação
Mestrado Integrado em Engenharia Informática

Dissertação supervisionada por
Professor António Joaquim André Esteves

Julho 2022

DIREITOS DE AUTOR E CONDIÇÕES DE UTILIZAÇÃO DO TRABALHO POR TERCEIROS

AGRADECIMENTOS

Em primeiro lugar fica um agradecimento especial ao meu orientador nesta dissertação, o professor António Esteves, pela paciência e pela dedicação nesta investigação.

A toda a minha família deixo o meu agradecimento, porque sem vocês não seria a pessoa que sou hoje.

Passando agora para agradecimentos mais pessoais, em primeiro fica o agradecimento especial aos meus pais pelo suporte ao longo desta etapa. Só nós sabemos como foi esta fase para mim, e só tenho uma palavra a dizer: obrigado.

Claro que ias ter um paragrafo só para ti, Moisés. O mesmo se aplica a ti o que referi sobre os pais. Não temos uma relação baseada em afetos ou palavras bonitas, mas aproveito aqui para te dizer que sempre foste o meu pilar em tudo, és o meu orgulho e sempre foste um exemplo a seguir.

Claro que, após os meus familiares mais próximos, tinham de ser vocês, tia Guida, tio Jonas, primo Tiago e primo Tomás. Em setembro de 2016 a vossa casa ganhou mais um membro, e fui tão bem recebido que imediatamente me senti em casa. Foi a minha casa por quase 4 anos, e vocês foram como pais e irmãos para mim. O meu agradecimento por tudo o que por mim fizeram. Prometo continuar a fazer as visitas à minha casa das Taipas. Por falar em Taipas, por aí também fiz uns amigos especiais, a família Ferreira, à qual devo um especial agradecimento à Nair.

Dentro desta incrível e grande família que tenho, há nomes que merecem destaque, por isso, fica aqui o meu agradecimento especial à prima Eduarda, prima Juliana, prima Margarida, ao primo André, à prima Susete, ao avô Manuel, ao avô Miro, avó Zira, tia Glória, tio José Manuel e aos meus padrinhos por todo o apoio.

Por fim, e não menos importantes, fica o agradecimento aos meus amigos. Claro que teria de começar por ti, Paulinha, e tu sabes o porquê. Fica aqui também o agradecimento ao Xico, ao Pedro, ao Vasco, à Pedroso, à Sofia e à Rafaela. Também àqueles que foram os meus parceiros e amigos nesta aventura universitária e com os quais espero manter o contacto por muitos anos: Filipa, Gui, Guilherme, Joana, Sara, Rafaela, João, Luisinho, Braga e Shahzod, o meu obrigado por todos os momentos que passamos juntos.

Para terminar devem ser aqui mencionadas as pessoas responsáveis pela criação do conjunto de dados que permitiu esta investigação. A recolha e partilha de dados para este projecto foi fundada pela *Alzheimer's Disease Neuroimaging Initiative* (ADNI) (*National Institutes of Health Grant U01 AG024904*) e DOD ADNI (*Department of Defense award number W81XWH-12-2-0012*). A ADNI é financiada pelo *National Institute on Aging*, o *National*

Institute of Biomedical Imaging and Bioengineering, e através de generosas contribuições dos seguintes: *AbbVie, Alzheimer's Association; Alzheimer's Drug Discovery Foundation; Araclon Biotech; BioClinica, Inc.; Biogen; Bristol-Myers Squibb Company; CereSpir, Inc.; Cogstate; Eisai Inc.; Elan Pharmaceuticals, Inc.; Eli Lilly and Company; EuroImmun; F. Hoffmann-La Roche Ltd* e a sua empresa afiliada *Genentech, Inc.; Fujirebio; GE Healthcare; IXICO Ltd.; Janssen Alzheimer Immunotherapy Research & Development, LLC; Johnson & Johnson Pharmaceutical Research & Development LLC; Lumosity; Lundbeck; Merck & Co.., Inc.; Meso Scale Diagnostics, LLC.; NeuroRx Research; Neurotrack Technologies; Novartis Pharmaceuticals Corporation; Pfizer Inc.; Piramal Imaging; Servier; Takeda Pharmaceutical Company; e Transition Therapeutics*. Os *Canadian Institutes of Health Research* que estão a fornecer fundos para apoiar os sítios clínicos ADNI no Canadá. As contribuições do sector privado são facilitadas pela *Foundation for the National Institutes of Health* (www.fnih.org). A organização bolseira é o *Northern California Institute for Research and Education*, e o estudo é coordenado pelo *Alzheimer's Therapeutic Research Institute* da *University of Southern California*. Os dados ADNI são divulgados pelo *Laboratory for Neuro Imaging* da *University of Southern California*.

DECLARAÇÃO DE INTEGRIDADE

Declaro ter atuado com integridade na elaboração do presente trabalho académico e confirmo que não recorri à prática de plágio nem a qualquer forma de utilização indevida ou falsificação de informações ou resultados em nenhuma das etapas conducente à sua elaboração.

Mais declaro que conheço e que respeitei o Código de Conduta Ética da Universidade do Minho.

Assinado por : **MATEUS FERREIRA DA SILVA**
Num. de Identificação: 15541836
Data: 2022.07.11 18:46:49+01'00'

11/07/2022

CHAVE MÓVEL

RESUMO

A doença de Alzheimer é o tipo mais predominante de demência e, apesar de não existir cura para a mesma, o seu diagnóstico prematuro é fundamental para um tratamento efetivo e que permita retardar o progresso dos sintomas. Desta forma, nos últimos anos, tem surgido um grande interesse em estudar e desenvolver sistemas automáticos de diagnóstico que usam como fonte de dados os exames médicos realizados pelos pacientes.

Esta dissertação enquadra-se na temática da utilização de aprendizagem profunda para diagnosticar a doença de Alzheimer. Pretende-se que seja avaliado o desempenho de redes neuronais profundas existentes da forma mais real possível, e que seja proposta uma arquitetura com bom desempenho, que possa ser usada num sistema de diagnóstico assistido por computador. A capacidade da aprendizagem profunda em encontrar padrões ocultos em imagens médicas permite reduzir o erro de diagnóstico humano e auxiliar num diagnóstico muito mais preciso.

É pretendido que, com base numa única imagem por paciente a rede neuronal proposta seja capaz de fazer um diagnóstico sobre a doença de Alzheimer. Para isso, serão utilizadas imagens do cérebro obtidas pela técnica de Ressonância Magnética Estrutural, adquiridas a partir do conjunto de dados da Iniciativa de Neuroimagem da Doença de Alzheimer.

Para a concretização do objetivo proposto, foram estudados e implementados, num cenário experimental recorrendo a ferramentas simples, os métodos mais indicados para a realização desta tarefa.

Os resultados mostram que as Redes Neuronais Convolucionais (CNNs) permitem construir modelos com enorme potencial, contudo, a sua utilização em ambientes reais ainda não é muito viável nos dias de hoje.

Palavras-chave: Doença de Alzheimer, Diagnóstico Assistido por Computador, Défice Cognitivo Ligeiro, Ressonância Magnética Estrutural, Aprendizagem Profunda, Redes Neuronais Convolucionais

ABSTRACT

Alzheimer's disease is the most prevalent type of dementia and, although there is no cure for it, its early diagnosis is fundamental for an effective treatment to slow the progression of symptoms. Therefore, in the last few years, there has been a great interest in studying and developing automatic diagnostic systems that use medical examinations performed by patients as a source of data.

This dissertation addresses the use of deep learning models to diagnose the Alzheimer's disease. It is intended to evaluate the performance of existing deep neural networks as realistically as possible, and to propose an architecture with good performance that can be used in a computer-assisted diagnostic system. The ability of deep learning to find hidden patterns in medical images makes it possible to reduce human diagnostic error and enables a much more accurate diagnosis.

It was intended that based on a single image per patient the proposed neural network was able to make a diagnosis about Alzheimer's disease. The present work was carried out with brain images obtained by the Structural Magnetic Resonance Imaging technique, acquired from the Alzheimer's Disease Neuroimaging Initiative (ADNI) dataset.

To achieve the proposed goal, the most suitable methods for performing the mentioned task were studied and implemented in an experimental scenario using simple tools.

The results show that Convolutional Neural Networks allow the construction of very powerful models, however, their use in real environments is still not very feasible nowadays.

Keywords: Alzheimer's disease, Computer Aided Diagnosis, Mild Cognitive Impairment, Structural MRI, Deep Learning, Convolutional Neural Networks

CONTEÚDO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE LISTAGENS

SIGLAS

ADNI Iniciativa de Neuroimagem da Doença de Alzheimer. 1, 19, 20, 24, 77–79, 81–83, 123
ANN Rede Neuronal Artificial. 1, 35, 36
AUC *Area Under the Curve*. 1, 76, 79, 106–108, 110–112, 115, 116

BN *Batch Normalization*. 1, 55, 56, 64, 65

CAD Diagnóstico Assistido por Computador. 1, 28
CADe Deteção Assistida por Computador. 1, 31
CADx Diagnóstico Assistido por Computador. 1, 31, 32, 71, 118, 122
CAS Sistema Assistido por Computador. 1, 15, 16, 23, 27
CNN Rede Neuronal Convolucional. 1, 18, 23, 24, 28, 31, 32, 35–41, 43, 45, 46, 49, 50, 54, 57, 63, 66, 77–80, 83, 90, 99, 101, 102, 108, 109, 117–119, 121–123
CRISP-DM *Cross-industry Standard Process for Data Mining*. 1, 20

DA Doença de Alzheimer. 1, 15–29, 32, 34, 36, 60–62, 69, 71, 72, 74, 76–82, 118, 120–123
DCL Défice Cognitivo Ligeiro. 1, 15, 16, 26, 28, 32, 34, 77–79, 81, 82
DL Aprendizagem Profunda. 1, 15–19, 21, 23, 25, 27–32, 35, 36, 40, 61–63, 66, 67, 69, 72, 76–78, 83, 96, 99, 118, 122, 123
DNN Rede Neuronal Profunda. 1, 18, 32, 36, 54, 60, 65, 68, 69, 71, 72, 77, 78, 90

ECE Expedente Clínico Eletrónico. 1, 27, 31, 71

GAN *Generative Adversarial Networks*. 1, 67–69

IA Inteligência Artificial. 1, 15, 16, 18, 20, 23, 27, 29–31, 69, 71, 122, 123

MIEI Mestrado Integrado em Engenharia Informática. 1
ML Aprendizagem Automática. 1, 16–18, 21–23, 26–28, 30, 32, 33, 35, 75, 97, 118
MRI Ressonância Magnética. 1, 16–24, 27, 28, 60–62, 67, 68, 70, 71, 77–79, 82, 83, 87, 114, 118–120, 122, 123

PET *Positron Emission Tomography*. 1, 17, 23, 70, 119

RGPD Regulamento Geral de Proteção de Dados. 1, 71
ROC *Receiver Operating Characteristic*. 1, 75, 76, 108, 110, 112, 116
ROI Região de Interesse. 1, 61, 77

SVM *Support Vector Machine*. 1, 28, 118

UM Universidade do Minho. 1

1

INTRODUÇÃO

O capítulo inicial desta dissertação introduz o tema a ser desenvolvido e investigado. De certa forma, é apresentada uma visão geral da Inteligência Artificial (IA) aplicada à medicina, acabando também por aprofundar o tema do diagnóstico de Doença de Alzheimer (DA) recorrendo a Aprendizagem Profunda (DL)[1].

Neste capítulo pode-se encontrar uma contextualização do tema e as razões que levaram à realização desta investigação, os objetivos da dissertação, os métodos e as técnicas usadas na investigação. Por fim, este capítulo termina com uma apresentação da estrutura da dissertação.

1.1 CONTEXTUALIZAÇÃO E MOTIVAÇÃO

A DA é uma desordem neurodegenerativa irreversível caracterizada pela deterioração progressiva da memória e das funções cognitivas (Li and Liu, 2018; Duraisamy et al., 2018). Esta doença é a forma mais comum de demência e ocorre principalmente no final da vida humana (McKhann et al., 1984). Genericamente, a DA caracteriza-se pela destruição progressiva de células do cérebro (Islam and Zhang, 2018), o que acaba por trazer inúmeras consequências na vida humana tais como a perda de memória, a perda de funções mentais e até mesmo a perda da capacidade de continuar a realizar as atividades diárias. Para além disso, e como aspeto fundamental, é de notar que, existe um espectro contínuo desde a cognição normal até à DA, incluindo o Défice Cognitivo Ligeiro (DCL)[2] que é geralmente considerado precursor clínico da DA (Ding et al., 2019; Li and Liu, 2018).

Presentemente ainda não há cura eficaz para a DA, mas alguns tratamentos podem ser feitos de modo a atrasar a sua progressão, especialmente se a DA for diagnosticada numa fase prematura (Li and Liu, 2018). Deste modo, um diagnóstico eficaz, preciso e precoce desta desordem é um ponto crítico e fundamental no seu tratamento. Assim sendo, muitos investigadores dedicam-se ao desenvolvimento de Sistema Assistido por Computador (CAS)[3] capazes de diagnosticar a DA ainda em fases iniciais (Basaia et al., 2019).

1 *Deep Learning*, na terminologia inglesa.
2 *Mild Cognitive Impairment*, na terminologia inglesa.
3 *Computed-Aided Systems*, na terminologia inglesa.

Ainda não existe uma forma exata de diagnosticar a DA. Para uma avaliação médica adequada são necessários testes físicos, testes neurobiológicos, *Mini-Mental State Examination* (MMSE) e a história detalhada do paciente. Recentemente, os médicos estão também a utilizar a Ressonância Magnética (MRI)[4] do cérebro para o diagnóstico da DA (Islam and Zhang, 2018). As imagens de MRI estrutural, que fornecem biomarcadores de perda neuronal, são parte integrante da avaliação clínica de pacientes com suspeitas de DA (Basaia et al., 2019).

Os diversos avanços, teóricos e práticos, na área de IA, especialmente nas últimas duas décadas, fizeram com que a introdução desta se tenha materializado em diversos campos, incluindo a medicina, a educação, o entretenimento, entre outras. No que toca à aplicação de IA na medicina, que é o foco desta dissertação, a IA já mostrou e provou o seu potencial em diversas tarefas, como o diagnóstico, a deteção de lesões e a segmentação de células e órgãos. A aplicação de IA na medicina tem uma motivação extra muito clara e que não está presente nas outras áreas: a possibilidade de melhorar e até mesmo de salvar vidas. Para além desta motivação extra, a aplicação de IA na medicina acaba por ter também os benefícios que se podem encontrar em qualquer outro domínio e que são essencialmente dois: (1) a automatização de processos que são tradicionalmente executados por humanos, que no caso da medicina são os profissionais clínicos, fazendo com que estas tarefas deixem de ser executadas inteiramente e exclusivamente por humanos, como por exemplo a tomada de decisão, e (2) a capacidade de identificar padrões ou mesmo executar tarefas que os humanos são incapazes de executar, ou que são capazes de executar mas com uma margem de erro considerável, como por exemplo a análise de imagens médicas para tentar obter um diagnóstico. Mesmo assim, e apesar de todas as vantagens descritas, o campo da medicina apresenta uma série de complicações e desafios adicionais, não só técnicos mas também éticos, que não se levantam nos restantes campos, o que torna a aplicação de IA na medicina diferente da sua aplicação nos restantes campos, onde a sua aplicação pode ser mais direta. Estes desafios irão ser abordados na seção 2.13. Dentro do paradigma da aplicação de IA à medicina, esta investigação foca-se no diagnóstico, com recurso a métodos de DL, da DA. No contexto da Aprendizagem Automática (ML)[5], indicar se um sujeito pode ou não sofrer de DA é um problema de classificação, e, neste contexto, são normalmente feitas três distinções: se o sujeito é um controlo saudável, se o sujeito está numa fase de DCL ou se o sujeito sofre de DA.

Para o diagnóstico da DA, os investigadores desenvolveram vários CASs. Nos anos 70 a 90 foram desenvolvidos sistemas baseados em regras e modelos supervisionados (Litjens et al., 2017a). Como entrada, estes modelos de ML supervisionados recebiam vetores de características que eram criados manualmente, por exemplo, extraídos manualmente de imagens médicas. Contudo, a extração desses vetores de características necessita de peritos

4 *Magnetic Resonance Imaging*, na terminologia inglesa.
5 *Machine Learning*, na terminologia inglesa.

humanos capazes de realizar esta tarefa e requerem muito tempo, dinheiro e esforço (Islam and Zhang, 2018).

Mais recentemente, um dos ramos de ML, conhecido como DL, tem vindo a alcançar resultados ótimos em diversos domínios, tais como o reconhecimento da fala, a visão por computador, a compreensão da lingua natural, e até mesmo a análise médica. Estes algoritmos de DL diferem dos métodos convencionais de ML pelo facto de exigirem pouco ou nenhum pré-processamento dos dados que lhe são passados como entrada, como por exemplo imagens, e poderem inferir automaticamente uma representação ótima dos dados em bruto, sem exigir uma seleção prévia e manual de caraterísticas, resultando num processo mais objetivo e menos tendencioso. Portanto, os algoritmos de DL são mais adequados para detetar anomalias anatómicas subtis em imagens médicas (Basaia et al., 2019).

A popularidade e evolução de DL levou a um aumento significativo na investigação, resultando em enumeras propostas para enfrentar o problema descrito até então - o diagnóstico da DA. A variedade de técnicas e opções é muito vasta e cada uma tem as suas vantagens e desvantagens. Além disso, sabe-se que os modelos propostos, na grande maioria, são capazes de diagnosticar a DA através de imagens médicas a um nível não inferior a um radiologista experiente, tendo por vezes um desempenho superior (Kloppel et al., 2008). Ou seja, a maioria dos modelos propostos na investigação feita ao longo dos anos tem o potencial de ser implementada em cenários clínicos reais se o seu desempenho for comparado com o humano, mas ainda assim, quando se olha apenas para métricas de avaliação do desempenho dos modelos, ainda há um longo caminho a percorrer até que estes passem a ser parte integrante do diagnóstico clínico. Posto isso, é fundamental identificar quais as técnicas que atingem melhor desempenho, mas também identificar aspetos que podem ser melhorados em trabalhos já realizados anteriormente, mesmo que isso comprometa os resultados finais, desde que estes sejam o mais realistas possíveis. Ou seja, o objetivo não passa apenas por apresentar mais um modelo, pois, como referido, foram já propostos imensos modelos que são até capazes de realizar o diagnóstico de melhor forma que radiologistas, mas também por apresentar os resultados mais realistas possíveis dentro deste contexto, mesmo que isso signifique sacrificar os valores finais das métricas de avaliação dos modelos propostos.

Um outro aspeto que vale a pena referir é o facto da construção destes modelos requerer, em certos casos, um conhecimento bastante profundo do domínio médico aplicacional. Ou seja, há tarefas que podem necessitar de conhecimentos profundos de fisiologia ou anatomia humanas, como por exemplo, para o pré-processamento das imagens médicas é necessário conhecimento profundo do que representa cada imagem, podendo estas ser de diversos tipos, como as imagens de MRI ponderadas em T1 ou imagens de *Positron Emission Tomography* (PET). Isto vai de encontro ao que foi referido anteriormente sobre os modelos desenvolvidos antes do surgimento dos métodos de DL, que exigiam a extração manual das características das imagens, o que implicava um conhecimento dos biomarcadores utilizados

para diagnosticar a DA. Daqui surge um outro objetivo desta dissertação para além do já referido. Ou seja, para além de desenvolver um modelo capaz de realizar o diagnóstico da DA com os resultados mais realistas possíveis, um outro objetivo é analisar até que ponto esta tarefa é viável para um profissional que não possui conhecimentos aprofundados da área médica e, consequentemente, que não possui conhecimentos profundos sobre a DA e sobre os conceitos a ela associados.

Não surpreende que um profissional que constrói modelos de IA ou um cientista de dados não possuam conhecimentos da temática médica, sendo-lhe por isso necessário adquirir os conhecimentos mínimos sobre o domínio aplicacional para poder realizar a tarefa a que se propõe. Contudo, as particularidades da temática médica podem dificultar esta tarefa.

De entre os diversos métodos de DL que têm vindo a ser aplicados com sucesso a dados obtidos por MRI, para diagnosticar a DA, estão as Redes Neuronais Convolucionais (CNNs)[6], que já provaram ser um modelo de DL bastante poderoso e adequado a imagens, tais como as obtidas por MRI. Deste o grande sucesso da `AlexNet` na classificação de imagens (Krizhevsky et al., 2012), as CNNs expandiram-se a diversos campos. Na análise de imagens médicas, as primeiras adoções bem-sucedidas das CNNs foram conseguidas em imagens bidimensionais (2D), tais como imagens do raio-X torácico (Bar et al., 2015) e depois alargadas a imagens tridimensionais (3D), tais como as obtidas por MRI (Oh et al., 2019).

1.2 Objetivos da Dissertação

Vivemos num mundo onde as técnicas de ML já deram provas da sua enorme capacidade para resolver diversos problemas. Na área médica tem sido desenvolvida uma série de métodos de aprendizagem automática supervisionada para auxiliar os profissionais de saúde na tomada de decisão e no tratamento de situações que requerem cuidados complexos (Lee and Mark, 2010). De entre estas técnicas, os métodos de DL são adequados para resolver problemas complexos e provaram ser uma tecnologia de ponta no que toca ao reconhecimento de padrões.

O principal propósito desta dissertação final de mestrado é desenvolver um modelo de DL, ou seja, uma Rede Neuronal Profunda (DNN)[7] capaz de realizar o diagnóstico, se possível de forma precoce, da DA a partir de um único exame MRI estrutural do paciente. Para a realização desta tarefa é necessário escolher, dentro do grande conjunto de possibilidades, o modelo de aprendizagem mais indicado para apoiar a tomada de decisão neste caso concreto. Isto implica, em primeira instância, uma leitura e revisão da literatura seguida da implementação e avaliação dos modelos de DL com um determinado conjunto de dados.

6 *Convolutional Neural Networks*, na terminologia inglesa.
7 *Deep Neural Neetwork*, na terminologia inglesa.

Para alcançar uma solução capaz de realizar de forma satisfatória a tarefa proposta, é necessário atingir um conjunto de metas intermédias:

- Compreender os principais conceitos relacionados com o objetivo da investigação;
- Reconhecer quais são os modelos de DL que estão a proporcionar melhores resultados no diagnóstico da DA, através de uma revisão sistemática da literatura;
- Recolher dados obtidos com exames de MRI estruturais, neste caso o conjunto de dados Iniciativa de Neuroimagem da Doença de Alzheimer (ADNI);
- Tratar e pré-processar os dados recolhidos;
- Implementar, treinar e avaliar os modelos selecionados, com o objetivo de comprovar a exequibilidade da sua utilização por profissionais sem extensos e profundos conhecimentos médicos;
- Identificar o melhor modelo para tratar o problema em análise, comparando e analisando os resultados obtidos;
- Aplicar técnicas que melhorem o desempenho do modelo selecionado;
- Apresentar um conjunto de recomendações relacionadas com o objetivo da dissertação.

1.3 Metodologia de investigação

De forma geral, podemos dizer que o processo de investigação nesta dissertação teve duas fases principais: a fase exploratória e a fase descritiva. A fase exploratória teve como principal objetivo a identificação dos modelos de DL que são mais comuns na realização do diagnóstico da DA. Esta fase incluiu como principais tarefas:

- Revisão sistemática da literatura;
- Identificação dos modelos de DL mais utilizados;
- Identificação das técnicas de pré-processamento mais utilizadas para neuro-imagens, mais concretamente para as imagens de MRI;
- Identificação e estudo das ferramentas utilizadas para elaborar os modelos selecionados, ou seja, a linguagem de programação e as bibliotecas utilizadas para desenvolver os modelos.

Concluída a revisão da literatura, foi possível compreender melhor a aplicação de DL no diagnóstico da DA, mas também ter mais informação sobre quais os modelos mais

adequados para implementar um sistema de diagnostico automático da DA num ambiente real e sobre os métodos de pré-processamento comummente aplicados neste tipo de dados.

Por sua vez, a fase descritiva foi planeada com o objetivo de implementar os diversos modelos, descrever o seu funcionamento e comparar os resultados obtidos. Para isso, foram delineadas as seguintes tarefas intermédias:

- Pré-processar os dados, que consistem num conjunto de imagens de MRI, colocando em prática as técnicas estudadas na fase exploratória, com o objetivo de preparar os dados da melhor forma para treinar os diversos modelos;
- Prototipar os modelos selecionados durante a fase exploratória;
- Treinar os modelos com recurso aos dados pré-processados;
- Avaliar os modelos e comparar os resultados obtidos;
- Elaborar recomendações para profissionais de IA sem conhecimentos profundos da área médica. Com base nas conclusões da atividade anterior, fornecer uma base sobre a qual se podem construir modelos de diagnóstico.

Apesar desta divisão em duas fases distintas, também é possível enquadrar o trabalho desenvolvido numa metodologia popular denominada por *Cross-industry Standard Process for Data Mining* (CRISP-DM).

1.3.1 *CRISP-DM*

O CRISP-DM é composto por 6 etapas que definem um fluxo de trabalho bem estruturado: compreensão do negócio, compreensão dos dados, preparação de dados, modelação, avaliação e implantação (figura 1).

Compreensão do Negócio

Sendo a doença de Alzheimer o tipo de demência mais comum, afetando um elevado número de pessoas com mais de 65 anos (Grundman, 2004), e tendo a si associado um conjunto enorme de consequências negativas para a vida humana que acabam por reduzir a qualidade de vida dos pacientes, a deteção precoce é fundamental. Deste modo, pode-se tentar retardar a progressão da doença com um tratamento adequado que previna danos nos tecidos cerebrais (Islam and Zhang, 2018).

Compreensão dos Dados

Os dados são o elemento mais importante em todo o processo. Neste caso, os dados utilizados foram *scans* 3D de MRI estrutural obtidos do conjunto de dados ADNI. As

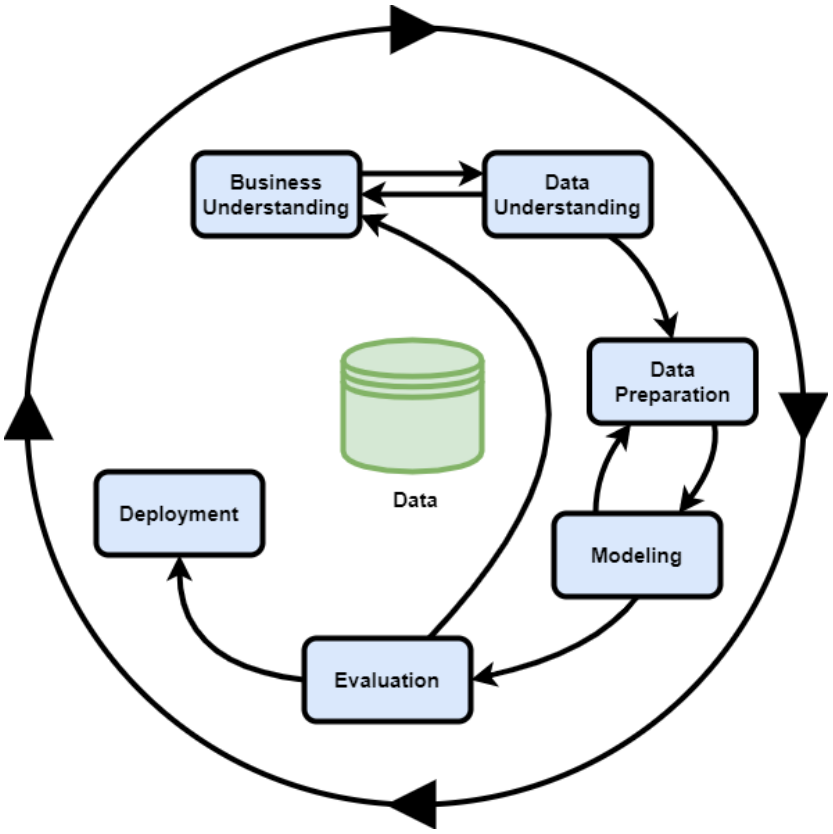


Figura 1: Diagrama de fases da metodologia CRISP-DM.

imagens MRI estruturais desempenham um papel importante na avaliação das alterações anatómicas do cérebro para o diagnóstico da DA (Li and Liu, 2018).

Preparação de Dados

Esta fase do processo de desenvolvimento é dedicada ao tratamento dos dados recolhidos na fase anterior. Esta preparação é fundamental e tem bastante influência no treino dos modelos de ML, e no caso desta dissertação, nos modelos de DL. O tratamento dos valores nulos, a eliminação de dados com "ruído", o tratamento de valores em falta, o tratamento de valores discrepantes[8] e a normalização, são tarefas que se enquadram nesta etapa.

Uma vez que estamos perante imagens médicas 3D, é possível aplicar diversas técnicas de pré-processamento, dependendo das características que quisermos extrair e da forma como queremos apresentar os dados. Contudo, neste contexto é comum seguir uma de três abordagens: orientada ao *voxel*, baseada na zona de interesse, e orientada ao *patch*. Estas técnicas serão apresentadas em maior detalhe na seção 2.8.

8 *Outliers*, na terminologia inglesa.

Modelação

A etapa de modelação contempla todo o processo de treino, avaliação e afinação dos diversos modelos, utilizando os dados pré-processados na fase anterior, com recurso às ferramentas estudadas na fase exploratória.

Avaliação

A aplicação de diferentes algoritmos ML requer um processo que assegure que os resultados são fiáveis e estatisticamente significativos. Consequentemente, têm de ser aplicadas métricas de desempenho para assegurar a avaliação da qualidade e das características dos modelos, garantindo a fiabilidade dos resultados. Neste caso, as métricas são medidas numéricas que avaliam o desempenho dos diversos modelos na tarefa de classificação, permitindo assim decidir qual é o melhor modelo. As métricas de avaliação de desempenho serão apresentadas na seção 2.14.

Implantação

Embora não fosse um objetivo desta dissertação, a etapa de implantação[9] consistiria em integrar o modelo anteriormente escolhido nos sistemas de apoio à decisão médica, auxiliando os profissionais de saúde na análise de exames de MRI e, consequentemente, no diagnóstico da DA.

1.4 Âmbito

É importante referir que esta dissertação foge um pouco à regra no que toca a investigações realizadas nesta área no sentido em que não procura alterar as soluções propostas no estado da arte, ou até procurar uma solução que seja melhor. Em vez disso, é pretendido que se identifiquem quais os modelos que estão a ter mais importância e fazer uma implementação direta dos mesmos, com recurso às bibliotecas disponíveis e tipicamente utilizadas por profissionais dedicados à implementação de modelos de ML. Com isto, pretende-se determinar se estas ferramentas são suficientes para criar bons modelos, como os propostos no estado da arte, sem necessidade de ter conhecimentos médicos profundos. Para além disso, procura-se identificar aspetos que podem ser melhorados em todo o processo de recolha e pré-processamento dos dados, do treino e da avaliação dos modelos.

Tendo em conta o que foi dito, fica claro que não se pretende adotar uma abordagem meramente clínica e, nesse sentido, são diversos os conceitos clínicos que não serão aprofundados.

9 *Deployment*, na terminologia inglesa.

Os biomarcadores específicos para o diagnóstico da DA são um dos exemplos de conceito que não será explorado profundamente. Isto não é um problema que impeça qualquer atividade proposta, e acaba por ser propositado, pois como já foi referido, a DL tem como uma das principais características a vantagem dos modelos serem capazes de identificar os biomarcadores de forma autónoma, contrastando com os modelos propostos antes do surgimento da DL, onde era necessária a especificação manual das características relevantes - como por exemplo os biomarcadores - a extrair das imagens.

Ainda relativamente aos termos clínicos, não se pretendia estudar em detalhe todos os conceitos relacionados com imagiologia médica dada a sua imensidão. Alguns exemplos são PET, MRI ponderada em T1 e MRI ponderada em T2. Vale apenas tomar nota que as imagens de MRI não são típicas imagens bidimensionais, mas que se tratam de imagens tridimensionais, geralmente de grande tamanho.

No que toca ao pré-processamento das imagens médicas, cada investigação adota as técnicas de pré-processamento que permitem adaptar as imagens aos modelos a desenvolver. Neste trabalho aplicaram-se técnicas de pré-processamento normalizadas para as imagens MRI.

Por fim, algumas investigações utilizam, para além das imagens médicas, uma combinação das mesmas com outros dados clínicos e demográficos dos pacientes. Nesta investigação serão usadas exclusivamente as imagens médicas por forma a obter resultados que indiquem de forma clara e verdadeira de que forma as imagens médicas por si só conseguem indicar o estado clínico dos pacientes.

1.5 Estrutura da dissertação

Esta seção sumaria como está organizado o resto do documento. Conforme foi dito na seção 1.3, existe uma fase exploratória nesta dissertação que consiste, essencialmente, na revisão sistemática da bibliografia do diagnóstico da DA, e esta fase encontra-se descrita no capítulo 2 deste documento. De forma mais detalhada este capítulo conta com:

- Na seção 2.1 é feita uma introdução sobre a DA, as suas características, as causas e os biomarcadores. É também aqui que se descrevem as imagens de MRI e os CASs.
- Em 2.2 é descrita a IA e em 2.3 o papel da IA na medicina.
- Passando para as seções 2.4 e 2.5, estas descrevem o que é ML e DL, respetivamente.
- A seção 2.6 inclui um resumo das CNNs, bem como a sua inspiração na natureza e a descrição geral da sua arquitetura.
- Na seção 2.7 são mencionadas as principais arquiteturas de CNNs bem como as inovações que cada uma trouxe à área da informáticas médica.

- Em 2.8 faz-se uma análise das principais técnicas de pré-processamento de imagens de MRI, em especial no contexto do estudo da DA.
- Nas seções 2.9, 2.10, 2.11 e 2.12 são detalhadas algumas das técnicas que permitem melhorar o desempenho das redes neuronais. Concretamente, aborda-se a transferência de aprendizagem, *batch normalization*, aplicação de *dropout* e aumento de dados.
- Em 2.13 são referidos os principais desafios e limitações do trabalho realizado. São aqui abordados problemas como o tamanho do conjunto de dados, problemas éticos relacionados com o tema investigado e também limitações de tempo e de poder de cálculo.
- Na seção 2.14 são elencadas as métricas de avaliação de desempenho comummente utilizadas.
- Por fim, a seção 2.15 resume os trabalhos relacionados com a investigação desenvolvida nesta dissertação.

No capítulo 3 começa a fase descritiva da dissertação e conta com:

- Na seção 3.1 é descrita a obtenção dos dados da base de dados ADNI.
- A seção 3.2 descreve os métodos de pré-processamento dos dados levados a cabo nesta investigação.

Seguidamente encontra-se o capítulo 4, onde se dá continuidade à fase descritiva do trabalho, nomeadamente:

- Na seção 4.1 são enunciadas, de forma resumida, as principais tecnologias usadas na investigação.
- As seções 4.2 e 4.3 descrevem, respetivamente, a implementação de CNNs 2D e de CNNs 3D, o treino das mesmas, bem como os resultados obtidos.
- Na seção 4.4 é descrito o processo de aumento de dados levado a cabo.
- Na seção 4.5 é feita uma análise dos resultados obtidos.
- Por fim, na seção 4.6 é feito um conjunto de recomendações úteis para a realização de uma investigação semelhante, tendo por base a experiência adquirida na fase descritiva.

Finalmente, o capítulo 5 encerra este documento com as conclusões do trabalho realizado e apresenta um resumo sobre as linhas de trabalho futuro.

2

ESTADO DA ARTE

Antes de avançar para o pré-processamento dos dados ou até mesmo para o desenvolvimento dos diversos modelos de DL e para uma melhor compreensão do tema, é apresentado um resumo do estado da arte no contexto do diagnóstico automático da DA.

2.1 A DOENÇA DE ALZHEIMER

A DA é uma doença neurodegenerativa incurável e progressiva (Islam and Zhang, 2018). É tipo de demência mais predominante (Qiu et al., 2020; Islam and Zhang, 2018; Oh et al., 2019; Li and Liu, 2018), afetando 1 em cada 9 pessoas com mais de 65 anos (Lu et al., 2018) e estima-se que até 2050, cerca de 0,64 biliões de pessoas serão diagnosticadas com a DA (Islam and Zhang, 2018). Esta doença envolve uma deficiência cognitiva progressiva, geralmente associada a uma perda precoce da memória que, em fases avançadas, pode levar os pacientes a não conseguirem fazer as suas atividades diárias sem assistência (Lu et al., 2018).

Existem 3 principais fatores de risco na DA: (1) envelhecimento, (2) histórico familiar da DA e (3) ser portador de certos genes (B.R et al., 2019). No corpo humano o cérebro é o órgão mais importante e complexo do sistema nervoso e, por isso, quaisquer problemas ou anomalias no cérebro podem afetar gravemente as funcionalidades regulares de todo o corpo humano (B.R et al., 2019). Esta doença tem bastante impacto na vida das pessoas que dela sofrem, pois destrói células cerebrais levando à perda de memória, de funções mentais e da capacidade de realizar as atividades diárias. Inicialmente, a DA afeta a parte do cérebro que controla a linguagem e a memória, resultando na confusão e na dificuldade em falar, ler ou escrever. Numa fase mais avançada, os doentes esquecem-se frequentemente da sua vida, podem não reconhecer os membros da sua família e têm dificuldade em realizar atividades diárias, como escovar o cabelo e os dentes. Tudo isto torna os doentes de DA ansiosos ou agressivos ou deambulantes. Em última análise, a DA destrói a parte do cérebro que controla a respiração e a funcionalidade do coração, levando à morte (Islam and Zhang, 2018).

Ao nível do cérebro as principais alterações que ocorrem são o encolhimento do hipocampo e do córtex cerebral e a ampliação dos ventrículos (Sarraf et al., 2016). O hipocampo é a parte

do cérebro responsável pela memória episódica e espacial e funciona também como uma estrutura de transmissão entre o nosso corpo e o cérebro. A redução do hipocampo causa perdas e danos celulares, mais especificamente nas sinapses e nos terminais dos neurónios. Assim, os neurónios já não podem comunicar através de sinapses. Consequentemente, as regiões cerebrais relacionadas com a memória, pensamento, planeamento e julgamento são afetadas (Islam and Zhang, 2018).

Por ser uma doença tão incapacitante e impactante, tanto para os doentes como para os que os rodeiam, é fundamental uma deteção prematura da DA para que sejam aplicados os tratamentos necessários. É de notar que os novos tratamentos médicos podem suprimir esta doença neurodegenerativa através de intervenção precoce (Duraisamy et al., 2018). É esta deteção prematura e precisa da doença, especialmente a identificação do risco de progressão de DCL para DA, que dá aos doentes a possibilidade de tratamento (Roberson and Mucke, 2006). Infelizmente, a identificação precoce dos doentes que terão um diagnóstico final de DA é complicado (Ding et al., 2019). Como foi referido, o diagnóstico prematuro da DA está principalmente associado à deteção do DCL, uma fase prodrómica da DA. Embora as queixas e défices de memória do DCL não afetem notavelmente as atividades diárias dos pacientes, foi relatado que o DCL tem um elevado risco de progressão para a DA ou outras formas de demência (Gauthier et al., 2006).

A deteção de alterações físicas no cérebro tem vindo a complementar as avaliações clínicas e tem um papel cada vez mais importante na deteção precoce da DA. Os investigadores têm dedicado os seus esforços a técnicas de neuro-imagem para medir as alterações cerebrais patológicas relacionadas com a doença. Foram desenvolvidas técnicas de ML para construir classificadores, utilizando imagens e medidas clínicas para diagnosticar a DA (Islam and Zhang, 2018).

De forma resumida, para retardar a progressão da demência o tratamento atempado é crucial, o que requer o diagnóstico precoce da DA e da sua fase prodrómica, o DCL. Para alcançar este objetivo é necessário um diagnóstico fiável, a partir da imagiologia cerebral, e um sistema robusto de diagnóstico auxiliado pela análise dos dados de neuroimagem (Oh et al., 2019).

2.1.1 *MRI*

As imagens médicas constituem um aspeto fundamental no processo de diagnóstico e tratamento de diversas doenças mas, para além disso, também são uma fonte muito útil de onde se podem extrair biomarcadores de variadas doenças (O'Connor et al., 2017). Os biomarcadores são provas clínicas objetivas, que podem ser medidas experimentalmente e que representam indiscutivelmente a "evidência clínica mais objetiva e quantificável que um laboratório científico moderno nos permite ter de forma reprodutível" (Strimbu and Tavel,

2010). Existem diversos tipos de biomarcadores, tais como, fisiológicos (funções de órgãos), físicos (alterações de características em estruturas biológicas), histológicos (amostras de tecido obtidas por biopsia) e anatómicos. Podem também ser células específicas, moléculas, genes, enzimas ou hormonas. Ou seja, os biomarcadores são também indicadores de outros processos, não necessária e exclusivamente de doença, mas neste trabalho vamos focar-nos apenas nestes.

Nas últimas duas décadas a MRI tem sido amplamente adotada para estudar o cérebro humano e as doenças que afetam o tecido cerebral (Adel et al., 2017). A análise de imagens MRI é uma prática comum para o diagnóstico da DA na investigação clínica (Islam and Zhang, 2018). As imagens MRI, incluindo as imagens de MRI estrutural (sMRI) e as imagens de MRI funcional (fMRI), são ferramentas de imagem não invasivas e poderosas para ajudar a compreender e a avaliar as alterações neuronais, anatómicas e funcionais, relacionadas com a DA (Liu et al., 2014a; McKhann et al., 2011; Herrup, 2011).

À medida que a DA progride vão sendo acumuladas proteínas anormais no cérebro dos pacientes, mais concretamente a *amyloid-β [Aβ]* e *hyperphosphorylated tau*. A acumulação anormal destas proteínas leva a danos progressivos a nível das sinapses, neurónios e axónios. As alterações no cérebro devidas à DA têm um padrão estereotipado de envolvimento precoce do lobo temporal medial (córtex entorhinal e hipocampo), seguido de dano neocortical progressivo (Qiu et al., 2020). Estas alterações ocorrem anos antes do aparecimento dos sintomas da DA. Parece que os efeitos tóxicos do *hyperphosphorylated tau* e/ou da *amyloid-β [Aβ]*, que gradualmente corroem o cérebro, quando um limiar clínico é ultrapassado começam a desenvolver sintomas amnésicos. As imagens de MRI estrutural podem ser utilizadas para medir estas alterações progressivas no cérebro.

Ultimamente tem havido um interesse crescente no processamento e análise automáticos de imagens médicas, o que traz consigo uma série de vantagens, como as mencionadas na secção 1.1. De facto, espera-se que a IA venha a desempenhar um papel muito importante na análise dos Expedientes Clínicos Electrónicos (ECEs) em geral, e não apenas nas imagens médicas (Ker et al., 2018).

2.1.2 *CAS*

Nos últimos anos foram feitos grandes esforços para desenvolver um CAS, utilizando vários métodos de ML para descodificar os diversos estágios da DA com imagens MRI (Herrup, 2011; McKhann et al., 2011; Zhang et al., 2011; Liu et al., 2013; Suk et al., 2013). Para além da deteção de DA, tem havido um esforço substancial para aplicar métodos de ML e DL em muitas doenças e tipos de imagem, como por exemplo a deteção do cancro da mama com mamografia, a deteção de nódulos pulmonares com tomografia computadorizada e a

classificação da osteoartrite da anca com radiografia. Contudo, a integração destes métodos no fluxo clínico ainda não foi desenvolvida e validada (Ding et al., 2019).

Estão publicados vários trabalhos que visavam classificar os pacientes com DA e DCL com métodos computacionais, empregando vários métodos de ML sobre imagens MRI. O método mais popular entre esses métodos é a *Support Vector Machine* (SVM). A SVM utiliza características informativas e de alta dimensão da MRI, para construir modelos de classificação preditiva que facilitam a automatização do diagnóstico clínico (Basaia et al., 2019). No entanto tendem a utilizar características manuais devido à sua incapacidade de extrair características (Oh et al., 2019). Ou seja, os estudos de ML realizados utilizando dados de neuro-imagem para o desenvolvimento de ferramentas de diagnóstico ajudaram muito na segmentação e classificação automática das imagens MRI cerebral, mas apresentam o problema de necessitarem da geração e extração manual de características a partir dos dados das imagens de MRIs. Isto implica a necessidade de peritos humanos no processo (Islam and Zhang, 2018). Embora muitos estudos tenham aplicado recentemente métodos de ML para o Diagnóstico Assistido por Computador (CAD) da DA, foi demonstrado um estrangulamento do desempenho do diagnóstico na maioria das investigações existentes, principalmente devido às limitações dos modelos de aprendizagem escolhidos (Liu et al., 2014b).

Recentemente, técnicas avançadas de DL têm demonstrado com sucesso o desempenho a nível humano em numerosos campos, incluindo a análise da imagem médica (Islam and Zhang, 2018). Com os modelos de DL podemos extrair características diretamente das imagens sem o envolvimento de peritos humanos. Assim, os investigadores podem focar-se no desenvolvimento de modelos de DL para o diagnóstico automático e preciso de doenças. As tecnologias de DL alcançaram grande sucesso em diversas tarefas de análise de imagens médicas, tais como a MRI, a microscopia, a TAC, o ultra-som, o raio-X e a mamografia. Modelos de DL mostraram também resultados proeminentes na segmentação de órgãos e de subestruturas, na deteção e classificação de várias doenças em áreas de patologia, cérebro, pulmão, abdómen, coração, mama, osso, ou retina (Litjens et al., 2017a). Os modelos de DL, especialmente as CNNs, têm demonstrado um excelente desempenho no campo da imagem médica, ou seja, na realização de tarefas como segmentação, deteção, registo e classificação (Litjens et al., 2017a). Para dados de neuro-imagem, os modelos de DL podem descobrir a representação latente ou oculta e capturar eficazmente as patologias relacionadas com as doenças. Assim, os investigadores começaram a aplicar modelos de DL no diagnóstico da DA e de outras doenças cerebrais (Islam and Zhang, 2018).

Apesar dos resultados promissores, por várias razões estes modelos ainda não conseguiram a plena integração na prática clínica. Primeiro, porque existe uma falta de validação externa dos algoritmos de DL, uma vez que a maioria dos modelos são treinados e testados num único corte. Segundo, porque existe uma noção crescente na comunidade biomédica de que os modelos de DL funcionam como "caixas negras" de difícil compreensão (Castelvecchi,

2016). Por outras palavras, embora os modelos de DL demonstrem um poder de classificação com alta precisão, num espectro alargado de doenças, não explicam com tomaram as decisões de diagnóstico subjacentes, nem indicam quais as características de entrada associadas com as previsões geradas. Finalmente, dado o início incerto e a heterogeneidade dos sintomas observados na DA, uma caracterização individual computorizada da DA continua por resolver. A superação destes desafios é crucial, não só para aproveitar o potencial dos algoritmos de DL e melhorar os cuidados aos pacientes, mas também para preparar o caminho para uma DL baseada em provas explicáveis na comunidade da imagiologia médica (Qiu et al., 2020).

2.2 INTELIGÊNCIA ARTIFICIAL

A história da IA está repleta de mitos. Desde o mundo antigo que entoam os contos de seres mecânicos e inteligentes, com personalidade e com capacidades extraordinárias. No final do século XIX, o autor italiano Carlo Collodi apresentou-nos o bem conhecido Pinóquio, o fantoche de madeira que ganha vida e sonha ser um menino real. Desde a sua criação que o Pinóquio luta para ser aceite pela sociedade. Enquanto que o filme de animação desta história, por parte de Walt Disney em 1940, teve um final feliz, muitos aspetos da versão original de Collodi antecipam os medos contemporâneos sobre a IA.

Menos de uma década após quebrar o código de encriptação Nazi e ajudar as forças aliadas a vencer a Segunda Guerra Mundial, Alan Turing mudou a história pela segunda vez. Desta vez colocou apenas a seguinte e simples questão: "Podem as máquinas pensar?". Turing (1995) com o seu documento denominado de "*Computing Machinery and Intelligence*" e o posterior Teste de Turing, que permitem determinar se uma máquina é inteligente, estabeleceu o objetivo fundamental e a visão da IA.

O termo "Inteligência Artificial"é cunhado no "*Dartmouth Summer Research Project on Artificial Intelligence*", no Dartmouth College em 1956. Liderada por John McCarthy, um professor de 28 anos, a conferência que definiu o âmbito e os objetivos da IA é amplamente considerada como o nascimento da IA. O considerado "pai" da IA, John McCarthy, define este ramo como "a ciência e engenharia de criar máquinas inteligentes, mais especificamente programas de computador inteligentes".

No seu cerne a IA é o ramo da informática que visa responder afirmativamente à pergunta de Turing, ou seja é o esforço de replicar ou simular a inteligência humana em máquinas. No entanto, o principal objetivo da IA tem originado muitos debates e questões, e ainda hoje nenhuma definição singular deste ramo da ciência é universalmente aceite. O problema em definir IA como sendo simplesmente "construção de máquinas inteligentes"é que esta definição não explica o que é realmente a IA, nem o que torna uma máquina inteligente. No seu revolucionário e didático livro "*Artificial Intelligence: A Modern Approach*", com primeira

edição em 1994, os autores Stuart Russell e Peter Norvig definem IA como "o estudo dos agentes que recebem perceções do ambiente e executam ações". Continuando o seu trabalho, estes investigadores exploram quatro abordagens que historicamente definiram o campo da IA. Estas abordagens assentam na capacidade de: (1) pensar humanamente, (2) pensar racionalmente, (3) agir humanamente e (4) agir racionalmente.

Mais recentemente, na *Japan AI Experience* em 2017, Jeremy Achin, CEO da DataRobot, definiu IA, num dos seus discursos, como sendo um "sistema informático capaz de executar tarefas que normalmente requerem inteligência humana... Muitos destes sistemas de inteligência artificial são alimentados pela ML, alguns deles são alimentados por DL e alguns deles são alimentados por coisas muito aborrecidas como regras".

A IA pode ser dividida em duas subcategorias: *Narrow AI*, também conhecida como "IA fraca"e *Artificial General Intelligence* (AGI), por vezes também referida como "IA forte". A *Narrow AI* opera num contexto limitado e é apenas uma simulação da inteligência humana. Normalmente é focada na realização de uma única tarefa extremamente bem sucedida. Este tipo de IA é a forma mais bem sucedida até ao momento. Como exemplos desta subcategoria temos a *Alexa*, a *Siri* e outros assistentes pessoais, as pesquisas no Google, os automóveis com condução automática e o reconhecimento facial. Grande parte da IA fraca é criada recorrendo a algoritmos de ML e DL. A *Artificial General Intelligence* é o tipo de inteligência que vemos nos filmes, como os robôs de *Westworld* ou o *The Terminator*. A AGI é uma máquina com inteligência geral e, tal como um ser humano, pode aplicar essa inteligência para resolver qualquer problema. A procura de um "algoritmo universal para aprender e agir em qualquer ambiente"não é nova, mas o tempo passado até então não diminuiu a dificuldade de criar uma máquina tão complicada, complexa e com um conjunto completo de capacidades cognitivas (Russell and Norvig, 2009). A preocupação em que este tipo de inteligência se sobreponha à humanidade é algo que, segundo os especialistas, não tenhamos com que nos preocupar tão cedo.

Nas últimas décadas tem havido algumas demonstrações da superioridade da IA sobre os meros mortais. Em 1997, o computador *Deep Blue* da IBM tornou-se o primeiro supercomputador a derrotar um campeão mundial de xadrez, neste caso Garry Kasparov. Outro marco foi alcançado em 2011, quando um sistema informático chamado Watson ganhou 1 milhão de dólares no programa de jogos de televisão dos EUA "Jeopardy". Mais tarde, em 2015, a tecnologia *AlphaGo* da Google venceu Fan Hui, o melhor jogador humano da Europa, no antigo jogo de tabuleiro chinês Go. No entanto, as coisas nem sempre correram assim tão bem. Por exemplo, em 2016 um robô realista chamado Sophia declarou que iria "destruir os humanos", durante uma demonstração na conferência *South by Southwest*.

Nos dias de hoje, desde a ajuda na luta global contra o Covid-19, a implementação de cidades inteligentes e a condução de automóveis, a IA está rapidamente a remodelar o mundo em que vivemos. Mas nem todos estão à vontade com esta nova realidade. O

bilionário empreendedor tecnológico Elon Musk referiu-se à IA como a "maior ameaça existencial"do nosso tempo. Mesmo não sendo bem aceite por todos, o interesse pela IA, guiado pelo grande avanço tecnológico em visão por computador no final dos anos 2000, pelo aumento exponencial da capacidade de processamento das máquinas, pelo aumento dos dados disponíveis no início dos anos 2010 (*Big Data*), bem como os avanços na DL por parte das empresas de tecnologia de topo (Adam Gibson, 2017) tem aumentado, pelo que nos próximos anos são esperados ainda mais avanços nesta área.

2.3 O papel da inteligência artificial no caso de estudo

Não é de agora a aplicação de sistemas de IA no domínio da medicina e a sua aplicação, no que toca ao processamento de ECEs, tem aumentado consideravelmente e isso deve-se essencialmente a 3 fatores:

- A ascensão da IA, mais especificamente da DL. Não teria sido viável trazer a IA para a área da medicina, pelo menos para a realização de tarefas tão complexas, se estes sistemas não tivessem mostrado um desempenho tão notável nas ultimas décadas.
- O crescimento, em número e em tamanho, dos dados registados. Aos poucos a área da medicina está a aproximar-se do *Big Data* e, tal como aconteceu na maioria das áreas onde este paradigma se notabilizou, a IA tem-se revelado uma ferramenta fundamental para gerir a informação de forma eficiente (Razzak et al., 2017). Pelo contrário, está a tornar-se cada vez mais difícil para os humanos lidarem com tanta informação.
- O erro humano. Como qualquer profissional, o desempenho dum radiologista é limitado, por aspetos como a experiência, e por isso pode cometer erros nas suas decisões (Ker et al., 2018; Kloppel et al., 2008). De acordo com Matsuda (2007), mesmo os mais experientes falham entre 10% e 15% do tempo.

Nos últimos anos a investigação sobre análise de imagens médicas tem-se concentrado nas arquiteturas CNN (Ker et al., 2018; Litjens et al., 2017b). O objetivo da utilização destes modelos é bastante diverso, mas aqui vamos realçar apenas dois exemplos de aplicação na análise de imagens médicas: Deteção Assistida por Computador (CADe)[1] e Diagnóstico Assistido por Computador (CADx)[2]. A deteção assistida por computador consiste em identificar os diversos elementos numa imagem, tais como células ou órgãos. Uma exemplo de aplicação pode ser a identificação de áreas de interesse para um médico, tais como lesões. Este conceito é muitas vezes confundido com a segmentação, mas eles são no fundo coisas diferentes. Enquanto a deteção consiste em identificar uma caixa onde se localiza

1 *Computer-aided Detection*, na terminologia inglesa
2 *Computer-aided Diagnosis*, na terminologia inglesa

cada elementos da imagem, a segmentação implica identificar o local de cada elemento e identificar (classificar) os pixeis que lhe pertencem.

O diagnóstico assistido por computador, que é uma das principais aplicações das CNNs e que também pode ser denominado apenas de classificação, consiste em efetuar um diagnóstico automático a partir da informação disponibilizada. Ou seja, a partir de um conjunto de dados introduzidos, o objetivo é classificá-los numa de várias classes. Essa classificação pode ser binária, como por exemplo entre doente ou não doente, mas existem também muitos casos com mais do que duas classes. O problema a tratar nesta dissertação encaixa neste último caso, ou seja, um CADx da DA para classificação numa de 3 classes: controlo normal, doente com DA e DCL. A figura 2 mostra a diferença entre os três tipos de aplicação mencionados.

Figura 2: CADx, CADe e segmentação.

2.4 APRENDIZAGEM AUTOMÁTICA

O interesse por ML explodiu ao longo da última década. Diariamente esta área das ciências da computação é discutida e abordada em programas de informática, conferências, ou jornais. O artigo "*A Fast Learning Algorithm for Deep Belief Nets*" (Hinton et al., 2006) mostra como treinar uma DNN capaz de reconhecer dígitos escritos à mão com uma precisão superior a 98%, utilizando uma técnica denominada pelos autores de "Aprendizagem Profunda". O treino de uma rede neuronal era visto como algo impossível, e isto fez com que grande parte dos investigadores tenha abandonado este tema desde os anos 90 (Géron, 2019). O artigo em causa reavivou o interesse da comunidade científica e, pouco tempo depois, novos trabalhos mostraram que a DL não só era possível, como era também capaz de fazer, com auxílio de grande poder computacional e grandes quantidades de dados, o que nenhum outro algoritmo de ML consegue. Esta euforia rapidamente se alargou às diversas áreas de ML e conquistou a indústria, sendo responsável por grande parte da magia dos produtos tecnológicos das mais diversas áreas: reconhecimento de voz, recomendação de produtos, reconhecimento facial, aspiradores inteligentes, entre muitas outras aplicações.

Fundamentalmente a ML usa algoritmos para extrair informação dos dados e para os representar através dum modelo. Esse modelo é depois usado para inferir coisas sobre dados ainda não modelados (Adam Gibson, 2017). Em 1959 Arthur Samuel afirma que a ML é a área de estudo que dá aos computadores a capacidade de aprender sem serem explicitamente programados. Em 1997 Tom Mitchell dá uma definição mais orientada para a engenharia, indicando que estamos perante ML quando um programa de computador aprende com a experimentação `E` na realização duma tarefa `T` e com uma medida de desempenho `P`, se o seu desempenho a realizar `T`, medido por `P`, melhorar com a experimentação `E`.

Nós estamos conscientes do significado do termo "aprendizagem", por exemplo melhorar numa tarefa física (como jogar futebol ou conduzir um carro) ou intelectual (como aprender a jogar um vídeo jogo ou aprender uma nova língua), ao longo de um período de tempo. ML foca-se no desenvolvimento de algoritmos que podem aprender como os seres humanos, ou seja, que ficam melhores na realização de numa tarefa ao longo de um período de tempo, através da experiência (Ketkar, 2017).

2.4.1 *Tipos de abordagens de ML*

Existem diversos tipos de abordagens de ML que são classificados em distintas categorias, tendo em conta um conjunto de critérios:

- Se são ou não treinados com supervisão humana (aprendizagem supervisionada, não supervisionada, semi supervisionada e por reforço)
- Se podem ou não aprender de forma incremental (*online* versus *batch learning*)
- Se funcionam simplesmente comparando as novas amostras com as amostras conhecidas, ou, em vez disso, detetam padrões nos dados usados para treino e constroem modelos preditivos (baseados em instâncias versus baseados em modelos)

Estes critérios não são exclusivos, podendo ser combinados da forma mais adequada ao problema a resolver.

Como os sistemas de ML podem ser classificados dependendo se são ou não treinados com supervisão humana, ou seja, de acordo com o tipo de dados utilizados. Nesta classificação consideram-se 4 categorias: aprendizagem supervisionada, não supervisionada, semi supervisionada e por reforço. Será feita apenas uma pequena introdução ao que é aprendizagem supervisionada, pois é a variante que interessa mais a esta dissertação.

Aprendizagem supervisionada

Em aprendizagem supervisionada os dados de treino que alimentam os modelos incluem a solução desejada, denominada de etiqueta ou alvo. Este é o tipo de abordagem que interessa

nesta investigação, pois estamos perante um conjunto de dados cujas instâncias se encontram etiquetadas. Uma tarefa que tipicamente se enquadra na aprendizagem supervisionada é a classificação (figura 3), onde cada instância dos dados de treino pertence a uma classe e o modelo prevê a que classe pertence uma nova instância desconhecida. Por exemplo, prever se uma mensagem é não solicitado ou se é regular, ou no caso desta dissertação prever se um paciente é saudável, se encontra num estado de DCL ou se possui a DA.

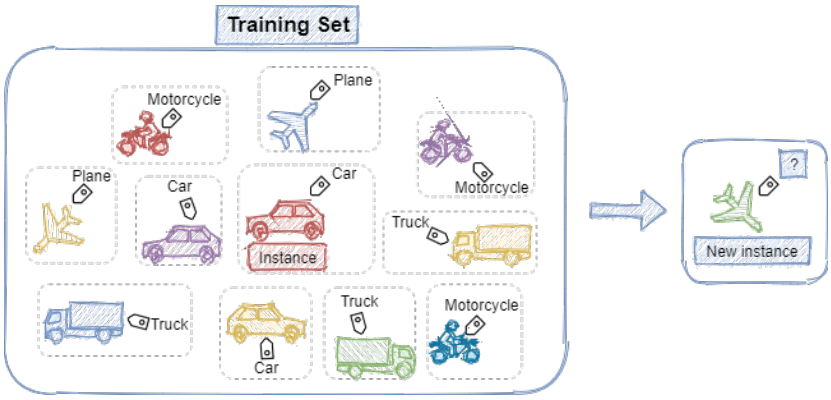


Figura 3: Conjunto de dados de treino com etiquetas para aprendizagem supervisionada.

Uma outra tarefa comum deste tipo de sistema é a regressão (figura 4), que consiste na previsão de um alvo do tipo numérico, como por exemplo a previsão do preço de um carro dando um conjunto de parâmetros (cor, potência, tipo de combustível, etc.), denominados de *predictors*. Para treinar um sistema deste tipo é necessário fornecer-lhe exemplos de carros, incluindo os seus *predictors* e as suas etiquetas, neste caso os preços.

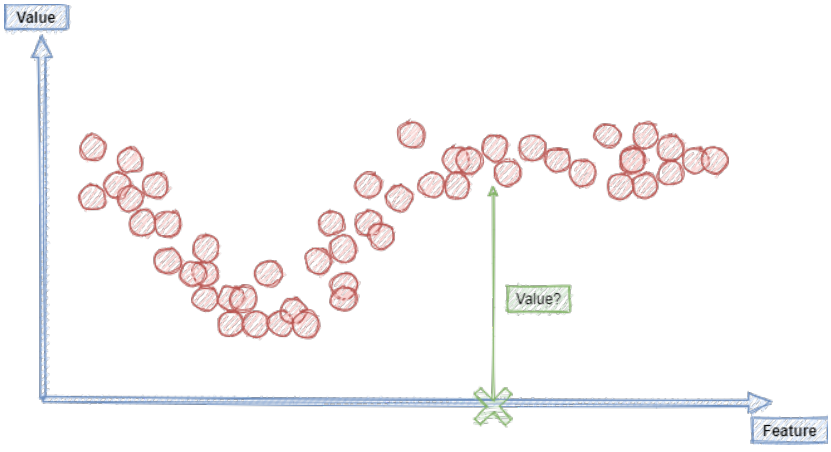


Figura 4: Regressão.

Alguns dos algoritmos de aprendizagem supervisionada mais importantes são a regressão linear, a regressão logística, a máquina de vetores de suporte, *k-Nearest Neighbors*, árvore de decisão, floresta aleatória e rede neuronal artificial.

2.5 APRENDIZAGEM PROFUNDA

Foram diversas as invenções humanas que se inspiraram na natureza. Como as aves foram inspiração para a criação dos aviões, faz sentido olhar para o cérebro como inspiração para construir máquinas inteligentes. Foi esta ideia chave que inspirou as Redes Neuronais Artificiais (ANNs). O cérebro é o órgão mais incrível do corpo humano na medida em que é ele que controla todos os nossos sentidos, permite guardar memórias, experimentar emoções e até sonhar. Sem ele seríamos organismos primitivos incapazes de elaborar as mais complexas tarefas. Intrinsecamente, é o cérebro que nos torna inteligentes.

Apesar de os aviões serem inspirados nas aves, não têm que bater as asas. Da mesma forma, as ANNs que são o conceito chave da DL são diferentes da sua inspiração biológica. As redes neuronais biológicas (cérebro) são compostas por cerca de 86 mil milhões de neurónios e os investigadores estimam que existem mais de 500 triliões de ligações entre neurónios no cérebro humano. Mesmo as maiores ANNs atuais nem sequer se aproximam deste número (Adam Gibson, 2017).

As ANNs estão na génese da DL e, por serem escaláveis, poderosas e versáteis, são ideais para tentar resolver problemas de ML altamente complexos (Géron, 2019). Como por exemplo, classificar biliões de imagens da Web, potenciar serviços de reconhecimento de voz, ou recomendar os melhores vídeos para ver aos utilizadores duma plataforma como o YouTube.

Surpreendentemente, as ANNs já existem há bastante tempo e foram apresentadas pela primeira vez pelo neuro-fisiologista Warren McCulloch e pelo matemático Walter Pitts, em 1943, através de um modelo computacional simplificado, naquela que foi a primeira arquitetura de uma rede neuronal (McCulloch and Pitts, 1943). O facto de não terem produzido resultados imediatos, de terem surgido outras técnicas de ML que ofereciam melhores resultados e terem fundamentos teóricos mais sólidos, como é o caso das máquinas de vetores de suporte, levaram a que este método de aprendizagem fosse esquecido e pouco desenvolvido durante muitos anos.

Atualmente estamos perante uma onda de elevado interesse pelas ANNs e consequentemente pela área de DL, mas desta vez há boas razões para acreditar que este entusiasmo não desapareça e que elas venham a ter um grande impacto nas nossas vidas. Para além da enorme quantidade de dados disponíveis e do elevado poder computacional para treinar as ANNs com dimensões grandes num tempo razoável, as ANNs frequentemente ultrapassam os tradicionais métodos de ML em tarefas complexas (Géron, 2019) como a análise de imagens médicas.

Dentro dos algoritmos de DL há uma variedade enorme de métodos e arquiteturas. No que toca a visão por computador são as CNNs que se destacam, mas justifica-se referir que outras arquiteturas e métodos, nomeadamente métodos de aprendizagem não supervisionada,

também são fundamentais para certo tipo de problemas. Por exemplo, os modelos inspirados nos *autoencoders* são adequados para realizar tarefas como extrair características dos dados, gerar novos dados ou restaurar dados danificados. Na seção 2.6 será feita uma descrição mais detalhada das CNN, pois são o tipo de modelo mais relevante o presente trabalho.

2.6 REDES NEURONAIS CONVOLUCIONAIS

São diversos os tipos de modelos que foram importantes nos últimos anos na análise de imagens médicas, desde *autoencoders* (Gupta et al., 2013; Payan and Montana, 2015), *stacked autoencoders* (Suk and Shen, 2013; Suk et al., 2013), sistemas baseados em energia como as *Deep Boltzmann Machines* (Suk et al., 2014) e as *Restricted Boltzmann Machines* (Li et al., 2015), e *Convolutional Autoencoders* (Hosseini-Asl et al., 2016a,b). Contudo, são as CNNs que se destacam como principal opção. No caso do diagnóstico da DA com recurso a neuro-imagens é bastante natural a opção pelas CNNs. Para além de serem a escolha óbvia, é importante focar a presente dissertação em apenas um tipo de modelo e explorar esse tipo de modelo ao máximo.

Apesar do supercomputador *Deep Blue* da IMB ter vencido o campeão mundial de xadrez Garry Kasparov em 1996, a execução fiável por computador de tarefas aparentemente triviais, como detetar um carro numa imagem ou reconhecer palavras ditas, é bastante mais recente.

As CNNs são um tipo de DNN, que se enquadram na área de DL, que emergiram do estudo do córtex visual do cérebro e têm como principal objetivo aprender características de alto nível nos dados através de convoluções. Devido ao aumento do poder computacional e da quantidade de dados, bem como o surgimento de novas técnicas de treino de DNNs, as CNNs conseguiram alcançar um desempenho sobre-humano em algumas tarefas visuais complexas. A eficácia das CNNs no reconhecimento de imagem é uma das principais razões pelas quais o mundo tem vindo a reconhecer o poder da DL. Contudo, apesar da sua efetividade com imagens no que toca a tarefas como identificar rostos, sinais de trânsito, carateres e outros objetos de dados visuais, as CNNs são também boas na análise de som. Por consequência, este tipo de ANNs estão a impulsionar grandes avanços na visão por computador, que tem aplicações óbvias na condução autónoma, nos drones, na robótica e no apoio a deficientes visuais (Adam Gibson, 2017).

As CNNs assentam num conceito poderoso que permite criar um espaço de características[3] mais robusto baseado numa técnica comum no processamento de sinais: a convolução. Por este motivo, as CNNs revelam-se mais efetivas quando existe alguma estrutura nos dados introduzidos, como por exemplo nas imagens e no áudio, que possuem um conjunto específico de padrões repetitivos e valores de entrada colocados ao lado uns dos outros e que se relacionam espacial ou temporalmente. Isto não acontece, por exemplo quando são

3 *Feature space*, na terminologia inglesa.

exportados dados de um sistema de bases de dados relacional, no qual os dados colunares exportados tendem a não ter relações estruturais no plano espacial (Adam Gibson, 2017). Sempre que existe uma topologia associada aos dados, as CNNs fazem um bom trabalho na tarefa de extrair as características importantes dos dados. Resumidamente, a arquitetura duma CNNs é organizada numa série de blocos. Os primeiros blocos são compostos por dois tipos de camadas: camadas convolucionais e camadas de *pooling*, enquanto que os últimos blocos são camadas totalmente ligadas.

Córtex Visual

São várias as invenções humanas que se inspiram na natureza e as CNNs são um bom exemplo disso mesmo. A inspiração para as CNNs é o córtex visual dos animais. O sentido humano da visão é incrivelmente avançado e permite que, dentro de frações de segundo, sejamos capazes de identificar os mais diversos objetos do nosso campo de visão sem pensar ou hesitar. Não só nos permite nomear os objetos observados, como também podemos perceber a profundidade, distinguir os seus contornos, e separar os objetos do meio onde estão inseridos (Buduma, 2017). De alguma forma os nossos olhos absorvem vóxeis, que são a unidade mínima de uma imagem digital tridimensional, e o nosso cérebro é capaz de transformar essa informação em linhas, curvas e formas primitivas com mais significado, que por exemplo nos indica que estamos a olhar para um avião (Buduma, 2017).

David H. Hubel e Torsten Wiesel realizaram diversas experiências com gatos em 1958 (Hubel, 1959) e em 1959 (Hubel and Wiesel, 1959), e alguns anos mais tarde com macacos (Hubel and Wiesel, 1968), que nos ofereceram uma visão crucial sobre a estrutura do córtex visual dos animais. Em particular, mostraram que muitos neurónios no córtex visual têm um pequeno campo recetivo local, ou seja, reagem apenas a estímulos visuais localizados numa região limitada do campo visual.

A figura 5 representa os campos recetivos locais de quatro neurónios. Os campos recetivos locais de diferentes neurónios podem sobrepor-se e em conjunto constituem o campo visual. Através das experiências realizadas, os autores mostraram também que existem neurónios que reagem apenas a imagens com linhas horizontais, enquanto outros reagem apenas a linhas com outro tipo de orientação. Ou seja, podem existir neurónios com o mesmo campo recetivo, mas que reagem a orientações de linhas diferentes. Para além disso, também notaram que alguns neurónios têm campos recetivos maiores e reagem a padrões mais complexos que são combinações dos padrões simples de nível inferior. Estas observações levaram à ideia de que os neurónios de nível superior se baseiam nas saídas dos neurónios vizinhos de nível inferior. Na figura 5 podemos ver que cada neurónio está conectado a um reduzido número de neurónios da camada anterior. É através desta arquitetura do córtex visual que somos capazes de detetar todos os tipos de padrões complexos em qualquer zona do campo visual (Géron, 2019).

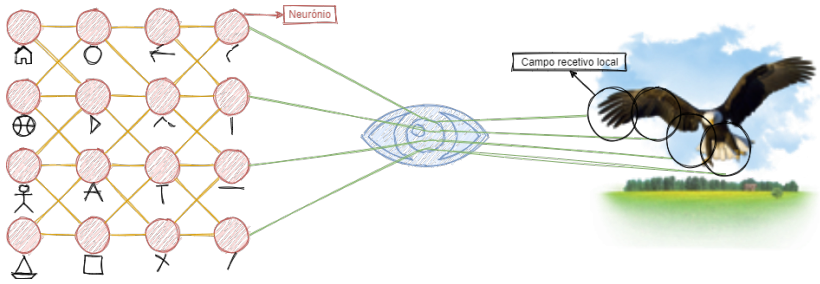


Figura 5: Campos recetivos locais no córtex visual.

Os estudos do córtex visual inspiraram o `neocognitron`, introduzido em 1980 (Fukushima, 1980), que gradualmente evolui para o que hoje denominamos por CNNs. Na evolução desta arquitetura inclui-se um importante marco: foi no artigo de Yann LeCun, Leon Bottou, Yoshua Bengio e Patrick Haffner (Lecun et al., 1998), publicado em 1998, que estes investigadores introduziram a famosa arquitetura `LeNet-5`, amplamente utilizada para reconhecer números manuscritos em cheques bancários.

Intuição

As redes neuronais *feed-forward* multi-camada recebem como entrada um único vetor unidimensional e transformam os dados com uma ou mais camadas ocultas totalmente ligadas. O resultado é dado pela rede através da camada de saída. O problema em usar estas redes neuronais *feed-forward* multi-camada é que não escalam bem com o aumento do tamanho das imagens de entrada (Adam Gibson, 2017). Por exemplo, para modelar o conhecido conjunto de dados `CIFAR-10` (figura 6), constituído por 60000 imagens distribuídas por 10 classes, apenas com 32 píxeis de largura, 32 píxeis de altura e 3 canais de informação RGB, seriam criados 3072 pesos por neurónio na primeira camada oculta, e provavelmente iríamos desejar ter mais do que um neurónio na primeira camada e múltiplas camadas ocultas, o que faria aumentar imenso o número de pesos. Uma imagem simples pode ultrapassar facilmente os 500 píxeis de largura, 500 píxeis de altura e os 3 canais de informação RGB, o que criaria 750000 pesos de ligação por cada neurónio da primeira camada oculta. Isto ilustra bem o grande número de ligações que seriam criadas numa rede neuronal *feed-forward* multi-camada quando se aplicam imagens na entrada.

A estrutura de dados de uma imagem permite alterar a arquitetura de uma rede neuronal de forma a poder tirar partido dessa estrutura. Com as CNNs é possível organizar os neurónios numa estrutura tridimensional, onde temos altura, largura e profundidade, que correspondem, respetivamente, à altura da imagem em píxeis, à largura da imagem em píxeis e aos canais RGB (figura 7).

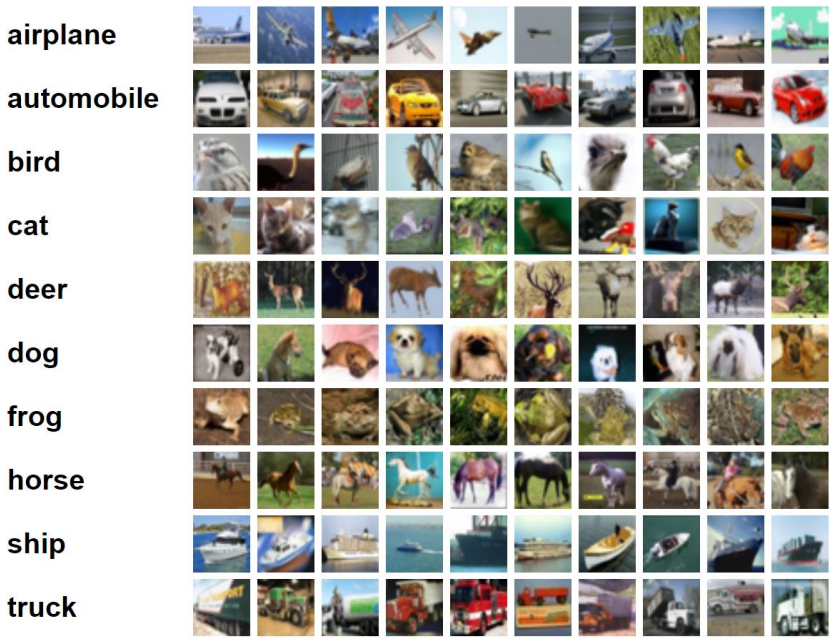


Figura 6: Amostras do conjunto de dados `CIFAR-10`.

2.6.1 *Descrição geral da arquitetura duma CNN*

Existem diversas variações na arquitetura das CNNs, mas todas se baseiam num padrão de camadas, como ilustra o exemplo da figura 8. Esta figura inclui os três principais grupos de camadas:

- Camada de entrada;
- Camadas de extração de características, ou seja, de aprendizagem;
- Camadas de classificação.

A **camada de entrada** recebe a entrada, geralmente tridimensional, na forma espacial do tamanho (*largura* $\times$ *altura*) da imagem e tem uma profundidade que representa os canais de cor, que geralmente são três no caso imagens RGB. Esta camada mantém a intensidade dos píxeis da imagem. Seguem-se as **camadas de extração de características** que seguem um padrão em que se repete a seguinte sequência:

- Camada convolucional, em que no exemplo da figura 8 foi usada a função de ativação Unidade Linear Retificada (ReLU), dado ser a mais comum na arquitetura das CNNs;
- Camada de *pooling*.

As **camadas de convolução** são responsáveis por encontrar características nas imagens e contribuir para as características de alto nível. São estas camadas que ficam com a função

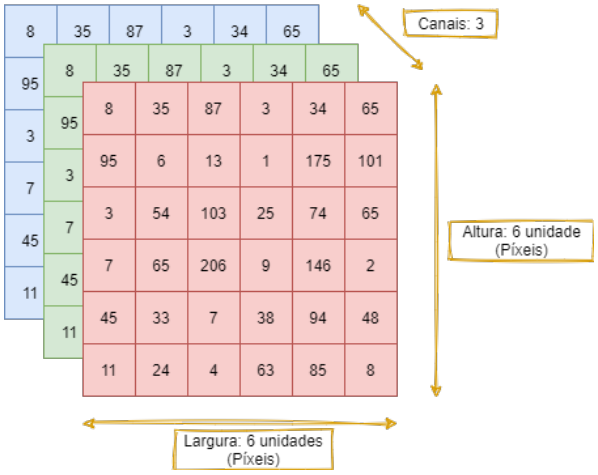


Figura 7: Representação de uma imagem.

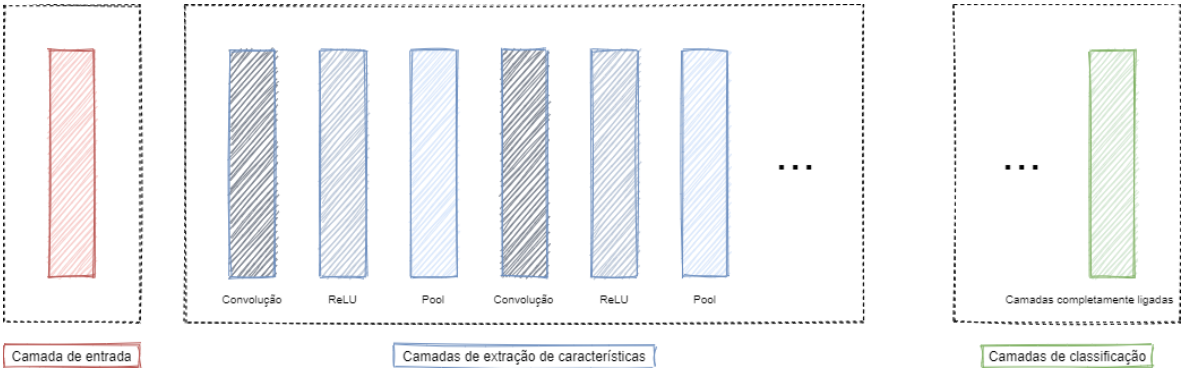


Figura 8: Representação de alto nível da arquitetura de uma CNN.

de extração automática das características, que é umas das principais particularidades dos algoritmos de DL. As camadas de convolução aplicam um número específico de filtros às imagens das camadas anteriores e criam os mapas de características. O número de mapas de características de saída é igual ao número de filtros. Os filtros aplicados podem ser 2D ou 3D (Pattanayak, 2017). As funções de ativação nas CNNs produzem uma saída com o mesmo tamanho da entrada. Normalmente é utilizada a ativação ReLU que acrescenta uma não linearidade à rede e ao mesmo tempo fornece gradientes não saturantes para as entradas positivas.

As **camadas de *pooling*** reduz os mapas de ativação 2D ao longo das dimensões de altura e largura. A profundidade ou o número de mapas de ativação não é comprometida e permanece a mesma. Por fim, as **camadas de classificação** são constituídas por uma ou mais camadas totalmente ligadas, que recebem as características de alto nível das camadas anteriores e produzem as probabilidades das classes.

2.6.2 *Camadas de entrada*

Habitualmente a informação aplicada na camada de entrada duma CNN são imagens. É nesta camada que são carregados e armazenados os dados brutos de entrada de cada imagem para processamento da rede. Estes dados de entrada especificam a largura, a altura, e o número de canais. Normalmente, o número de canais é três, correspondendo aos valores R, G e B de cada píxel.

2.6.3 *Camadas convolucionais*

As camadas convolucionais são o bloco de construção central e mais importante na arquitetura das CNNs. Estas camadas transformam os dados de entrada utilizando uma amostra de neurónios de ligação local da camada anterior. É calculado um produto entre a região dos neurónios da camada de entrada e os pesos a que estão localmente ligados na camada de saída. Geralmente, a saída destas camadas tem o mesmo tamanho ou um tamanho ligeiramente menor, mas por vezes aumenta-se o número de elementos na terceira dimensão da saída, ou seja, na profundidade (Adam Gibson, 2017).

Os neurónios da primeira camada convolucional não estão ligados a todos os píxeis da imagem de entrada, apenas a uma amostra de píxeis que correspondem ao seu campo recetivo (figura 9). Por sua vez, cada neurónio da segunda camada convolucional está ligado apenas aos neurónios localizados dentro de um pequeno retângulo da primeira camada. É esta arquitetura que permite que a rede se concentre em pequenas características de baixo nível na primeira camada oculta, e depois consegue combinar essas características para construir características de nível superior nas camadas seguintes. Isto vai de encontro ao que acontece no córtex visual humano e é esta estrutura hierárquica que faz com que as CNNs tenham excelentes resultados no reconhecimento de imagens (Géron, 2019).

Convolução

A convolução é um conceito bastante importante em diversas áreas, como a física e a matemática, e define uma ponte entre o domínio espaço/tempo e o domínio da frequência através da utilização de transformadas de Fourier. Essencialmente uma convolução é uma operação matemática que descreve uma regra de como fundir dois conjuntos de informação (Adam Gibson, 2017). No contexto das CNNs, a camada de convolução recebe as entradas, aplica um *kernel* de convolução e dá-nos um mapa de características como saída.

A operação de convolução ilustrada na figura 10 é conhecida como um detetor de características de uma CNN. A convolução recebe como entrada dados brutos ou a saída (mapa de características) de outra convolução. Esta operação é frequentemente interpretada como um filtro no qual o *kernel* filtra os dados de entrada para identificar certo tipo de informação,

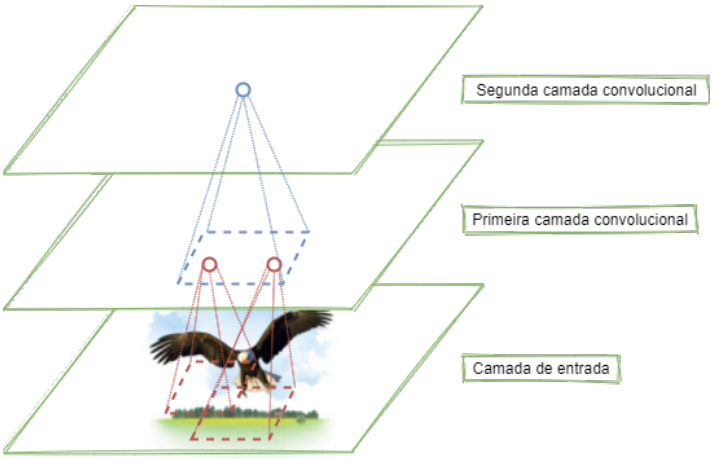


Figura 9: Camadas convolucionais com campo recetivo local.

como por exemplo identificar as linhas verticais numa imagem. O *kernel* é deslizado através dos dados de entrada para produzir as características de saída. Em cada passo o *kernel* é multiplicado pelos valores dos dados de entrada dentro dos seus limites, criando um único valor no mapa de características de saída. Na prática, o valor de saída da aplicação do *kernel* é grande se a característica que procuramos for identificada na entrada do *kernel*, e é pequeno quando não for identificada.

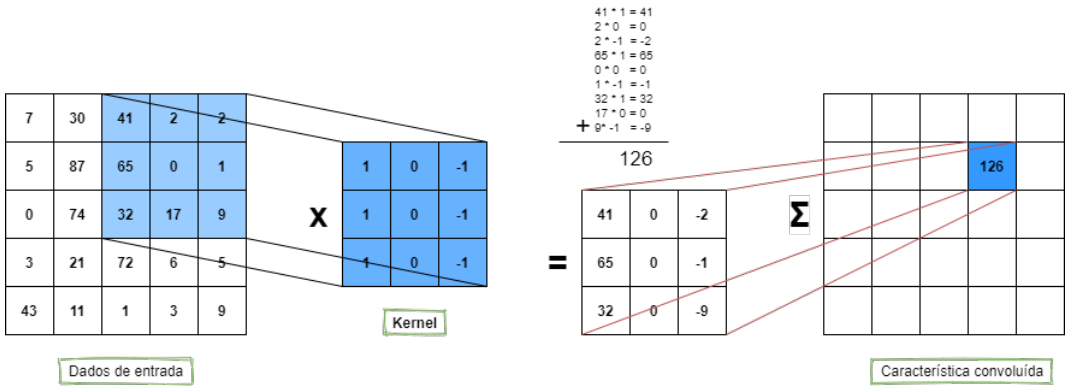


Figura 10: Operação de convolução.

Cada camada convolucional tem associado um conjunto de pesos, que geralmente são denominados de filtro ou *kernel*. Este filtro é convoluído com os dados de entrada e o resultado é um mapa de características, ou mapa de ativação. O mapa de ativação que resulta da aplicação de cada filtro é empilhado ao longo da dimensão de profundidade para construir o volume de saída 3D. As camadas convolucionais também têm associados alguns hiperparâmetros. O algoritmo do gradiente descendente é utilizado para treinar os pesos das camadas de modo a que pontuações das classes sejam consistentes com as etiquetas do

conjunto de treino. As características principais das camadas convolucionais são os filtros, os mapas de ativação, a partilha de parâmetros e os hiperparâmetros.

Filtros

Os parâmetros que caraterizam uma camada convolucional configuram o conjunto de filtros dessa camada. Os filtros têm uma largura e uma altura inferiores à largura e altura do volume de entrada, e são aplicados em toda a largura e altura do volume de entrada em forma de janela deslizante. Para além disto, os filtros são aplicados para cada profundidade do volume de entrada. A saída do filtro é calculado através operação de convolução explicada anteriormente e é conhecido como mapa de ativação desse filtro. Os filtros aprendidos só são ativados quando os padrões que pretendemos identificar ocorrem nos dados de treino, dentro do seu campo recetivo. À medida que avançamos nas camadas da CNN, os filtros são capazes de reconhecer padrões mais complexos e globais, resultantes de combinações não lineares dos padrões mais elementares.

As arquiteturas das CNNs de alto desempenho têm demonstrado que a profundidade, definida pelo número de camadas da rede, é um hiperparâmetro importante. Convém ainda referir que o número de filtros é um hiperparâmetro de cada camada convolucional e que, em cada camada podem ser produzidos diversos mapas de ativação, sendo o número de mapas definido por este hiperparâmetro. O número de mapas de ativação por ser visto como a terceira dimensão no volume de ativação de saída.

Mapas de ativação

Ao deslizarmos um filtro através ao longo da largura e da altura do volume de entrada é produzida uma saída bidimensional denominada por mapa de ativação (figura 11) para esse filtro específico. Quando dizemos que o filtro "ativa", queremos dizer que o filtro deixa a informação passar através dele a partir do volume de entrada para o volume de saída, sendo que este filtro representa os pesos que estão a ser multiplicados pela janela móvel a um subconjunto das ativações de entrada. As redes aprendem filtros que são ativados quando vêm certos tipos de características nos dados de entrada numa posição espacial específica. Criamos o volume de saída tridimensional para a camada de convolução, empilhando estes mapas de ativação ao longo da dimensão de profundidade na saída, como mostra a figura 12. Cada neurónio que constitui o volume de saída está ligado apenas a uma região local do volume de entrada, como se ilustra na figura 13.

A conectividade local deste processo é controlada pelo hiperparâmetro designado de campo recetivo, que controla qual a largura e altura do volume de entrada é mapeada pelo filtro, valores que são normalmente conhecidos como sendo as dimensões do filtro. O facto dos filtros estarem ligados apenas a um subconjunto do volume de entrada, a dinâmica da conectividade local descrita anteriormente permite não só ter uma extração de características

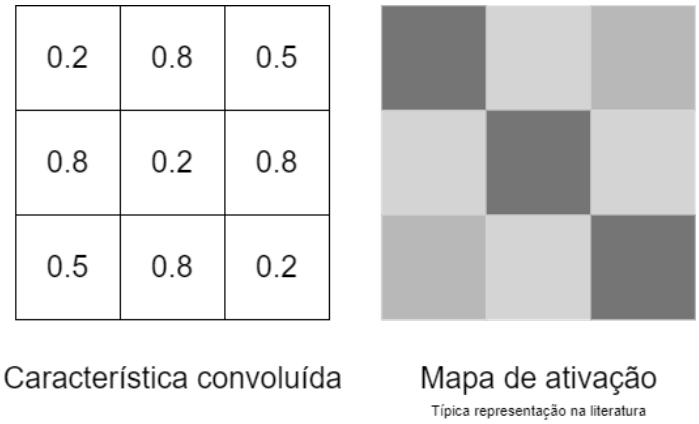


Figura 11: Convolução e mapa de ativação.

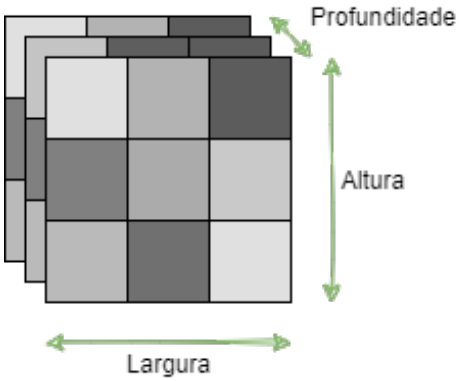


Figura 12: Volume de ativação de saída de uma camada convolucional.

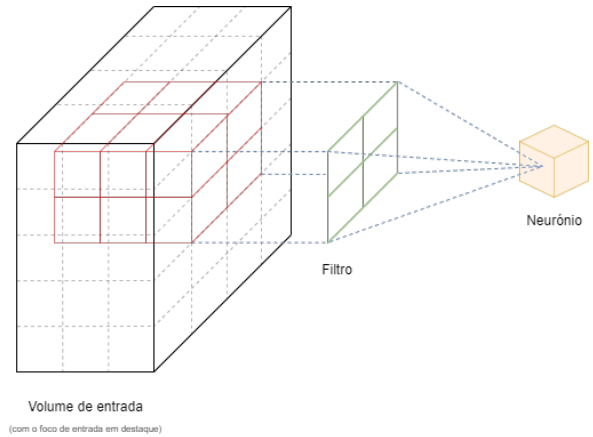


Figura 13: Geração de um volume de ativação para saída.

de qualidade, mas também reduzir o número de parâmetros por camada que precisamos de treinar. As camadas convolucionais reduzem ainda mais a contagem de parâmetros, utilizando uma técnica chamada de partilha de parâmetros.

Partilha de parâmetros

As CNNs utilizam uma técnica denominada de partilha de parâmetros que tem como principal objetivo reduzir a quantidade de parâmetros a treinar e, consequentemente, reduzir o tempo de treino pois serão utilizados menos recursos para aprender o conjunto de dados. Para implementar a partilha de parâmetros nas CNNs começamos por destacar uma única fatia bidimensional, obtendo um plano 2D como mapa de características. Em seguida, restringimos os neurónios em cada fatia (da profundidade) ao utilizar os mesmos pesos e viés. Isto dá-nos significativamente menos parâmetros (ou pesos) por cada camada convolucional. A vantagem desta simplificação é que o mesmo detetor de características funciona em qualquer zona da entrada.

Não somos capazes de tirar partido da partilha de parâmetros quando as imagens de entrada com que estamos a realizar o treino têm uma estrutura centrada. Vemos este efeito em rostos, onde esperamos que uma característica específica apareça num local específico (em rostos centrados). Neste caso, provavelmente não deveríamos utilizar partilha de parâmetros. A partilha de parâmetros é o que torna as CNNs invariantes à posição das características a detetar.

Filtros aprendidos

Os pesos dos filtros são aprendidos ao longo do treino e a técnica de partilha de parâmetros permite que, a partir do momento que a CNN aprendeu a reconhecer um padrão num local, o possa reconhecer em qualquer outro local, pois todos os neurónios de um mapa de características partilham os mesmos parâmetros. Por exemplo, a deteção de uma borda horizontal é útil em diversos locais da imagem devido à natureza invariante das imagens. Contudo, devido à partilha de parâmetros é possível aprender um filtro que identifique a borda horizontal num só lugar e depois não é preciso aprender essa característica em todas as posições da imagem.

Ativações ReLU como camadas

Nas CNNs encontramos frequentemente camadas ReLU. A camada ReLU aplica uma função de ativação sobre os dados de entrada, dando-nos a mesma dimensão de saída que a entrada para a camada, ou seja, a execução desta função sobre o volume de entrada alterará os valores de píxel mas não alterará as dimensões espaciais dos dados de entrada na saída. As camadas ReLU não têm parâmetros nem hiperparâmetros adicionais.

Hiperparâmetros da camada convolucional

Os parâmetros que definem a disposição espacial e o tamanho do volume de dados de saída de uma camada convolucional são:

- **Tamanho do filtro ou do *kernel***: Normalmente o tamanho dos filtros é pequeno, ou seja, possuem uma largura, altura e profundidade pequenas. Por exemplo, uma primeira camada convolucional pode ter filtros com tamanho $5 \times 5 \times 3$, significando isso que o filtro tem 5 píxeis de altura, 5 píxeis de largura e profundidade igual a 3.
- ***Stride***: O *stride* define o salto que a janela deslizante de filtragem dá em cada passo. Configurações baixas de *stride*, por exemplo 1, levam a que sejam criados volumes de saída maiores, pois a janela deslizante dá passos de apenas uma unidade. Os valores baixos de *stride* levam também a uma maior sobreposição dos campos recetivos, enquanto que valores de *stride* altos resultam em menos sobreposição e consequentemente volumes de saída mais pequenos.
- **Profundidade de saída**: É possível escolher manualmente a profundidade do volume de saída. O hiperparâmetro de profundidade controla a contagem de neurónios na camada convolucional que está ligada à mesma região do volume de entrada.
- ***Zero-padding***: Com este hiperparâmetro controlamos o tamanho espacial dos volumes de saída. Este controlo é importante quando queremos gerar um volume saída com o mesmo tamanho do volume de entrada.

Requisitos de memória

Um problema normalmente associado às CNNs é o facto das camadas convolucionais requerem uma grande quantidade de memória RAM. Isto é especialmente verdade durante a fase de treino, pois a passagem inversa da retro-propagação requer que sejam mantidos em memória todos os valores intermédios calculados durante a passagem para a frente.

Durante a inferência, ou seja, quando fazemos uma previsão para uma nova instância, a RAM ocupada por uma camada pode ser libertada assim que a camada seguinte tiver sido calculada, pelo que só é necessária a quantidade de RAM necessária por duas camadas consecutivas. Mas durante o treino tudo o que é calculado na passagem para a frente tem que ser preservado para a passagem inversa, pelo que a quantidade de RAM necessária é pelo menos a quantidade total de memória necessária a todas as camadas.

Se o treino falhar devido a um erro do tipo fora de memória[4], existem diversas formas para tentar contornar esse problema: pode-se tentar reduzir o tamanho do *mini-batch*, pode-se tentar reduzir a dimensionalidade dos dados usando um *stride* superior ou remover mesmo algumas camadas, pode-se tentar usar *floats* com 16 *bits* em vez de *floats* de 32 *bits* e pode-se distribuir a CNN por múltiplas máquinas.

4 *Out-of-memory*, na terminologia inglesa.

2.6.4 *Camadas de pooling*

As camadas de *pooling* são normalmente inseridas entre camadas convolucionais com o objetivo de reduzir progressivamente a dimensão espacial dos dados. No fundo este objetivo pode ser visto como criação de sub-amostras ao encolher-se os dados de entrada, reduzindo a carga computacional, a utilização de memória e o número de parâmetros. Nestas camadas, tal como nas camadas convolucionais, cada neurónio está ligado a um número limitado de neurónios de saída da camada anterior, localizados dentro de um pequeno campo recetivo, e são caracterizados pelo seu tamanho, o *stride* e tipo de *padding*. Um neurónio da camada de *pooling* não tem pesos e tudo o que faz é agregar as entradas usando uma função de agregação, como por exemplo o máximo ou a média. A camada de *pooling* mais comum é o *pooling* máximo, seguido do *pooling* médio (Adam Gibson, 2017). Estas camadas utilizam filtros para realizar o processo de redução, ou *downsampling*, do tamanho do volume de entrada.

É importante notar que as camadas de *pooling* funcionam de forma independente em cada fatia de profundidade da entrada, pelo que não afetam a profundidade. A configuração mais comum numa camada de *pooling* é aplicar *pooling* máximo com filtro de tamanho 2×2, com *stride* igual 2 e sem *padding*. Utilizando esta configuração como exemplo, podemos ver pela figura 14 que apenas o valor máximo de entrada, dentro do cada campo recetivo, passa para a camada seguinte, sendo descartadas as restantes entradas. O campo recetivo localizado no canto inferior direito do volume de dados de entrada da figura 14 tem os valores 5, 9, 7 e 3, sendo que apenas o valor máximo 9 passa para a camada seguinte. Com um *stride* igual a 2, a imagem de saída tem metade da altura e metade da largura da imagem de entrada. No exemplo da figura 14 passamos de uma imagem 10×8 para uma imagem com 5×4. A operação de redução apresentada faz com que 75% das ativações sejam descartadas.

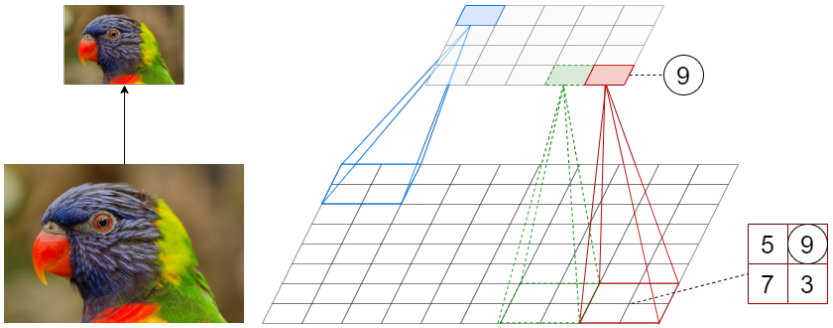


Figura 14: Camada de *pooling* máximo com *kernel* de tamanho 2×2, *stride* igual a 2, sem *padding*.

Para além de reduzirem os cálculos, estas camadas reduzem a utilização de memória e o número de parâmetros, e introduzem algum grau de invariância a pequenas translações. Um exemplo desta invariância pode ser vista na figura 15, onde assumimos que os píxeis brilhantes têm um valor inferior aos píxeis escuros, e consideramos 3 imagens. As imagens

do centro e da direita são as mesmas que a imagem da esquerda, mas deslocadas por um e dois píxeis para a direita, respetivamente. Após passarem por uma camada de *pooling* máximo com *kernel* de tamanho 2×2 e *stride* 2, podemos ver que as saídas para as imagens da esquerda e do centro são iguais, significando isto invariância de tradução. Para a imagem da direita a saída é diferente, estando deslocada de um píxel para a direita, mas ainda há invariância de 75% (Géron, 2019). Ao inserir uma camada de *pooling* máximo, após um pequeno número de camadas convolucionais e repetindo este padrão de camadas, é possível obter um grau de invariância de tradução a uma escala maior. Além disso, o *pooling* máximo também oferece uma pequena quantidade de invariância rotacional e uma ligeira invariância de escala. Tal invariância, mesmo que limitada, pode ser útil nos casos em que a previsão não deve depender destes detalhes, como por exemplo nas tarefas de classificação (Géron, 2019).

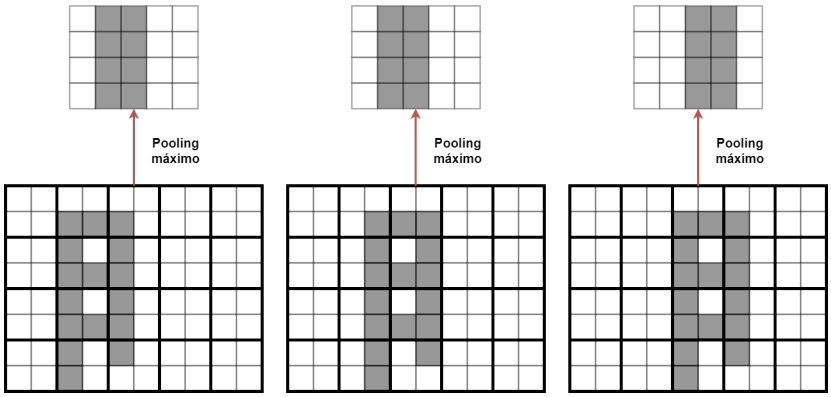


Figura 15: Invariância a pequenas translações.

Apesar de todas as vantagens, o *pooling* máximo tem algumas desvantagens. Em primeiro lugar é muito destrutivo: no exemplo apresentado na figura 14 podemos ver que mesmo com um *kernel* pequeno de 2×2 e um *stride* igual a 2, a saída será duas vezes menor em ambas as dimensões, por isso a sua área será quatro vezes menor, fazendo com que 75% das ativações sejam descartadas. Em algumas aplicações a invariância não é desejável, por exemplo na segmentação semântica. Neste caso, a classificação de cada píxel da imagem depende do objeto a que o píxel pertence. Se a imagem de entrada for deslocada em 1 píxel para a direita, a saída também deve ser deslocada do mesmo píxel para a direita. O objetivo neste caso é a equivariância e não a invariância, ou seja, uma pequena alteração na entrada deve traduzir-se numa pequena alteração correspondente na saída.

2.6.5 *Camadas totalmente ligadas*

As camadas totalmente ligadas são utilizadas para calcular a pontuação de cada classe a gerar na saída da rede. As dimensões do volume de saída são $[1 \times 1 \times N]$, em que N é o

número de classes em consideração. Por exemplo, no caso do conjunto de dados `CIFAR-10` o N é igual a 10, pois existem 10 classes de objetos no conjunto de dados. Numa camada totalmente ligada, cada neurónio está ligado a todos os neurónios da camada anterior. Estas camadas realizam transformações no volume de dados de entrada através das funções de ativação e dos pesos e viés dos neurónios. No caso das arquiteturas das CNNs algumas utilizam múltiplas camadas totalmente ligadas no final da rede, seguidas de uma camada *softmax* para saída (figura 16).

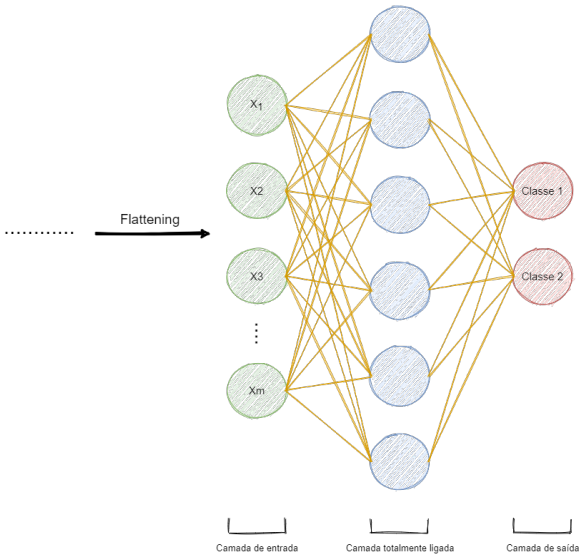


Figura 16: Camadas totalmente ligadas numa *CNN*.

2.7 Arquiteturas de Redes Neuronais Convolucionais

Como visto anteriormente, a arquitetura típica duma CNN empilha algumas camadas convolucionais, seguidas de uma camada ReLU, depois uma camada de *pooling*, novamente algumas camadas convolucionais com as respetivas camadas ReLU, seguida de outra camada de *pooling*, e por aí em diante seguindo esta ordem (figura 8). No final desta sequência encontra-se uma rede neuronal *feed-forward* regular, composta por algumas camadas totalmente ligadas com as respetivas ativações, que normalmente são ReLU, sendo que a camada final produz a previsão utilizando a função `softmax` para obter a probabilidade associada a cada uma das classes.

Esta seção fará um apanhado das arquiteturas de redes neuronais convolucionais amplamente utilizadas. Estas arquiteturas de rede não são utilizadas apenas em classificação mas também, com pequenas modificações, podem ser utilizadas em segmentação, localização e deteção. Além disso existem versões pré-treinadas de cada uma destas redes que permitem

à comunidade fazer a transferência de aprendizagem ou afinar[5] os modelos. Alguns destes modelos têm mostrado sucesso em concursos como o Desafio de Reconhecimento Visual em Grande Escala ImageNet (ILSVRC) (Deng et al., 2009). Segue-se uma lista das arquiteturas mais emblemáticas.

`LeNet-5`

Possivelmente a `LeNet-5` é a arquitetura de CNN mais conhecida, tendo sido esta uma das primeiras arquiteturas de sucesso (Lecun et al., 1998). Foi criada por Yann LeCun em 1998 e era originalmente usada para reconhecer dígitos manuscritos como os incluídos no famoso conjunto de dados MNIST. Esta arquitetura é bastante simples, sendo constituída por 2 camadas convolucionais e 3 camadas totalmente ligadas, daí o "5" no nome. É frequente o nome das redes neuronais incluir o número de camadas convolucionais e totalmente ligadas que formam a rede. As camadas de *pooling* médio, tal como as conhecemos agora, eram chamadas de camadas de sub-amostragem[6] e tinham pesos treináveis, o que não é a prática atual. Esta arquitetura tem cerca de 60000 parâmetros e tornou-se o modelo para as arquiteturas mais recentes: empilhamento de convoluções com função de ativação, camadas de *pooling*, e terminar a rede com uma ou várias camadas totalmente ligadas.

A figura 17 e a tabela 1 contêm um resumo das camadas existentes na arquitetura `LeNet-5`. Esta rede recebe como entrada as imagens do conjunto de dados MNIST, que são de 28×28 píxeis, mas depois aplica-se às imagens MNIST *zero-padding* para obter imagens de 32×32 píxeis, e daí o tamanho da entrada da rede ser 32×32. A restante rede não usa qualquer *padding* e por isso o tamanho diminui à medida que se avança na rede.

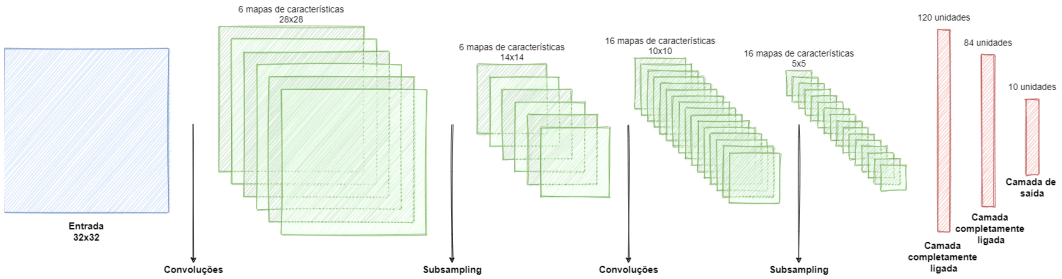


Figura 17: Representação da arquitetura da `LeNet-5`.

`AlexNet`

A arquitetura `AlexNet` foi desenvolvida por Alex Krizhevsky, resultando daí o seu nome, mais Ilya Sutskever e Geoff Hinton, em 2012 e venceu o desafio ImageNet ILSVRC 2012 por uma grande margem (Krizhevsky et al., 2012). Esta arquitetura é bastante semelhante à

5 *Fine-tuning*, na terminologia inglesa.
6 *sub-sampling*, na terminologia inglesa.

Camada	Mapas	Tamanho	Tamanho do Kernel	Stride	Ativação
Entrada	1	32x32	-	-	-
Convolucional (1)	6	28x28	5x5	1	tanh
Pooling médio (1)	6	14x14	2x2	2	-
Convolucional (2)	16	10x10	5x5	1	tanh
Pooling médio (2)	16	5x5	2x2	2	-
Totalmente ligada (1)	-	120	-	-	tanh
Totalmente ligada (2)	-	84	-	-	tanh
Saída	-	10	-	-	RBF

Tabela 1: Arquitetura da `LeNet-5`.

`LeNet-5`, mas muito maior e mais profunda e foi a primeira arquitetura a empilhar camadas convolucionais diretamente umas sobre as outras, fugindo à arquitetura comum que consistia em empilhar uma camada de *pooling* em cima de uma camada convolucional. Foi também a primeira a usar ReLU como função de ativação. As camadas desta arquitetura podem ser vistas na figura 18.

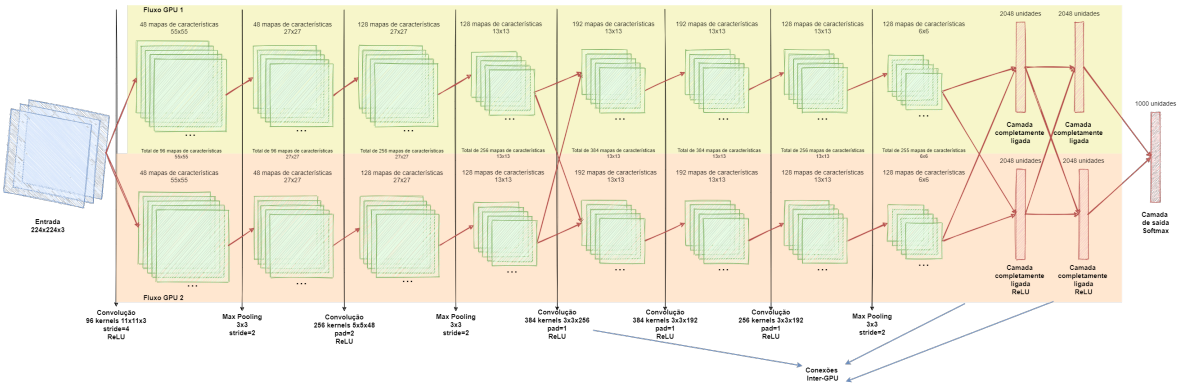


Figura 18: Representação da arquitetura da `AlexNet`.

A `AlexNet` é constituída por cinco camadas convolucionais, camadas de *pooling* máximo, três camadas totalmente ligadas e a camada de saída. Na entrada na rede são aplicadas imagens de tamanho $224 \times 224 \times 3$. Todas as funções de ativação usadas nas camadas convolucionais são ReLU e exige aproximadamente 60 milhões de parâmetros. Para reduzir o excesso de *overfitting*, os autores utilizaram duas técnicas de regularização: primeiro aplicaram às camadas totalmente ligadas uma taxa de 50% de *dropout* (seção 2.11); em segundo lugar, efetuaram um aumento de dados (seção 2.12) deslocando as imagens de treino de um valor aleatório, virando-as horizontalmente e alterando as condições de iluminação.

A `AlexNet` também utiliza uma etapa de normalização competitiva imediatamente após a etapa ReLU da primeira e terceira camadas convolucionais, denominada normalização de

resposta local[7]. A normalização de resposta local faz com que os neurónios mais fortemente ativados inibam outros neurónios localizados na mesma posição em mapas de características vizinhos. A ativação competitiva também foi observada em neurónios biológicos. Isto encoraja a especialização de diferentes mapas de características, afastando-os e forçando-os a explorar uma gama mais ampla de características, melhorando a generalização.

Uma variante da `AlexNet` chamada `ZF Net` foi desenvolvida por Matthew Zeiler e Rob Fergus e ganhou o desafio do ILSVRC de 2013. É essencialmente uma `AlexNet` com alguns hiperparâmetros afinados, tais como o número de mapas de características, o tamanho do *kernel* e os *strides*.

GoogLeNet

A arquitetura `GoogLeNet`, também conhecida como `Inception-V1`, foi desenvolvida por Christian Szegedy e a sua equipa na Google e venceu o desafio ILSVRC 2014 (Szegedy et al., 2014). A primeira grande diferença em relação às arquiteturas anteriores é que esta rede é muito mais profunda. Esta arquitetura introduz a utilização de sub-redes, denominadas por módulos *inception*, que permitem à `GoogLeNet` utilizar parâmetros muito mais eficientemente do que as arquiteturas anteriores: `GoogLeNet` tem na realidade 10 vezes menos parâmetros do que `AlexNet`, ou seja, cerca de 6 milhões em vez de 60 milhões.

Na figura 19 podemos ver a arquitetura de um módulo *inception*. A entrada é primeiro copiada e alimentada em 4 camadas distintas que utilizam a função de ativação ReLU. O segundo conjunto de camadas convolucionais utiliza diferentes tamanhos de *kernel*, 1×1, 3×3 e 5×5, o que permite captar padrões em diferentes escalas e mais tarde juntar todas as características captadas na saída do módulo[8]. Todas as camadas, incluindo as camadas de *pooling*, usam *stride* igual a 1 e *padding* do tipo `SAME`, o que faz com que as saídas destas camadas tenham a mesma dimensão que as suas entradas, permitindo concatenar todas as saídas ao longo da dimensão de profundidade na camada de concatenação[9], ou seja, empilhar os mapas de características das quatro camadas convolucionais superiores.

As camadas convolucionais existentes nos módulos *inception* com *kernels* de tamanho 1×1, apesar de não serem capazes de capturar características uma vez que olham para um píxel de cada vez, servem 3 propósitos[10]:

- Embora não possam capturar padrões espaciais, uma vez que olham para um píxel de cada vez, podem capturar padrões ao longo da dimensão de profundidade;

7 *Local response normalization*, na terminologia inglesa. Pode ser implementado em TensorFlow com a função tf.nn.local_response_normalization() e utilizada em modelos Keras, recorrendo a uma camada *Lambda*.

8 Esta ideia é motivada por Arora et al. (2013).

9 A camada de concatenação pode ser implementada em TensorFlow utilizando a operação tf.concat com o parâmetro axis=3 (profundidade).

10 Foram introduzidas em Lin et al. (2013) e eram usadas para adicionar não linearidade e assim aumentar o poder representativo da rede, uma vez que os autores acreditam que os dados estão em forma de não-linearidade.

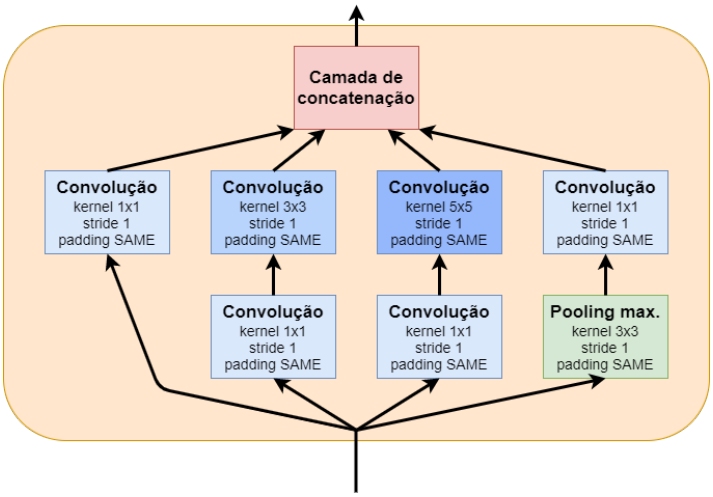


Figura 19: Módulo *inception*.

- São configuradas para produzir menos mapas de características do que as suas entradas, pelo que servem como camadas de estrangulamento[11], o que significa que reduzem a dimensionalidade. Isto reduz o custo computacional e o número de parâmetros, acelerando o treino e melhorando a generalização;

- Cada par de camadas convolucionais $[1 \times 1, 3 \times 3]$ e $[1 \times 1, 5 \times 5]$ atua como uma única e poderosa camada convolucional, capaz de captar padrões mais complexos. De facto, em vez de percorrer um simples classificador linear através da imagem, como faz uma única camada convolucional, este par de camadas convolucionais corresponde a passar uma rede neural com duas camadas pela imagem. Devido à função de ativação da convolução 1×1, também é acrescentada uma não linearidade ao modelo.

Uma desvantagem no uso de módulos *inception* é o aumento do número de hiperparâmetros a ajustar. O número de *kernels* convolucionais é um hiperparâmetro para cada camada convolucional, logo por cada módulo *inception* que se adiciona, juntam-se mais seis hiperparâmetros para ajustar.

Analisando a arquitetura da `GoogleLeNet`, apresentada na figura 20, podemos ver que esta arquitetura é bastante profunda e utiliza nove módulos *inception*. Podemos também ver o número de mapas de características produzidas por cada camada convolucional e cada camada de *pooling*, bem como os mapas de características produzidos por cada camada convolucional dentro dos módulos *inception*, representados por retângulos arredondados encarnados (de acordo com a representação da figura 19). Todas as camadas convolucionais utilizam ReLU como função de ativação.

11 *Bottleneck layers*, na terminologia inglesa.

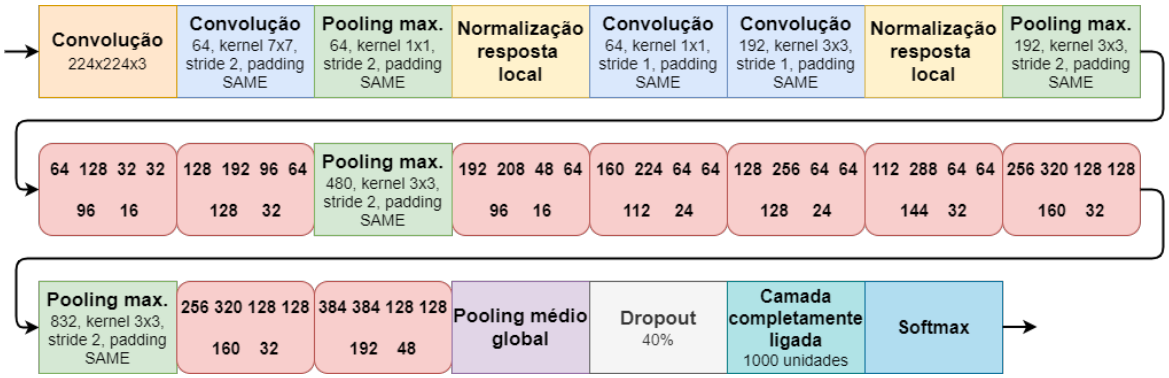


Figura 20: Representação da arquitetura da `GoogLeNet`.

Podemos ver também que existe uma camada de *pooling* médio global[12] que produz a média de cada mapa de características: isto deixa cair qualquer informação espacial remanescente, o que é ótimo, uma vez que já não há muita informação espacial nesta fase.

Várias variantes da arquitetura `GoogLeNet` foram posteriormente propostas pelos investigadores da Google, incluindo `Inception-v3` e `Inception-v4`, utilizando módulos *inception* ligeiramente diferentes e atingindo um desempenho ainda superior.

VGGNet

A `VGGNet`, também conhecida como `VGG-16`, foi desenvolvida por Karen Simonyan e Andrew Zisserman do grupo *Visual Geometry Group* (VGG), e ficou em segundo lugar no desafio ILSVRC 2014 (Simonyan and Zisserman, 2014). Esta CNN tem uma arquitetura muito simples e clássica, mas bastante profunda, com 2 ou 3 camadas convolucionais, uma camada de *pooling*, depois novamente 2 ou 3 camadas convolucionais, uma camada de *pooling*, e assim por diante, contendo um total de 16 camadas: 13 camadas convolucionais e 3 camadas totalmente ligadas. Esta ideia de que a forma mais direta de melhorar o desempenho das DNNs é aumentando o seu tamanho era defendida pelos autores desta rede. A `VGGNet` é constituída por 138 milhões de parâmetros e ocupa cerca de 500 MB de espaço de armazenamento.

Em vez de usar um tamanho grande para o *kernel* da convolução, utiliza *kernels* de 3×3 seguidos de ativação ReLU, e um *pooling* máximo com um campo recetivo de 2×2. Os autores defendem a ideia que usar duas camadas convolucionais com filtros de 3×3 equivale a usar uma camada convolucional com filtro de 5×5, mantendo assim as vantagens de usar filtros pequenos, como a redução do número de parâmetros, bem como conseguir uma maior não-linearidade devido ao uso de duas camadas de ativação ReLU em vez de uma. Uma outra propriedade que caracteriza esta rede é que, à medida que se avança na rede as dimensões espaciais do volume de entrada diminuem, devido à convolução e ao *pooling*

12 Ideia também introduzida por Lin et al. (2013).

máximo, e o número de mapas de características aumenta devido ao aumento do número de filtros (Pattanayak, 2017).

A figura 21 apresenta a arquitetura da `VGG-16`. Esta arquitetura simples e profunda tem ganho muita popularidade desde então e mostrou que a profundidade da rede é um fator importante para o bom desempenho. O grupo VGG também desenvolveu uma variante mais profunda, a `VGG-19`.

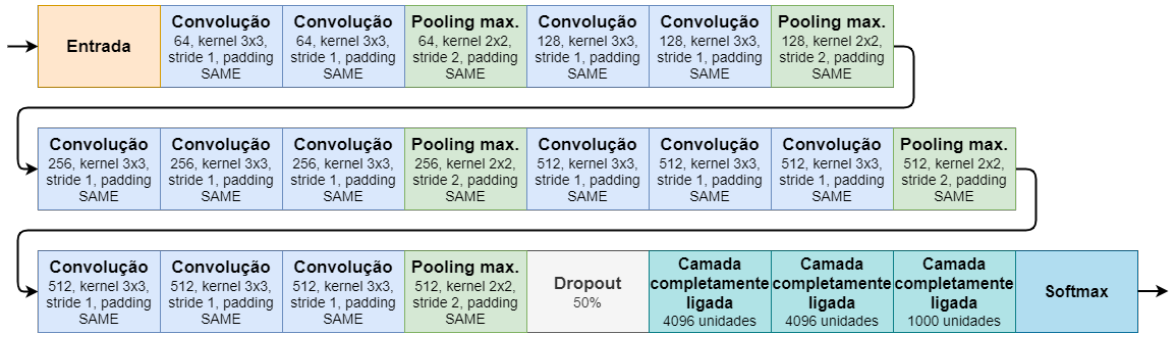


Figura 21: Representação da arquitetura da `VGGNet`.

ResNet

Desenvolvida por Kaiming He em 2015, a Rede Residual (`Residual Network` ou `ResNet`) foi a vencedora do desafio ILSVRC 2015 (He et al., 2015). Composta por 152 camadas, confirmou a tendência em desenvolver modelos cada vez mais profundos com menos parâmetros. Foi também uma das primeiras redes neuronais profundas a utilizar a técnica de *Batch Normalization* (BN) (seção 2.10).

Para permitir o treino de uma rede tão profunda são utilizadas ligações de salto[13], também conhecidas como ligações de atalho[14]. Isto é, ao sinal de entrada de uma camada será adicionado a saída de uma camada anterior. Apesar de não terem sido os primeiros a utilizar esta técnica, foi esta arquitetura que a popularizou. Treinar uma rede neuronal é fazer com que ela modele uma função objetivo $h(x)$. Ao adicionar a entrada x à saída da rede, isto é, ao adicionar uma ligação de salto a rede neuronal é forçada a modelar $f(x) = h(x) - x$ em vez de $h(x)$. A isto chama-se aprendizagem residual (figura 22) (Géron, 2019).

Quando se inicializa uma rede neuronal, geralmente os seus pesos estão próximos de zero, o que leva a rede a produzir valores próximos de zero. Se for adicionada uma ligação de salto, a rede resultante produz valores semelhantes às suas entradas, por outras palavras, inicialmente modela a função de identidade. O facto da função destino estar próxima da função identidade faz com que o treino seja muito mais rápido. Alem disso, acrescentar

13 *Skip connections*, na terminologia inglesa.
14 *Shortcut connections*, na terminologia inglesa.

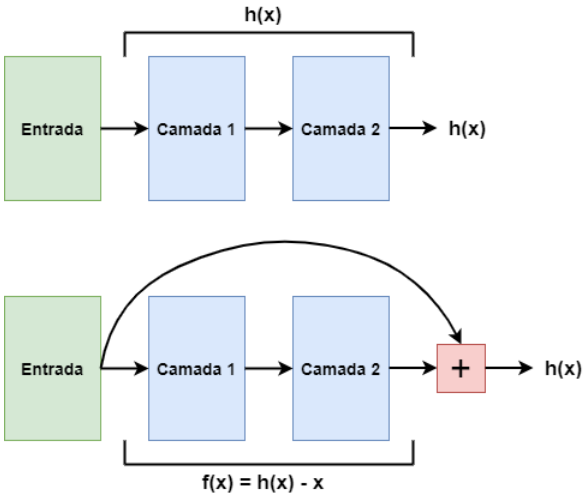


Figura 22: Aprendizagem residual.

muitas ligações de salto permite à rede fazer progressos mesmo que as várias camadas não tenham começado a aprender (Géron, 2019). Uma rede residual pode ser vista como uma pilha de unidades residuais, onde cada unidade residual é uma pequena rede neuronal com uma ligação de salto.

Analisando a arquitetura da `ResNet` apresentada na figura 23 podemos ver que esta é bastante simples e semelhante à `GoogLeNet`, pois começa e termina da mesma forma. A exceção está na utilização de *dropout*. Entre a parte inicial e final está uma pilha bastante profunda de unidades residuais. Estas unidades residuais são bastante simples: incluem duas camadas convolucionais (sem camada de *pooling*), utiliza BN e ativação ReLU, *kernels* de tamanho 3×34 e preserva as dimensões espaciais, ou seja, o *stride* é igual a 1 e o *padding* é `SAME`.

A cada poucas unidades residuais o número de mapas de características é duplicado, e a sua altura e largura são reduzidas para metade, através do uso de camadas convolucionais com *stride* igual a 2. Isto faz com que as entradas a somar nas saídas das unidades residuais tenham tamanhos diferentes, o que impossibilita a sua adição. Este problema ocorre na ligação de salto representada a cor encarnada na figura 23. Para resolver este problema, as entradas são passadas através de uma camada convolucional de tamanho 1×1, com *stride* 2 e em número igual à quantidade de mapas de características da unidade residual (figura 24).

Existem várias variantes da `ResNet`, como a `ResNet-34` e a `ResNet-152`, desenvolvida pela Microsoft e vencedora do concurso ILSVRC 2015. A equipa da Google criou uma arquitetura bastante famosa, a `Inception-v4`, que junta características da `GoogLeNet` e da `ResNet` (Szegedy et al., 2016).

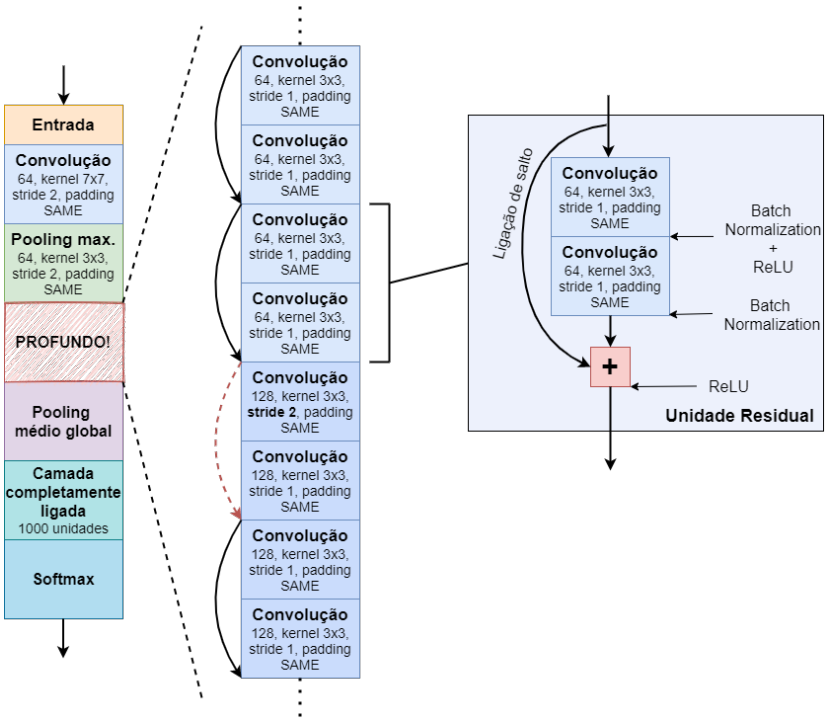


Figura 23: Representação da arquitetura da `ResNet`.

Xception

Uma outra arquitetura bastante conhecida e inspirada na arquitetura `GoogLeNet` é a `Xception`, que significa *Extreme Inception*, e foi proposta por François Chollet o autor do Keras (Chollet, 2016). Esta arquitetura teve um desempenho significativamente superior ao da `Inception-v3` numa enorme tarefa de visão: classificar 350 milhões de imagens em 17.000 classes. Tal como a `Inception-v4`, funde as ideias de `GoogLeNet` e `ResNet`, mas substitui os módulos *inception* por um tipo especial de camada, chamada de camada convolucional separável em profundidade[15], ou simplesmente camada convolucional separável. Estas camadas já tinham sido utilizadas anteriormente em algumas arquiteturas de CNNs, mas não eram tão centrais como na arquitetura `Xception` (Géron, 2019).

Enquanto uma camada convolucional regular, como as que temos vindo a falar, utiliza filtros que tentam capturar simultaneamente padrões espaciais (como por exemplo curvas) e padrões de canal transversal (como por exemplo "boca + nariz + olhos = face"), uma camada convolucional separável faz a forte suposição de que padrões espaciais e padrões de canal transversal podem ser modelados separadamente (Géron, 2019). Assim, a convolução é composta de duas partes: a primeira parte aplica um único filtro espacial para cada mapa de características de entrada, depois a segunda parte procura exclusivamente padrões de canal transversal, ou seja, é uma camada convolucional regular com filtros de tamanho 1×1.

15 *Depthwise separable convolution*, na terminologia inglesa.

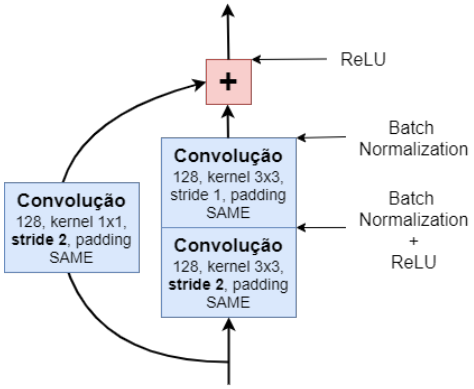


Figura 24: Ligação de salto.

Uma vez que as camadas separáveis convolucionais têm apenas um filtro espacial por canal de entrada, deve-se evitar utilizá-las depois das camadas que têm poucos canais, como por exemplo a camada de entrada. Por este motivo, a arquitetura `Xception` começa com 2 camadas convolucionais regulares, mas depois o resto da arquitetura utiliza apenas convoluções separáveis, num total de 34, mais algumas camadas de *pooling* máximo e as camadas finais habituais, ou seja, uma camada de *pooling médio global* e uma camada de saída densa.

As convoluções separáveis utilizam menos parâmetros, menos memória e menos cálculos do que as camadas convolucionais regulares, e em geral apresentam melhor desempenho, pelo que se deve considerar a sua utilização por omissão, exceto depois de camadas com poucos canais.

`SENet`

A arquitetura *Squeeze-and-Excitation Network* (`SENet`) foi a vencedora do desafio ILSVRC 2017 (Hu et al., 2017). Esta arquitetura alarga as arquiteturas já existentes, como as redes `Inception` ou `ResNets`, fazendo com que o seu desempenho aumente. As versões alargadas das redes `Inception` e `ResNet` são chamadas `SE-Inception` e `SE-ResNet`, respetivamente. O aumento no desempenho nesta arquitetura surge do facto da `SENet` acrescentar uma pequena rede neuronal, denominada bloco SE, a cada unidade da arquitetura original, ou seja, a cada módulo *inception* ou a cada unidade residual (figura 25).

Um bloco SE analisa a saída da unidade a que está ligado, concentrando-se exclusivamente na dimensão de profundidade (não procura qualquer padrão espacial), e aprende quais as características que são normalmente ativas em conjunto com mais regularidade. Após essa análise, utiliza a informação para recalibrar os mapas de características, como mostra a figura 26. Por exemplo, um bloco SE pode aprender que as bocas, narizes e olhos aparecem geralmente juntos em imagens. Assim, ao identificar uma boca e um nariz espera ver

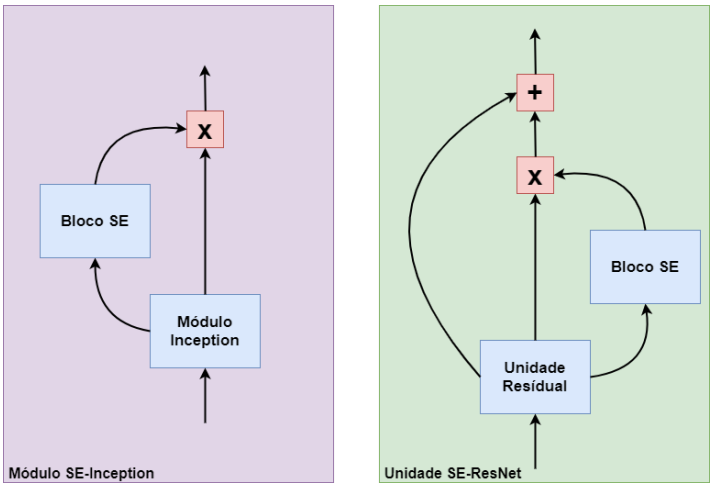


Figura 25: Módulo *SE-Inception* e unidade *SE-ResNet*.

também os olhos. Se um bloco SE vir uma forte ativação nos mapas de características da boca e do nariz, mas apenas uma ligeira ativação no mapa de características dos olhos, irá aumentar o mapa de características dos olhos. Mais precisamente, irá reduzir os mapas de características irrelevantes. Se os olhos forem confundidos com outra caraterística, a recalibração do mapa de características ajudará a resolver a ambiguidade (Géron, 2019).

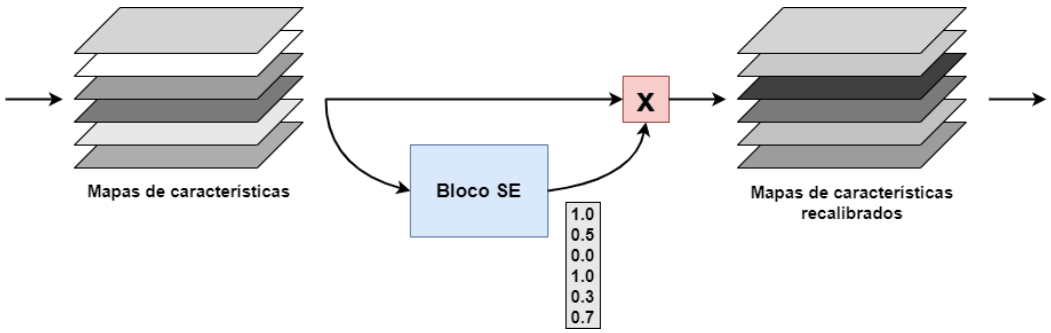


Figura 26: Recalibração dos mapas de características com um bloco SE.

Um bloco SE é composto apenas por 3 camadas: uma camada de *pooling* médio global, uma camada densa escondida usando a função de ativação ReLU, e uma camada densa de saída usando a função de ativação `sigmoid` (figura 27). A camada de *pooling* médio global calcula a ativação média para cada mapa de características. Por exemplo, se a entrada contiver 256 mapas de características, produzirá 256 números. A camada seguinte é onde acontece a compressão de informação, pois esta camada tem muito menos que 256 neurónios, normalmente tem 16 vezes menos neurónios do que o número de mapas de características. Por exemplo, se tiver 16 neurónios, os 256 números são comprimidos num vetor de tamanho $\frac{256}{16} = 16$. Esta é uma representação vetorial compacta da distribuição das características, também designada de *embebedding*. Este passo de estrangulamento obriga o bloco SE a

aprender uma representação geral das combinações das características. Finalmente, a camada de saída no vetor compacto e produz um vetor de recalibração contendo um valor numérico por mapa de características entre 0 e 1, num total de 256 valores no exemplo citado. Os mapas de características são então multiplicados por este vetor de recalibração, de modo que as características irrelevantes (com uma pontuação de recalibração baixa) sejam reduzidas, enquanto as características relevantes (com uma pontuação de recalibração próxima de 1) devem ser mantidas sem grande alteração.

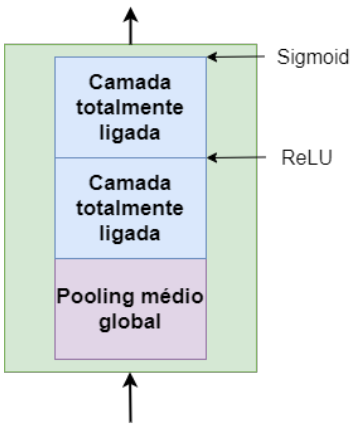


Figura 27: Arquitetura do bloco SE.

Todas as arquiteturas mencionadas foram treinadas com o conjunto de dados Imagenet e em alguma altura foram vencedoras, ou bem classificadas, no concurso ILSVRC. Dada a sua popularidade, a grande maioria das bibliotecas de desenvolvimento de DNNs, como o Keras, disponibilizam estes modelos já treinados com o conjunto de dados Imagenet. Isto permite efetuar apenas um ajuste fino dos modelos com dados do domínio de aplicação desejado, possivelmente em muito menor quantidade do que o Imagenet. O ajuste fino é uma técnica que consiste em aproveitar os pesos de uma rede neuronal já treinada e, utilizando aprendizagem supervisionada, aperfeiçoá-los para uma área de aplicação específica (Erhan et al., 2010). Esta técnica é bastante utilizada atualmente.

Idealmente, a rede neuronal deveria ser treinada com imagens semelhantes às do domínio de aplicação. Ou seja, como na presente dissertação o problema em estudo é o diagnóstico da DA, a rede neuronal a utilizar deveria ser treinada com imagens MRI. Contudo, isto não é muitas vezes possível, uma vez que a quantidade de dados disponíveis é limitada, como acontece principalmente com imagens médicas (secção 2.13). Mesmo com este inconveniente, esta técnica continua a ser muito usada, pois o Imagenet não é um conjunto de dados qualquer, é um conjunto de dados muito grande e variado, onde estão presentes amostras de 1000 classes de objetos.

2.8 TÉCNICAS DE PRÉ-PROCESSAMENTO

As imagens de MRI em bruto, para além da heterogeneidade que apresentam, são demasiado grandes para serem diretamente utilizadas na classificação, sendo por isso necessário pré-processá-las e realizar a extração e classificação das características para o diagnóstico de doenças (Li and Liu, 2018). Esta fase de pré-processamento é uma fase fundamental na construção de modelos de DL, pois prepara os dados para que sejam aproveitados da melhor forma pelos modelos.

Antes de mais, convém referir que há diversas abordagens que podem ser seguidas, e cada uma delas leva a pré-processamentos diferentes. As abordagens ao pré-processamento podem ser categorizadas em 4 principais categorias: métodos baseados nos vóxeis das imagens (*voxel-based*), métodos baseados em Regiões de Interesse (ROIs)[16], métodos baseados em *patches*, e métodos que potenciam características da imagem global, isto é, sem considerar estruturas locais dentro das imagens MRI (Esmaeilzadeh et al., 2018).

As abordagens baseadas nos vóxeis são propensas a *overffiting* devido à grande dimensão das imagens, dado que nestas abordagens a entrada dos modelos é a totalidade da imagem. Os métodos baseados em regiões de interesse estão restritos a um número limitado de ROIs em que podem ignorar a informação que se encontra dentro. As abordagens baseadas em *patches* ignoram uma representação global do cérebro e focam-se apenas em *patches* retangulares (ou cúbicos) de tamanho fixo. Enquanto as abordagens que utilizam a totalidade da imagem podem não conseguir identificar mudanças subtis nas estruturas cerebrais finas, as outras abordagens podem perder informação na divisão em regiões de interesse, ou então ignoram a representação do cérebro. Deste modo, é fundamental encontrar uma solução de compromisso entre as representações globais e locais, podendo isso contribuir para uma melhor compreensão da doença.

Mesmo sabendo que o grande poder da DL se deve, acima de tudo, à extração automática de características, existem poucos métodos de DL *end-to-end* que aproveitem indicações locais e globais de imagens MRI para classificar as neuro-imagens nos diferentes grupos de diagnóstico. Segundo Esmaeilzadeh et al. (2018), o não desenvolvimento de modelos de DL *end-to-end* no contexto do diagnóstico da DA, através de imagens MRI, deve-se principalmente às seguintes limitações:

- não existência de amostras etiquetadas, nos conjuntos de dados, suficientes para treinar modelos *end-to-end*;
- as imagens de MRI cerebrais serem imagens 3D com tamanho grande, o que implica um custo computacional elevado;

16 *Regions of interest*, na terminologia inglesa.

- dificuldades na interpretabilidade dos resultados de técnicas de DL *end-to-end* a partir de um ponto de vista neuro-científico.

Para além das abordagens mencionadas, há um conjunto de técnicas de pré-processamento frequentemente aplicadas às imagens MRI. Em primeiro lugar é fundamental que todas as imagens tenham o mesmo tamanho, pois só desta forma podem ser utilizadas como entrada nos modelos de DL. O registo das imagens e a remoção do crânio também são técnicas de pré-processamento bastante comuns neste domínio de aplicação.

O registo das imagens é uma técnica que consiste em adaptar uma determinada imagem a uma outra, que é usada como referência e à qual se dá o nome de *atlas*. O principal objetivo desta técnica passa por fazer com que os mesmo pixeis de todas as imagens representem a mesma informação anatómica. Esta técnica não é aplicada apenas em neuro-imagens, mas também em diversos outros campos da medicina. Em termos dos modelos de DL podemos considerar que esta técnica ajuda na aprendizagem, na medida em que é mais fácil para um modelo identificar uma determinada região das imagens como sendo relevante, se essa informação estiver localizada no mesmo local em todas as imagens.

A extração do crânio[17] é um procedimento bastante comum e consiste, tal como o nome indica, na remoção da informação relativa ao crânio das imagens. O principal objetivo deste procedimento é obter uma imagem final mais limpa e que contenha apenas informação relevante para a tarefa em curso, ou seja, o crânio não é uma parte fundamental no diagnostico da DA, dado que não é um biomarcador, e por isso pode ser removido das imagens.

Para além dos procedimentos comuns no domínio de aplicação desta dissertação, existem outras técnicas mais genéricas que se podem aplicar, como por exemplo a normalização das imagens. A normalização pode ser dividida em duas variantes: normalização de intensidade e normalização espacial. A normalização de intensidade consiste em adaptar a gama dos valores dos pixeis (vóxeis em 3D) de uma imagem de acordo com um determinado critério. Por exemplo, pode reduzir-se os valores ao intervalo [0,1], ao intervalo [-1,1] ou subtrair aos valores a sua média e dividir pelo seu desvio padrão, uma técnica conhecida por *whitening*. A normalização espacial consiste em adaptar os pixeis (ou vóxeis) para representar um determinado espaço (2D ou 3D). Por exemplo, Ding et al. (2019) adaptam imagens 3D de modo a que cada vóxel represente $2mm^3$ de espaço. Neste sentido, o registo de imagens também pode ser considerado uma forma de normalização espacial (Darias Plasencia, 2019).

2.9 TRANSFERÊNCIA DE APRENDIZAGEM

A transferência de aprendizagem assenta na ideia do ajuste fino mencionado anteriormente, e mais não é do que uma técnica que consiste em armazenar o conhecimento adquirido

17 *Skull stripping*, na terminologia inglesa.

na resolução de um problema e reutilizar esse conhecimento para resolver um problema diferente num domínio semelhante (Weiss et al., 2016; Shao et al., 2015). Está técnica tem sido aplicada com sucesso nos métodos de DL.

É de conhecimento geral que os modelos de DL têm um elevado número de parâmetros, e que para treinar estes modelos são necessários muitos dados, caso contrário o modelo irá sofrer de *overfitting*. O problema está exatamente aqui, muitas vezes não temos disponíveis dados necessários para treinar um modelo deste tipo, mas a natureza do problema requer uma solução de DL, como acontece por exemplo no reconhecimento de padrões em imagens. Nestes casos a transferência de aprendizagem pode ser utilizada para gerar características genéricas a partir de um modelo de DL pré-treinado e depois utilizar essas características para construir um novo modelo. Deste modo, os únicos parâmetros para este problema são os necessários para o novo problema. Os modelos pré-treinados que se usam para transferência de aprendizagem são treinados com uma grande quantidade de dados para terem parâmetros fiáveis (Pattanayak, 2017).

Ao classificar imagens através de CNNs com diversas camadas convolucionais, sabemos que as camadas iniciais aprendem a detetar características genéricas, elementares, como curvas e linhas, e as camadas convolucionais mais profundas aprendem a detetar características mais complexas e específicas do conjunto de dados. Suponhamos que temos uma CNN com arquitetura da `AlexNet`, treinada para classificar as 1000 categorias do *ImageNet*. Se agora tivermos um conjunto de dados mais pequeno com menos categorias, mas com imagens semelhantes às do conjunto de dados *ImageNet*, usado para treinar a `AlexNet`, podemos utilizar a mesma rede `AlexNet` até à camada totalmente ligada e treinar a camada de saída para classificar as classes do novo conjunto de dados. Ou seja, podemos manter fixos os pesos da rede até à camada totalmente ligada e só treinamos o modelo para aprender os pesos da camada totalmente ligada antes da saída. O que nos permite fazer isto é o facto de a natureza dos conjuntos de dados ser a mesma, o que faz com que as características aprendidas no modelo pré-treinado através dos diferentes parâmetros são suficientemente boas para o novo problema de classificação, e só precisamos de aprender os pesos da camada totalmente ligada localizada antes da camada de saída. Isto leva a uma enorme redução no número de parâmetros que a rede tem de aprender, para além da redução do *overfitting* que provavelmente iríamos ter dificuldade em contornar devido ao reduzido tamanho do conjunto de dados e ao elevado número de parâmetros a aprender (Pattanayak, 2017).

Quando estamos perante 2 conjuntos de dados de natureza muito diferente, ou seja a forma da distribuição da informação presente nos 2 conjuntos é muito diferente, não podemos utilizar a técnica de transferência de aprendizagem, mas podemos na mesma tirar partido desta técnica de transferência de aprendizagem. Nestes casos podemos usar na mesma o modelo pré-treinado mas fixar apenas os parâmetros das primeiras camadas da rede, responsáveis por detetar características elementares e genéricas, e adicionar algumas

novas camadas que têm que ser treinadas para aprenderem a detetar as características inerentes ao novo conjunto de dados. A camada totalmente ligada e a camada de saída também têm que ser treinadas. Desta forma conseguimos reduzir o número de parâmetros a aprender, nomeadamente nas primeiras camadas.

O TensorFlow e o Keras disponibilizam um conjunto de funcionalidades que permite uma fácil implementação de transferência de aprendizagem recorrendo a modelos pré-treinados.

2.10 NORMALIZAÇÃO DE BATCH

Apesar de existirem diversos métodos que permitem reduzir o *vanishing/exploding* dos gradientes no inicio do treino, essas técnicas não garantem que este problema não volte numa fase mais avançada do treino. Sergey Ioffe e Christian Szegedy, num artigo publicado em 2015, propuseram uma técnica denominada BN para resolver o problema de *vanishing/exploding* dos gradientes (Ioffe and Szegedy, 2015).

De forma bastante resumida, e sem entrar em profundos detalhes matemáticos, esta técnica consiste em adicionar uma operação no modelo imediatamente antes ou depois da função de ativação de cada camada oculta. O que se faz é centrar em torno do zero e normalizar cada entrada para ter desvio padrão 1. No fundo desloca-se e escala-se os valores usando dois novos vetores de parâmetros por camada: um para deslocar e o outro para escalar os valores. Esta operação permite ao modelo aprender a escala e a média ótimas de cada uma das entradas da camada. A fim de centrar em zero e normalizar as entradas, o algoritmo precisa estimar a média e o desvio padrão de cada entrada e fá-lo avaliando a média e o desvio padrão de cada entrada no *mini-batch* atual, e daí surge o nome "normalização de *batch*".

Durante a fase de teste, podemos ter de fazer previsões para instâncias individuais e não para *batches* de instâncias: neste caso, não teremos forma de calcular a média e o desvio padrão de cada entrada. Além disso, mesmo que tenhamos um *batch* de instâncias, este pode ser demasiado pequeno pelo que calcular estatísticas sobre as instâncias do *batch* não seria fiável. Uma solução poderia ir aplicando todo o conjunto de treino e calculando a média e o desvio padrão de cada entrada das camadas BN. Estas médias e desvios padrão globais poderiam então ser utilizadas em vez das médias e desvios padrão calculados ao nível do *batch* para fazer previsões. No entanto é muitas vezes preferível estimar estas estatísticas durante o treino, utilizando uma média móvel das médias e dos desvios padrão. Deste modo é preciso aprender quatro vetores de parâmetros em cada camada normalizada por *batch*: γ (o vetor de escala de saída) e β (o vetor de *offset* de saída) são aprendidos através de retro-propagação[18] regular, μ (o vetor de média global) e σ (o vetor de desvio padrão global) são estimados utilizando uma média móvel exponencial.

18 *Backpropagation*, na terminologia inglesa.

Esta técnica melhorou consideravelmente todas as DNNs que os autores experimentaram, levando a uma enorme melhoria na tarefa de classificação do *ImageNet*. O problema dos *vanishing/exploding* dos gradientes foi fortemente reduzido, ao ponto de poderem utilizar funções de ativação saturantes, como o `tanh`. Para além disso, as redes ficaram muito menos sensíveis à inicialização dos pesos. Esta técnica permite que se utilizem taxas de aprendizagem superiores, acelerando o processo de aprendizagem. Apesar do treino parecer mais lento, uma vez que cada época leva mais tempo com o uso de BN devido ao acréscimo dos cálculos, a verdade é que este facto é contrabalançado pelo facto de a convergência ser muito mais rápida com BN, pelo que serão necessárias menos épocas para se atingir o mesmo desempenho.

A utilização de BN também atua como regulador, reduzindo a necessidade de utilização de outras técnicas de regularização, como por exemplo o *dropout*. Pese as inúmeras vantagens, esta técnica acrescenta complexidade aos modelos, havendo uma penalização no tempo de cálculo, ou seja, a rede neuronal fica mais, especialmente a fazer predições devido aos cálculos adicionais necessários em cada camada. Assim, se existir necessidade de uma predição rápida deve experimentar-se outras técnicas, como formas diferentes de inicialização dos pesos, antes de experimentar BN.

A implementação de BN em Keras é bastante simples, sendo apenas necessário uma camada `BatchNormalization` antes ou depois da função de ativação de cada camada oculta.

2.11 DROPOUT

O *dropout* é uma das técnicas de regularização mais populares para redes neuronais profundas. Foi proposta por Geoffrey Hinton (Hinton et al., 2012) em 2012 e detalhada por Nitish Srivastava (Srivastava et al., 2014) e permite obter uma rede mais robusta que generaliza melhor (Géron, 2019). Esta técnica é bastante simples e faz com que, em cada etapa do treino cada neurónio, quer seja da camada de entrada quer seja camada intermédia, tem uma probabilidade p de ser temporariamente desativado (figura 28), ou seja, será totalmente ignorado durante essa etapa do treino. Na etapa de treino seguinte esses neurónios que foram abandonados podem voltar a estar ativos. O hiperparâmetro p é chamado de taxa de *dropout*. Nas fases de validação e teste os neurónios não se aplica *dropout*.

Existe um detalhe que vale a pena tomar nota. Supondo que temos p=50%, em média na fase de teste um neurónio está ligado ao dobro dos neurónios de entrada do que estava ligado durante o treino. Para compensar este facto precisamos multiplicar os pesos de conexão de entrada de cada neurónio por 0,5 após o treino. Se não o fizermos, cada neurónio irá receber um sinal de entrada total que é aproximadamente duas vezes superior ao que recebia quando a rede foi treinada: Neste caso é pouco provável que tenhamos um bom desempenho. De um modo mais geral, após o treino e em cada ligação de entrada num

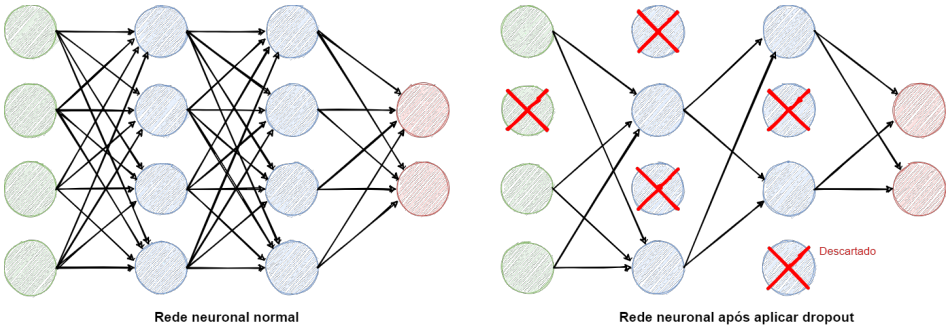


Figura 28: Aplicação de *dropout* numa rede neuronal.

neurónio, multiplicamos o seu peso pela probabilidade de retenção $(1 - p)$. Em alternativa podemos dividir a saída de cada neurónio pela probabilidade de manutenção durante o treino. Estas alternativas são não são perfeitamente equivalentes, mas funcionam igualmente bem.

Para implementar o *dropout* podemos usar a camada `keras.layers.Dropout` que, durante o treino, desativa aleatoriamente algumas entradas, fixando-as em zero, e divide as restantes entradas pela probabilidade de retenção. Esta função recebe apenas um parâmetro que é a taxa de *dropout*. Se constatarmos que um modelo está a incorrer em *overfitting*, significando que o modelo em vez de generalizar está a mimetizar os dados, pode aumentar-se a taxa de *dropout* para lhe diminuir a capacidade de aprendizagem. Inversamente, deve-se tentar diminuir a taxa de *dropout* se o modelo não conseguir aprender as caraterísticas do conjunto de treino (*underfitting*).

2.12 AUMENTO DOS DADOS

As CNNs profundas têm mostrado excelente comportamento em diversas tarefas de visão por computador. No entanto, estas redes estão extremamente dependentes de grandes dados para evitar o *overfitting*. Infelizmente em muitos domínios de aplicação não se tem acesso a grandes quantidades de dados, como é caso da análise de imagens médicas. Consequentemente as redes acabam muitas vezes por sofrer de *overfitting* e são incapazes de generalizar quando lhe fornecemos entradas novas. Para construir modelos de DL úteis o erro de validação deve continuar a diminuir com o erro de treino e o aumento de dados é um método muito poderoso para o conseguir. Os dados aumentados representarão um conjunto mais abrangente de dados, minimizando assim a distância entre os conjuntos de treino e de validação, bem como quaisquer conjuntos de teste futuros (Shorten and Khoshgoftaar, 2019).

Como temos visto, o aumento de dados não é a única técnica desenvolvida para reduzir o *overfitting*. Outras estratégias pensadas para aumentar a capacidade de generalização focam-se na própria arquitetura do modelo. Soluções funcionais como pré-treino (seção 2.6),

transferência de aprendizagem (seção 2.9), normalização de *batch* (seção 2.10) e *dropout* (seção 2.11), foram desenvolvidas para tentar aplicar DL a conjuntos de dados mais pequenos. Outras estratégias que ainda não foram referidas nesta dissertação são a aprendizagem *one-shot* e *zero-shot*, que representam outro paradigma para a construção de modelos com dados extremamente limitados (Palatucci et al., 2009; Xian et al., 2017). De forma muito resumida, *one-shot* é normalmente utilizada em aplicações de reconhecimento facial (Taigman et al., 2014) e uma das suas concretizações passa por usar redes siamesas (Koch et al., 2015), que aprendem uma função de distância entre pares de imagens, uma imagem do par é da pessoa a identificar e a outra imagem do par é de uma pessoa diferente. Deste modo a classificação das imagens de uma pessoa é possível mesmo que a rede só tenha sido treinada com uma ou poucas imagens dessa pessoa. Uma outra abordagem muito popular da aprendizagem *one-shot* é a utilização de redes com memória[19] (Santoro et al., 2016). A aprendizagem *zero-shot* é um paradigma mais extremo, no qual uma rede utiliza *embeddings* de entrada e de saída, fornecidos por modelos como *Word2Vec* (Mikolov et al., 2013) ou *GloVe* (Pennington et al., 2014), para classificar imagens com base em atributos descritivos.

Em contraste com as técnicas acima mencionadas, o aumento de dados aborda o *overfitting* a partir da raiz do problema, o conjunto de dados de treino. O racional por trás desta técnica é pressupor que mais informação pode ser extraída do conjunto de dados original, se ele for aumentado por via de transformações. Este aumento artificial dos dados de treino pode ser categorizado em transformações nos dados (*data wrapping*) ou sobre-amostragem (Shorten and Khoshgoftaar, 2019). *Data wrapping* consiste na transformação das imagens existentes de modo a que a sua etiqueta seja preservada. Nesta categoria encaixam os aumentos com transformações geométricas e de cor, *random erasing*, *adversarial training*, e *neural style transfer*. Os aumentos baseados em sobre-amostragem[20] consistem em criar instâncias sintéticas para serem acrescentadas ao conjunto de dados de treino. Nesta categoria incluem-se a mistura de imagens, *feature space augmentations* e as *Generative Adversarial Networks* (GANs). A utilização destas técnicas não é mutuamente exclusiva, podendo por isso ser utilizadas em conjunto. Por exemplo, às instâncias criadas por uma GAN podem ser aplicados cortes e outras transformações antes de serem adicionadas ao conjunto de dados de treino.

Nalepa et al. (2019) faz uma revisão das técnicas de aumento de dados aplicadas a imagens de MRIs no contexto de deteção de tumores cerebrais. Os autores dividem os métodos de aumento de dados em 4 categorias: transformações *affine* de imagem, transformações elásticas de imagem, transformações de imagem ao nível do pixel e geração artificial de dados. Nas abordagens *affine*, a informação da imagem original passam por diferentes operações como rotação, ampliação, corte, inversão ou translação, de modo a aumentar o número de amostras de treino (Pereira et al., 2016; Liu et al., 2017). Shin et al. (2018) salientaram que tais técnicas tradicionais de aumento de dados produzem fundamentalmente imagens muito

19 *Memory-augmented networks*, na terminologia inglesa.
20 *Oversampling*, na terminologia inglesa.

correlacionadas, pelo que podem oferecer muito poucas melhorias ao processo de treino em DNN e à capacidade de generalização a futuros dados de teste. Além disso ainda podem gerar amostras anatomicamente incorretas, como acontece por exemplo quando se aplica uma rotação a uma imagem MRIs. No entanto, as transformações de imagem *affine* são fáceis de implementar, tanto em 2D como em 3D, são bastante flexíveis (devido aos seus hiperparâmetros) e são amplamente aplicadas nos trabalhos mencionados na literatura.

Os algoritmos de aumento de dados baseados em transformações elásticas introduzem variações de forma. Podem trazer muito ruído e danos ao conjunto de treino, em determinados domínios de aplicação. Ou seja, a sua utilização deve ser limitada a certos domínios de aplicação, sendo as imagens MRIs um dos casos em que não se devem usar.

Existem também técnicas de aumento de imagem que não alteram a forma geométrica das imagens, mas alteram outras caraterísticas como a intensidade dos píxeis, tanto localmente como em toda a imagem. Estas operações podem ser especialmente úteis na análise de imagens médicas, onde diferentes imagens de treino são adquiridas em diferentes locais e utilizando diferentes scanners, podendo assim ser intrinsecamente heterogéneas na intensidade dos píxeis (Nalepa et al., 2019). Nas alterações ao nível do píxel, a intensidades dos píxeis é normalmente modificada utilizando ruído Gaussiano aleatório ou de média zero. Outras alterações ao nível de píxeis incluem o deslocamento e a escala dos valores de intensidade, de onde resulta alteração do brilho da imagem, a aplicação de correção gama, aumentar o contraste ou esbater a imagem (Galdran et al., 2017).

Para atenuar os problemas relacionados com as abordagens de aumento de dados básicos, onde se inclui o problema da geração de amostras demasiado correlacionadas, na literatura sugerem-se várias abordagens de geração de dados artificiais. Por exemplo as GANs, originalmente introduzidas em Goodfellow et al. (2014), estão a ser aplicada no aumento de conjuntos de dados médicos (Han et al., 2019; Shorten and Khoshgoftaar, 2019). Uma GAN é treinada de modo a que o módulo gerador produza novas amostras que o módulo discriminador não consiga distinguir das amostras reais.

Na sua investigação, Sundaram and Hulkund (2021) avaliam a utilização do aumento de dados com base em GANs para expandir artificialmente o conjunto de dados CheXpert das radiografias torácicas e comparam o desempenho com o aumento de dados tradicional, ou seja, investigam se a incorporação de dados gerados pelas GANs em dados de treino como técnica de aumento de dados pode melhorar a eficácia das DNNs no diagnóstico de doenças pulmonares a partir de radiografias torácicas. Embora o aumento de dados utilizando uma GAN exija uma rede adicional treinada, em oposição às técnicas de aumento de dados padrão, o trabalho realizado demonstra que esta técnica conduz a melhorias de desempenho que são particularmente importantes na tomada de decisões clínicas de alto risco. Os autores utilizam uma GAN condicional, com estruturas espelhadas para o gerador e discriminador, que foi progressivamente pré-treinada em radiografias do tórax do conjunto

de dados CheXpert e utilizam a mesma rede para gerar imagens sintéticas de patologias pouco representadas, corrigindo assim os desequilíbrios nas classes presentes no conjunto de dados de treino. As conclusões indicam que o aumento de dados com base em GANs pode ser uma ferramenta eficaz para corrigir conjuntos de dados médicos com desequilíbrio nas classes. Este estudo pressupõe ter acesso a uma GAN que tenha sido pré-treinada num conjunto de dados de grande dimensão. Como trabalho futuro os autores indicam a possibilidade de investigar se as vantagens do aumento de dados baseado em GANs também existem quando se utilizam GANs treinadas em conjuntos de dados mais pequenos.

Para dar uma representação visual do processo de aumento de dados, a figura 29 mostra a geração de novas imagens a partir da imagem 2D mais à esquerda ma figura. É possível observar a aplicação de transformações simples como rotação, mudança de intensidade dos píxeis, cortes, assim como uma combinação destas.

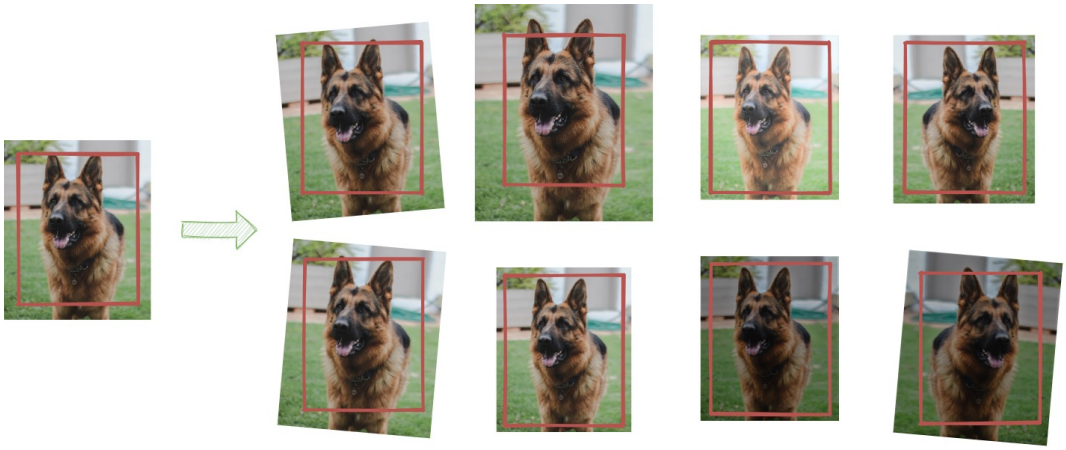


Figura 29: Exemplos do aumento de dados.

2.13 O problema, os seus desafios e as limitações

A aplicação da IA no domínio médico enfrenta uma série de desafios. A grande maioria destes desafios relaciona-se com os dados, que são a base fundamental para a construção de modelos precisos. Há também um conjunto de limitações de tempo e de recurso de *hardware* que se pode ter que enfrentar. Para além disto, existem também desafios éticos e filosóficos.

Como se tem vindo a afirmar ao longo deste documento, o principal intento desta dissertação passa por detetar a DA recorrendo a DNNs, ou seja, recorrendo a métodos de DL. É de conhecimento geral que um dos requisitos para o bom desempenho deste tipo de métodos é ter disponível uma enorme quantidade de dados para treinar os modelos. E aqui reside o grande desafio no desenvolvimento desta dissertação. O primeiro grande problema relacionado com os dados é a sua disponibilidade. Como referido na secção 2.3, uma das razões que levaram ao aumento da aplicação de IA no domínio médico é a aproximação

desta área a *Big Data*, à medida que a quantidade de dados aumenta enormemente. Embora isto seja totalmente verdade, também devemos ter em conta que, mesmo com o crescimento que tem existido, estamos muito longe de ter informação necessária para alcançar avanços semelhantes aos que estão a ser obtidos com imagens naturais. Por exemplo, o conjunto de dados Imagenet inclui mais de 14 milhões de imagens, enquanto os conjuntos de dados de neuro-imagens contam apenas com algumas centenas de imagens de MRI e de PET. Frequentemente a escassez de dados para treino não é um problema de falta de dados em absoluto, mas sim a não existência de dados etiquetados. Mas sem dados etiquetados, as abordagens supervisionadas não podem ser aplicadas (Darias Plasencia, 2019). A pouca disponibilidade de dados conduz muitas vezes ao *overffiting* dos modelos.

Uma das soluções propostas para este problema é a extração de fragmentos aleatórios da imagem (Gupta et al., 2013; Payan and Montana, 2015; Suk et al., 2014), sendo que alguns autores defendem que esta solução se aproxima da forma como os radiologistas analisam as imagens. Ou seja, os radiologistas analisam apenas uma parte das imagens. No entanto, esta solução tem um problema associado: nem todos os fragmentos de uma imagem, pertencente a uma determinada classe, são representativos dessa mesma classe. O aumento de dados é uma outra solução para combater este problema da disponibilidade dos dados mas, como visto na seção 2.12, pode trazer apenas trazer informação redundante ao conjunto de dados. Por fim, a transferência de aprendizagem (seção 2.9) pode também ser vista como uma medida para lidar com a falta de dados.

Um outro problema relacionado com os dados é o desequilíbrio das classes, que normalmente surge na forma de sub-representação da classe positiva nos conjuntos de dados. Isto não surpreende porque é mais fácil encontrar informação de pacientes não doentes do que de pacientes doentes, devido à tendência dos pacientes não partilharem a a informação dos seus exames, especialmente se forem doenças raras ou de natureza mais sensível. Para piorar ainda mais a situação, Greenspan et al. (2016) afirma que a classe negativa é muitas vezes fortemente correlacionada, enquanto que existe muita variação na classe positiva. Nas suas experiências, Mazurowski et al. (2008) descobriram que o desequilíbrio nas classe afeta os resultados de forma negativa, mesmo que os desequilíbrios sejam pequenos. Sugerem ainda que quase nunca é benéfico tentar resolver este problema removendo amostras da classe sobre-representada, uma técnica denominada por sub-amostragem[21]. Enquanto que por vezes a duplicação de amostras da classe sub-representada, técnica denominada por sobre-amostragem, pode melhorar ligeiramente os resultados. Como induz apenas pequenas melhorias, que quase são insignificantes, a sobre-amostragem não é muito mencionada na literatura.

Ainda relacionado com os dados, não sob a forma de problema mas sim como uma opção que se pode tomar, é a combinação de informação de tipos diferentes. São vários os trabalhos

21 *Undersampling*, na terminologia inglesa.

onde se complementam as imagens médicas com outros dados categóricos e numéricos. Por exemplo, num sistema de diagnóstico da DA podemos combinar uma imagem de uma MRI com a idade e o género do paciente. Na prática isto não funciona bem (Darias Plasencia, 2019). Embora em alguns casos possa haver uma melhoria, a quantidade de informação presente nas imagens é tão grande em comparação com os campos individuais que estes últimos se tornam irrelevantes (Litjens et al., 2017a). As DNNs tendem a concentrar-se em características extraídas de imagens, dando pesos muito mais baixos a informação adicional.

Há ainda uma decisão relacionada com os dados que tem que se tomar, que tipo de imagens médicas serão utilizadas. Geralmente nos sistemas CADx da DA utilizam-se imagens de MRI estrutural. Há ainda quem utilize imagens PET 18F-FDG (Ding et al., 2019), bem como a combinação de imagens de MRI estrutural com imagens PET e ainda a combinação com medições do líquido cefalorraquidiano (CSF) (Suk and Shen, 2013). O tipo menos utilizada é a MRI funcional, utilizada por Sarraf et al. (2016) em combinação com MRI estrutural. Existem também diferentes formas de lidar com estas estruturas.

Podem também identificar-se questões filosóficas e éticas no tema desta dissertação. As ECEs contêm informação muito sensível sobre os pacientes, e esta é uma realidade com que os médicos lidam quotidianamente e que torna a confidencialidade entre médico e paciente uma questão fundamental. Ademais, com o crescente debate nos meios de comunicação social sobre a importância da privacidade dos dados das pessoas, surgiram na União Europeia leis como o Regulamento Geral de Proteção de Dados (RGPD), que implicam pesadas multas se os dados privados não forem tratados adequadamente, dando um maior nível de controlo aos indivíduos sobre a forma como a sua informação é tratada pelas empresas e instituições (Darias Plasencia, 2019). Tudo isto limita a quantidade de dados acessíveis. Para construir bases de dados de imagens médicas é preciso obter o consentimento dos indivíduos cujas informações estão a ser partilhadas. Estas condições acentuam o problema de desequilíbrio das classes, uma vez que uma pessoa doente estará menos disposta a partilhar a sua história médica.

Outra questão que contribui para que este modelos ainda não tenham uma plena integração na prática clínica, é a falta de confiança nos sistemas de IA. Como referido na seção 2.2, ainda existe um grande desconhecimento entre a população sobre o que é a IA ou como ela realmente funciona. Os cenários apocalípticos apresentados em filmes e outros meios de comunicação, bem como as declarações de certas figuras públicas, acentuam esta desconfiança. Como resultado, uma grande parte da população está relutante à incursão destes sistemas em aspetos relevantes da vida, tais como a medicina. Este é um problema que não só afeta os pacientes, mas também os próprios médicos. Isto deve-se em grande parte ao facto destes modelos não terem mecanismos para explicar as decisões tomadas.

O diagnóstico de um paciente é uma questão particularmente sensível e que pode ter consequências muito graves se forem cometidos erros. Por conseguinte, um médico não

pode tomar uma decisão sobre este assunto de ânimo leve, contando simplesmente com os resultados obtidos por um modelo. É necessária uma explicação clara da razão pela qual determinada decisão é tomada, especialmente quando o diagnóstico feito pelo médico entra em conflito com o resultado determinado pelo modelo. Por forma a atenuar este problema, Ding et al. (2019) criaram um sistema que calcula os gradientes para a classe AD e cria um mapa de saliência que permite visualizar as áreas das imagens que são consideradas importantes pela rede neural. Apesar de não ser uma justificação explícita da decisão tomada pelo modelo, os mapas de saliência ajudam aperceber quais as zonas da imagem foram consideradas relevantes para a tomada de decisão.

Normalmente os modelos apresentados pelos investigadores são dedicados a uma doença específica, no máximo têm de lidar com diferentes fases da doença. No caso da DA é frequente considerar as 3 classes que têm vindo a ser referidas: AD, MCI, NC. Isto simplifica demasiado o problema real do diagnóstico de um paciente pois, quando uma deficiência cognitiva é detetada há vários tipos de demência candidatas a ser a verdadeira doença, e não apenas a DA. Portanto, com vista a colocar um sistemas destes em produção, seria necessário treinar um modelo para detetar uma gama mais vasta de doenças. Neste contexto encontraríamos novamente o problema da quantidade limitada de dados rotulados, pois seria necessário um aumento significativo dos dados rotulados para cada nova classe ou doença. Para além disso, há um outro problema relacionado com as próprias características da DA. Esse problema centra-se no facto desta doença se manifestar na forma de um espectro contínuo, ou seja, há um agravamento dos sintomas e dos indicadores desta doença de forma contínua, o que causa maior dificuldade na distinção das diferentes fases da mesma.

Por fim, um dos grandes desafios aqui colocados são as limitações de *hardware* e de otimização dos hiperparâmetros. No que toca ao poder de processamento, não havia equipamento demasiado potente disponível. Foi disponibilizada uma máquina pelo orientador desta dissertação, mas a mesma tinha que ser partilhada por outros investigadores. Por isso foi tomada a decisão de usar o computador pessoal, o qual possui uma GPU de gama média, mas isso já representa uma vantagem para treinar as DNNs. O computador utilizado possui uma GeForce RTX 2070 com 8GB de memória VRAM GDDR6. Daqui surgiu a dificuldade de otimizar os hiperparâmetros, pois não havia tempo e poder de processamento suficientes para se realizar uma pesquisa dos melhores hiperparâmetros para cada modelo.

2.14 Métricas de Avaliação de Desempenho

Um dos aspetos mais importantes no desenvolvimento de algoritmos de DL é saber como avaliar o seu desempenho na resolução em causa. Avaliar um classificador é mais complexo do que avaliar um regressor, pois num problema de classificação não basta olhar para a

frequência com que um modelo faz uma previsão correta, pode ser necessário fazer outro tipo de análise.

2.14.1 *Matriz de confusão*

A matriz que confusão é uma tabela que representa as previsões e os resultados reais (etiquetas) de um classificador, e que permite avaliar o desempenho de um classificador (figura 30).

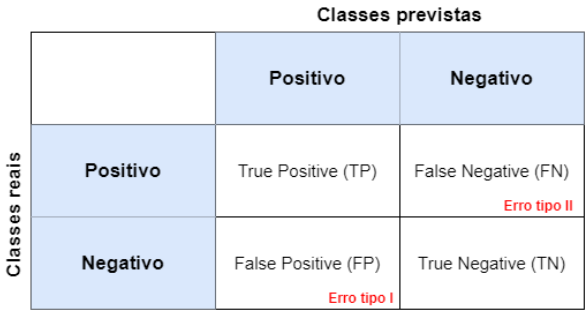


Figura 30: Matriz de confusão num problema de classificação binária.

Na matriz de confusão podemos ver 4 valores numéricos:

- ***True positives***: previsões positivas com etiqueta positiva
- ***False positives***: previsões positivas com etiqueta negativa, também conhecidas como "erro de tipo I"
- ***True negatives***: previsões negativas com etiqueta negativa
- ***False negatives***: previsões negativas com etiqueta positiva, também conhecidas como "erro de tipo II".

Através destas contagens podemos calcular um conjunto de métricas que nos permitem fazer uma análise mais detalhada sobre o desempenho de um modelo.

Acurácia

A acurácia corresponde à percentagem total de instâncias bem classificadas (equação 1).

$$acuracia = \frac{TP + TN}{TP + FP + FN + TN} \tag{1}$$

Esta métrica pode ser enganosa a avaliar a qualidade de um modelo, principalmente quando estamos perante um conjunto de dados bastante desequilibrado. Isto acontece porque se o modelo classificar todas as amostras como sendo da classe mais representada,

o modelo estará a classificar corretamente uma grande percentagem das amostras, o que resulta numa acurácia elevada. Contudo esta métrica é enganosa, porque neste caso o modelo não consegue classificar corretamente as instâncias da classe menos representada, e para isso não precisa aprender nada.

Sensibilidade versus Especificidade

A sensibilidade e a especificidade são duas métricas distintas que permitem tirar algumas conclusões acerca de classificadores binários. A sensibilidade, também denominada por *recall*, refere-se à taxa de verdadeiros positivos, ou seja, mede a quantidade de vezes que o modelo classifica uma instância como positiva e a sua etiqueta é realmente positiva. Por exemplo, classificar um paciente como tendo DA quando o doente está realmente com essa condição. Esta métrica quantifica o quão bem o modelo evita os falsos negativos e pode ser calculada com a equação 2.

$$sensibilidade = \frac{TP}{TP + FN} \tag{2}$$

A especificidade refere-se à taxa de verdadeiros negativos, ou seja, classificações negativas que foram bem classificadas, ou seja, que possuem etiqueta negativa. Por exemplo, classificar um paciente como não tendo DA e o doente não ter na realidade a doença. A especificidade quantifica a forma como o modelo evita falsos positivos e pode ser calculada com a equação 3.

$$especificidade = \frac{TN}{TN + FP} \tag{3}$$

São diversos os casos em que estas duas métricas são fundamentais e devem der avaliadas. Por exemplo, um modelo que detete uma doença grave, no qual uma falha no diagnóstico tem um elevado custo caso o paciente esteja doente e o modelo diga o contrário. O ideal seria ter um modelo com especificidade igual a 100%, correspondendo a todos os doentes serem detetados, e com especificidade igual a 100%, em que ninguém que não esteja doente é classificado como estando doente.

Precisão

Precisão é uma métrica que muitas vezes é confundida com a acurácia e mede a exatidão das previsões positivas através da equação 4. Uma modelo pode apresentar uma boa acurácia e uma boa precisão, pode ter uma boa acurácia e uma baixa precisão, pode ter uma baixa acurácia e uma boa precisão, e pode ter uma baixa acurácia e uma baixa precisão.

$$precisao = \frac{TP}{(TP + FP)} \tag{4}$$

F1-score

A pontuação F1, também conhecida como *F-score* ou *F-measure*, permite avaliar o desempenho dos modelos de classificação binária. Esta métrica consiste na média harmónica entre o *recall* e a precisão, reunindo estas duas métricas numa só, e pode ser calculada com a equação 5. Enquanto a média tradicional trata os valores todos da mesma forma, a média harmónica dá um maior peso a valores baixos. Em consequência disso os classificadores terão um F1 alto somente quando a precisão e o *recall* forem ambos altos.

$$F_1 = \frac{2}{\frac{1}{precisao} + \frac{1}{recall}} = 2 \times \frac{precisao \times recall}{precisao + recall} = \frac{TP}{TP + \frac{FN+FP}{2}} \tag{5}$$

O valor de F1 pode variar entre 0 e 1, onde 0 é a pior pontuação e 1 é a pontuação ideal e é muitas vezes usada em modelos ML como a pontuação global de desempenho dos modelos. Esta métrica favorece classificadores com precisão e *recall* semelhantes, o que pode ser algo que não queremos. Em alguns casos podemos querer prestar mais atenção à precisão e em outros ao *recall*. Utilizando o exemplo do livro de Géron (2019), que nos permite bem perceber de forma clara os casos onde necessitamos de uma boa precisão e os casos onde precisamos de um bom *recall*. Se treinámos um classificador para detetar vídeos que são seguros para as crianças, provavelmente preferimos um classificador que rejeite muitos vídeos seguros (baixo *recall*) mas acerte nos vídeos que não são seguros (alta precisão), em vez de um classificador que tem uma *recall* muito superior mas falha na classificação de vídeos não seguros. Por outro lado, suponhamos que treinamos um classificador para detetar ladrões de lojas em imagens de vigilância: provavelmente nos importamos se o classificador tiver apenas 30% de precisão, desde que tenha 99% de *recall*. Claro que os seguranças vão receber alguns falsos alertas, mas quase todos os ladrões de lojas vão ser apanhados. Infelizmente, não se pode ter as duas coisas: o aumento da precisão reduz o *recall*, e vice-versa. A isto chama-se compromisso entre precisão e *recall*.

As métricas apresentadas, apesar de serem muito usadas em diversos contextos, não são as ideais para o problema atual, pois adequam-se a problemas maioritariamente binários ou são enganosas. O problema desta investigação é multi classe, pelo que têm que se encontrar outras métricas.

2.14.2 *A curva ROC*

A curva *Receiver Operating Characteristic* (ROC) é uma outra ferramenta frequentemente usada para avaliar classificadores. É bastante semelhante à curva precisão/*recall*, mas em vez de representar a precisão vs *recall*, a curva ROC representa a taxa de verdadeiros positivos (TPR), outro nome dado ao *recall*, contra a taxa de falsos positivos (FPR), que representa as instâncias negativas classificadas incorretamente como positivas. O FPR é igual a um

menos a taxa de verdadeiros negativos (TNR), que são as instâncias negativas classificadas corretamente, também denominado de especificidade. Desta forma, curva ROC representa a sensibilidade (*recall*) versus $1 - especificidade$. Um exemplo desta curva pode ser visto na figura 31.

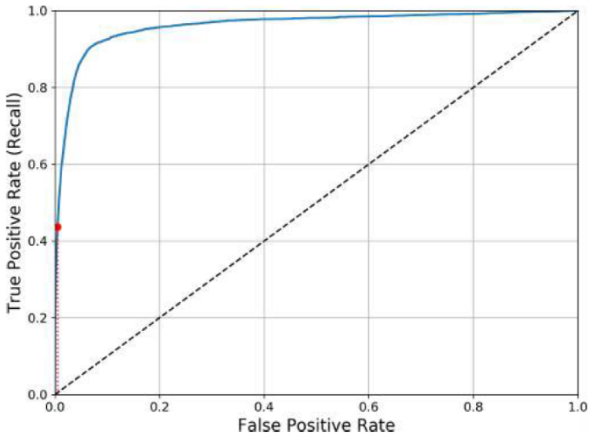


Figura 31: Curva ROC. Origem: (Géron, 2019).

Mais uma vez estamos perante um compromisso: quanto maior for o *recall* (TPR), maior será o número de falsos positivos (FPR) que o classificador produz. Na figura 31 a linha tracejada representa a curva ROC de um classificador completamente aleatório. Uma curva de um bom classificador mantém-se afastada dessa linha e o mais aproximado do canto superior esquerdo (curva a verde na figura 31).

Relacionado com esta curva, e para comparar diversos classificadores, existe ainda a métrica *Area Under the Curve* (AUC). Um classificador perfeito tem uma AUC igual a 1 enquanto que um classificador puramente aleatório tem uma AUC igual a $0,5$. A AUC pode ser calculada para cada classe que o modelo está a classificar e por isso mesmo é bastante utilizada em problemas como o desta dissertação.

Para terminar esta seção é conveniente referir que o contexto do problema desempenha um papel fundamental na forma como os modelos devem ser avaliados, ou seja, as métricas e pontuações a usar para avaliar os modelos podem ser diferentes quando o contexto não é o mesmo.

2.15 Trabalho relacionado

São diversos os trabalhos desenvolvidos na deteção da DA utilizando métodos de DL. É de conhecimento geral que a DA tem a si associada uma diminuição brutal da qualidade de vida dos pacientes que dela sofrem, por isso, vários estudos têm sido realizados nesta área. Esta secção faz um pequeno resumo de alguns desses trabalhos.

Esmaeilzadeh et al. (2018) utilizaram uma CNN 3D bastante simples para diagnosticar DA e DCL. Os dados utilizados neste estudo foram imagens de MRI estruturais do conjunto de dados ADNI. Para o pré-processamento dos dados, os autores optaram por fazer as mínimas alterações possíveis e utilizaram um método simples para aumento de dados, sendo que a abordagem deste pré-processamento se enquadra na técnica *voxel-based*, que faz com que sejam tomados em conta todos os vóxeis do cérebro. Segundo os autores, o uso desta representação detalhada das imagens de MRI baseada em vóxeis elimina quaisquer julgamentos *a priori* para a escolha de ROIs ou *patches* (as outras técnicas de pré-processamento de dados), preferindo ter em conta todo o cérebro. Para evitar o potencial *overfitting* causado pela grande dimensão da entrada, os autores criaram cuidadosamente a arquitetura do modelo de aprendizagem de forma sistemática, não tendo utilizado arquiteturas padrão de visão por computador. Com isto, os autores conseguiram atingir uma *accuracy* no diagnóstico da DA de 94.1%. Posteriormente, os autores utilizaram transferência de aprendizagem para fazer o diagnóstico da DCL, juntamente com as duas outras classes, num ambiente de classificação de três classes (AD, MCI, NC), que mostrou ter um desempenho superior a com outros métodos, com uma *accuracy* de 61.1%. Por fim, é apresentado um método simples para identificar biomarcadores da DA na MRI, observando as diferentes regiões do cérebro que contribuem para a correta classificação. Os biomarcadores identificadores foram de acordo com a literatura.

Lu et al. (2018) propuseram uma técnica distinta para identificar indivíduos em risco de desenvolver a DA: um novo sistema baseado em DL para discriminar indivíduos com DA utilizando uma rede neuronal multimodal e multiescala profunda. Esta abordagem combina informação multimodal a partir de imagens de MRI e FDG-PET em múltiplas escalas dentro de uma estrutura de DNN. Para isso, os autores propõem um pré-processamento que extrai características a escalas estruturais *coarse-to-fine*. Isto é conseguido através da segmentação da imagem estrutural em compartimentos cortical e sub-cortical de matéria cinzenta, e subdividindo cada um deles em *patches* de tamanho hierárquico. De seguida, são extraídas características de cada *patch* fazendo uma média dentro do *patch* e utilizando estas características multi-escala tiradas de múltiplas modalidades numa modelo de aprendizagem profunda. Ao contrário das abordagens simples de amostragem, que podem levar à perda de informação, a abordagem multi-escala descrita preserva a informação estrutural e metabólica em múltiplas escalas e pode potencialmente melhorar a *accuracy* da classificação para esta tarefa de diagnóstico. Para validar esta metodologia inovadora, os autores realizaram experiências de *cross-validation* com todos os dados ADNI disponíveis (sujeitos que incluem tanto uma MRI estrutural T1 como uma imagem do metabolismo FDG-PET). Os resultados procurados pelos autores são: (1) deteção de pacientes de controlo e com DCL que se convertem em DA em função dos anos de conversão, e (2) pacientes de controlo e pacientes com DA, para cada modalidade em separado e combinadas. O método proposto proporciona

uma *accuracy* de 82,4% na identificação dos indivíduos com DCL que se irá converter em DA 3 anos antes da conversão e uma *accuracy* de 86,4% quando a conversão se dá dentro de 1 a 3 anos, uma *sensitivity* de 94,23% na classificação de indivíduos com diagnóstico clínico da DA, e uma *specificity* de de 86,3% na classificação de controlos não dementes. De forma resumida, a DNN multi-escala e multi-modal foi concebida para incorporar múltiplas escalas de informação de múltiplas regiões na matéria cinzenta do cérebro retirada de múltiplas modalidades (T1-MRI e FDG-PET) e mostrou resultados muito satisfatórios. Importante referir que, a rede foi pré-treinada utilizando aprendizagem não supervisionada, com *stacked-autoencoder* (SAE).

Qiu et al. (2020) criou uma estratégia de DL interpretável que traça as assinaturas únicas da DA partir de entradas multimodais de MRI, idade, sexo, e *Mini-Mental State Examination score*. A arquitetura criada liga uma rede totalmente convolucional, que constrói mapas de alta resolução da probabilidade de doença a partir da estrutura cerebral local, a um *perceptron* de várias camadas e gera uma visualização precisa e intuitiva do risco individual de DA. O modelo foi treinado utilizando o diagnóstico clínico da DA e sujeitos cognitivamente normais do conjunto de dados ADNI (n = 417) e validado em três coortes independentes: o *Australian Imaging, Biomarker and Lifestyle Flagship Study of Ageing* (AIBL) (n = 382), o *Framingham Heart Study* (n = 102), e o *National Alzheimer's Coordinating Center* (NACC) (n = 582). No que toca ao desempenho, este modelo multimodal, foi consistente nos diversos conjuntos de dados, tendo atingido valores médios de AUC de 0,996, 0,974, 0,876 e 0,954 para os conjunto de dados ADNI, AIBL, *Framingham Heart Study* e NACC, respetivamente. Para além disto, os autores defendem que a sua abordagem excedeu o desempenho de diagnóstico de uma equipa multi-institucional de neurologistas praticantes (n = 11), e que as regiões cerebrais de alto risco previstas pelo modelo, seguem de perto os resultados histopatológicos pós-morte.

Li and Liu (2018), motivado pelo sucesso da DL na classificação de imagens, propõe no seu artigo um método de classificação baseado numa CNN densa (*DenseNets*) para aprender várias características locais das imagens de MRI cerebral, que serão combinadas para a classificação da DA. Em primeiro lugar, os autores dividem toda a imagem do cérebro em diferentes regiões e extraem um certo número de *patches* 3D de cada região. Se seguida, os *patches* de cada região são agrupados em distintos *clusters* utilizando o método de agrupamento *K-Means*. Depois disso foi construída uma *DenseNet* para aprender as características dos *patches* de cada *cluster*. As características com os *clusters* discriminatórios de cada região são agrupadas (*emsembled*) para classificação. Por fim, os resultados da classificação de diferentes regiões locais são combinados para melhorar a classificação final da imagem. O método proposto pode gradualmente aprender as características de imagens de MRI a partir dos *patches* locais até ao nível de imagem global para a tarefa de classificação. Para pré-processamento dos dados, não é necessário métodos rígidos de

registo e segmentação das MRIs. O método desenvolvido é avaliado usando *T1-weighted* MRIs de 831 sujeitos, incluindo 199 doentes com DA, 403 sujeitos com DCL e 229 sujeitos de controlo normal (NC) dp conjunto de dados ADNI. Resultados experimentais mostram que o método proposto atinge uma *accuracy* de 89,5% e uma AUC de 92,4% para a classificação AD vs. NC, e uma *accuracy* de 73,8% e uma AUC de 77,5% para a classificação MCI vs. NC, demonstrando os resultados promissores da classificação.

Islam and Zhang (2018), desenvolveu um modelo que, ao contrário da maioria dos modelos que fazem apenas classificação binária, é capaz de identificar diferentes estágios da DA, utilizando o conjunto de dados *Open Access Series of Imaging Studies* (OASIS), que contém 416 MRIs estruturais. Os autores desenvolveram uma CNN que aprende as características diretamente das MRIs estruturais, eliminando assim a necessidade de criação de características de forma manual. Para pré-processamento, os autores cortam partes da imagem das MRIs e utilizam aumento de dados. No fundo, o modelo desenvolvido por estes autores não é apenas uma CNN, mas sim um agrupamento (*ensemble*) de 3 CNNs. São testados diversos modelos para serem comparados dos resultados dos mesmos com o modelo proposto pelos autores. A comparação é realizada com recurso a diversas métricas de avaliação de desempenho. O principal problema com este estudo é o facto de os dados estarem muito desequilibrados no que toca ao número de instâncias existentes para cada classe.

Basaia et al. (2019), utiliza MRIs estruturais do conjunto de dados ADNI e de um conjunto de dados recolhido pelos autores, para fazer seis classificações binárias distintas (DA vs HC (controlos saudáveis), *converter* DCL (c-MCI) vs HC, *stable* DCL (s-MCI) vs HC, DA vs c-MCI, AD vs s-MCI, c-MCI vs s-MCI.), recorrendo a uma CNN 3D. Após recolhidos os conjuntos de dados, as MRIs passaram por um complexo pré-processamento. Os autores decidiram optar por desenvolver uma CNN 3D devido à natureza volumétrica das MRIs. A entrada do modelo são as imagens 3D *T1-weighted* MRI normalizadas e a saída é a previsão ao grupo ao qual pertence a instância avaliada. A arquitetura desenvolvida contem 12 camadas convolucionais consecutivas (2 camadas com 50 *kernels* de tamanho $5 \times 5 \times 5$ com *strides* de 1 e 2, seguidas por 10 camadas com número de *kernels* a variar de 100 a 1600, com tamanho tamanho $3 \times 3 \times 3$ e com *strides* a alternar entre 1 e 2), usa a ReLU como camada de ativação, possui uma camada completamente ligada, e uma camada de saída que usa a regressão logística. Esta CNN desenvolvida difere das CNNs tradicionais na medida em que as camadas de *pooling* máximo são substituídas por camadas convolucionais com *stride* 2. Para cada classificação são feitos essencialmente 3 passos: (1) treino, (2) validação, e (3) teste. Os dados foram divididos aleatoriamente, sendo que 90% dos dados foram usados para treino e validação e os restantes 10% foram usados para teste. Após esta divisão dos dados, os autores aplicaram técnicas de aumento de dados nos dados de treino e validação, por forma a aumentar artificialmente o número de imagens e consequentemente prevenir o

overfitting. Dentro deste conjunto de dados aumentados, 90% foram usados para treino e 10% para validação, sendo que a validação foi feita recorrendo ao método *cross validation* com 10 *folds*. Para melhorar a performance os classificadores foi utilizada a técnica de transferência de aprendizagem, ou seja, os pesos da CNN usada para classificação de DA vs HC foram transferidos para os restantes classificadores e usados como pesos iniciais, sendo que esta técnica permite reduzir o tempo de treino e aumenta e eficiência da rede. Por fim, para avaliar a performance dos diversos classificadores foram usadas diversas métricas. De forma geral, os resultados obtidos são bastante satisfatórios.

3

CONJUNTO DE DADOS

É nesta secção que se dá início à fase descritiva desta dissertação. É aqui descrito o processo de obtenção e pré-processamento dos dados, e de seguida, será descrito o procedimento levado a cabo para implementar os diversos modelos.

3.1 OBTENÇÃO DE DADOS

Como tem vindo a ser referido ao longo desta dissertação foi usado o conjunto de dados público ADNI. O acesso a estes dados é gratuito, mas é necessário requerer acesso aos mesmos em http://adni.loni.usc.edu. Liderada pelo investigador principal Dr. Michael W. Weiner, a ADNI surgiu em 2004 como uma associação público-privada e, dentro dos seus diversos objetivos, pode-se destacar a deteção da DA o mais cedo possível (pré-demência) e identificar formas de acompanhar a progressão da doença com biomarcadores. Informação atualizada sobre esta associação pode ser encontrada em http://adni.loni.usc.edu/about/.

Existem 4 fases distintas do ADNI: ADNI-1, ADNI-GO, ADNI-2 e ADNI-3. A fase ADNI-1, iniciada em 2004 e com duração de 5 anos, teve como principal objetivo desenvolver biomarcadores para funcionarem como medidas de avaliação dos resultados obtidos em ensaios clínicos e contou com a participação de 200 idosos de controlo, 400 pacientes classificados como estando numa fase de DCL e 200 pacientes com DA. A fase ADNI-GO começou em setembro de 2009 e teve duração de 2 anos, o principal objetivo desta fase era examinar os biomarcadores em fases muito iniciais da doença e contou com os pacientes da fase ADNI-1 e mais 200 pacientes em fases iniciais de DCL. A fase ADNI-2 teve como principal objetivo desenvolver biomarcadores para medir o declínio cognitivo. Começou em setembro de 2011, teve duração de 5 anos e contou com os pacientes das fases anteriores, mais 150 novos idosos de controlo, 100 pacientes em fases iniciais de DCL, 150 pacientes em fases avançadas de DCL e 150 pacientes com DA. Por fim, a fase ADNI-3 consistiu no estudo da utilização de PET e técnicas de imagiologia funcional em ensaios clínicos, começou em setembro de 2016, teve uma duração de 5 anos e contou com os pacientes das fases anteriores, mais 133 idosos de controlo, 151 pacientes com DCL e 87 pacientes com DA.

Do conjunto de dados foram recolhidas imagens originais 3D de MRI estruturais ponderadas em T1. Mais concretamente foram recolhidas imagens das fases ADNI1, ADNI-GO e ADNI2. A decisão por recolher as imagens originais passou, acima de tudo, por evitar recolher as imagens disponíveis neste conjunto de dados que tinham já sido submetidas a diversas e distintas técnicas de pré-processamento. Em conjunto, estas fases contam com um total de 1450 participantes distintos.

Nestas mesmas fases, foram recolhidas imagens de ressonância magnética dos vários participantes, em datas distintas e com algum espaçamento temporal. Apesar do distanciamento temporal, foi decidido tomar em conta apenas uma imagem por cada participante. Esta decisão foi tomada porque se percebeu que grande parte dos estudos utiliza diversas imagens do mesmo paciente, e isso acaba por influenciar os resultados obtidos, permitindo tirar conclusões mais positivas do que a realidade. Há imagens dos mesmos pacientes recolhidas com o espaçamento temporal de 2 meses, e isso pode gerar imagens muito semelhantes, pois nesses 2 meses pode não existir grandes alterações nos biomarcadores. Assim sendo, colocar diversas imagens do mesmo paciente foi considerada uma prática que não é totalmente correta e que pode acabar por produzir resultados que não são os mais realistas. Desta forma, foi utilizada a ferramenta de pesquisa de imagens disponível na base de dados do ADNI, e foram recolhidas todas as imagens originais de todos os pacientes e depois selecionada aleatoriamente apenas uma imagem por paciente.

Desta forma, após interceção dos dados das 3 fases do ADNI consideradas, e tendo em conta que apenas se quer uma imagem de cada participante, o conjunto de dados final considerado nesta dissertação conta com 1051 imagens de MRI estruturais, das quais 291 são indivíduos com diagnóstico de DA, 387 são indivíduos com diagnóstico de DCL e os restantes 373 são controlos normais. Para a classificação dos indivíduos, nomeadamente das classes DCL e DA, foram tidos em consideração diversos critérios clínicos, tais como a pontuação *Mini-Mental State Examination* (MMSE) e *Clinical Dementia Rating* (CDR). Olhando para o número de instâncias de cada classe consideradas nesta investigação, podemos ver que o conjunto de dados é razoavelmente equilibrado, por isso não é necessária a aplicação de técnicas de balanceamento de classes. Olhando para o tamanho total do conjunto de dados, e como já dito em 2.13, a sua dimensão é uma das limitações a esta investigação.

3.2 Pré-processamento

A natureza das imagens de MRI obtidas, quer em termos de dimensão quer em termos de heterogeneidade nas recolha das imagens, torna a fase de pré-processamento essencial por diversos motivos.

Como foi indicado na seção 2.8, podem ser diversas as abordagens no que toca a pré-processamento e também foram referidos os principais entraves à construção de modelos de DL *end-to-end*.

As CNNs 3D propostas neste trabalho são baseadas em vóxeis, de modo a ter em consideração todos os vóxeis do cérebro e capturar detalhes locais subtis, além das características globais mais facilmente identificáveis nas imagens de MRI. A utilização desta representação detalhada de imagens de MRI baseada em vóxel elimina quaisquer julgamentos *a priori* para a escolha de regiões de interesse ou *patches*, levando em conta todo o cérebro. Ou seja, a entrada dos modelos será um volume 3D, embora não se use a totalidade das MRIs que são originalmente recolhidas.

Além disso, e para comparar o desempenho dos diversos tipos de CNNs, foram criadas CNNs 2D baseadas em *patches* que, como referido, têm a desvantagem de perderem uma representação global do cérebro.

Após esta aproximação às abordagens possíveis e sabendo o que se pretende nesta dissertação, procede-se a uma explicação mais detalhada do pré-processamento realizado.

As imagens de MRIs obtidas do conjunto de dados ADNI apesar de serem as originais, ou seja, não terem sido foram sujeitas a técnicas de pré-processamento, possuem uma enorme variedade em termos da zona cerebral monitorizada, da posição e da inclinação da cabeça do participante, e da dimensão das imagens. Na figura 32 podemos observar algumas diferenças nas imagens originais, onde se pode destacar a posição da cabeça dos participantes e as diferentes dimensões das imagens. Por isso, é fundamental o pré-processamento das imagens, tendo aqui o registo das mesmas um papel fundamental, e que permitirá que todas as imagens passem a ter as mesmas dimensões e que os mesmos vóxeis representem a mesma informação anatómica.

Para além disso, no final deste processo estarão disponíveis quatro conjuntos de dados, dois 2D e dois 3D. Apesar das técnicas de pré-processamento serem quase as mesmas para a construção destes conjuntos de dados, mudando apenas alguns parâmetros, foi decidido criar dois conjuntos de dados, um 2D e um 3D, com imagens reduzidas, ou seja, que durante este processo de pré-processamento perdem bastante informação, e outro conjunto que mantém quase toda a informação das imagens originais.

3.2.1 *Normalização espacial*

O primeiro passo do pré-processamento das imagens passou por uma normalização espacial, com o objetivo de assegurar que a estrutura espacial de todas as imagens do conjunto de dados fosse o mais semelhante possível. Para isso começou por se efetuar a reamostragem das imagens. Para os conjuntos de dados mais simples, foi feita uma reamostragem das imagens para uma resolução isotrópica de $2mm^3$ por vóxel, ou seja, cada vóxel passa a

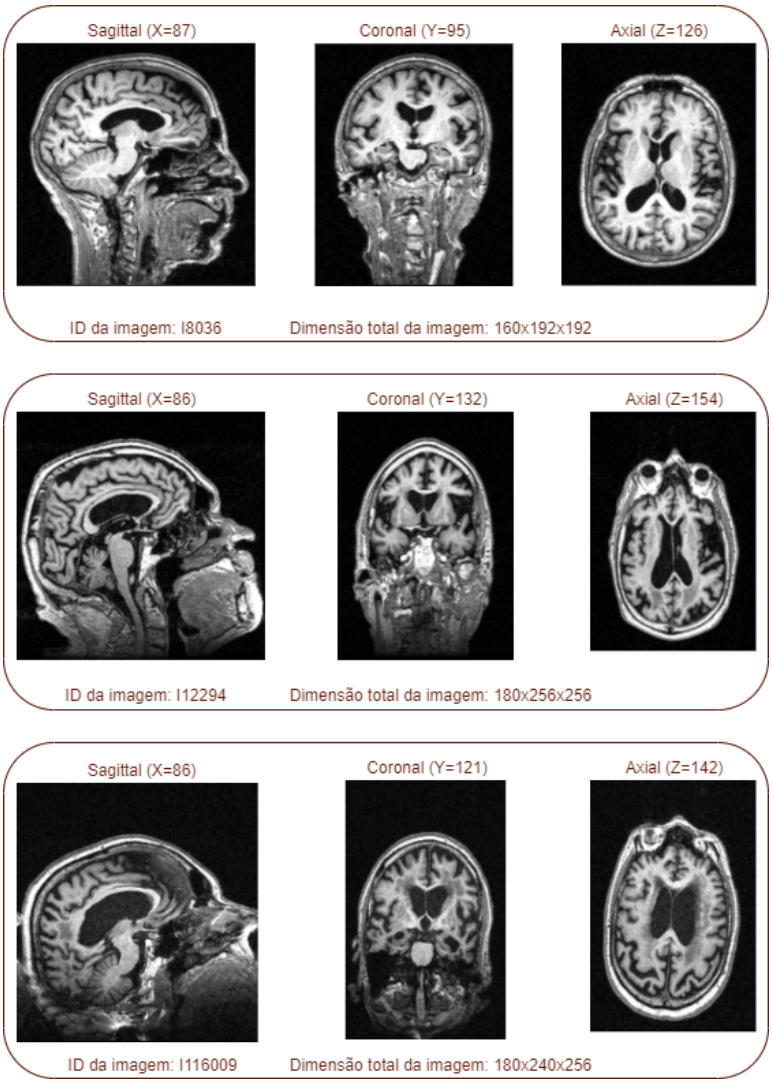


Figura 32: Variedade existente nas imagens de MRI originais.

representar $2mm^3$ de espaço no "mundo real". Para os conjuntos de dados mais complexos, foi realizada uma reamostragem das imagens para uma resolução isotrópica de $1mm^3$. Após a reamostragem para a resolução isotrópica de $2mm^3$, o tamanho das imagens diminuiu. Enquanto que após a reamostragem para a resolução isotrópica de $1mm^3$ o tamanho das imagens diminuiu, na maioria dos casos. Na figura 33 podemos ver uma representação deste processo de reamostragem para a imagem com ID I12294 onde é possível verificar que aconteceu uma alteração do tamanho da imagem original, a qual depende da reamostragem realizada. Para simplificar, apenas se apresenta um corte axial por imagem. Olhando mais atentamente para as imagens, é possível notar uma diminuição na qualidade da imagem resultante da reamostragem para uma resolução isotrópica de $2mm^3$, resultante

da diminuição do tamanho da imagem. A reamostragem foi implementada recorrendo à biblioteca DLTK (Pawlowski et al., 2017).

```
# bibliotecas relevantes para o processo de reamostragem
import SimpleITK as sitk
from dltk.io import preprocessing
from skimage import filters

def resample_img(itk_image, out_spacing=[1.0, 1.0, 1.0]):
    ''' Esta funcao faz reamostragem das imagens para uma resolucao isotropica
        passada como argumento, neste caso 1mm.
        Parametros:
            itk_image   -- Imagem no formato simpleitk
            out_spacing -- Representacao do espaco por cada voxel
        Devolve:
            Imagem em formato simpleitk
    '''
    # Reamostragem das imagens usando SimpleITK
    original_spacing = itk_image.GetSpacing()
    original_size = itk_image.GetSize()
    out_size = [
        int(np.round(original_size[0] * (original_spacing[0] / out_spacing[0]))),
        int(np.round(original_size[1] * (original_spacing[1] / out_spacing[1]))),
        int(np.round(original_size[2] * (original_spacing[2] / out_spacing[2])))]
    resample = sitk.ResampleImageFilter()
    resample.SetOutputSpacing(out_spacing)
    resample.SetSize(out_size)
    resample.SetOutputDirection(itk_image.GetDirection())
    resample.SetOutputOrigin(itk_image.GetOrigin())
    resample.SetTransform(sitk.Transform())
    resample.SetDefaultPixelValue(itk_image.GetPixelIDValue())
    resample.SetInterpolator(sitk.sitkBSpline)
    return resample.Execute(itk_image)

for path, dirs, files in os.walk(DATABASE):
    if files:
        for file in files:
            try:
                complete_file_path = os.path.join(path, file)
                sitk_image = sitk.ReadImage(complete_file_path)
                res = resample_img(sitk_image)
                target_file_path = os.path.join(REG_DB, file)
                sitk.WriteImage(res, target_file_path)
            except RuntimeError:
                print('Exception with', os.path.join(path, file))
```

Listagem 3.1: Implementação do processo de reamostragem das imagens.

O código desenvolvido para implementar a normalização espacial é apresentado na listagem 3.1, utilizando como exemplo a reamostragem para a resolução isotrópica de $1mm^3$.

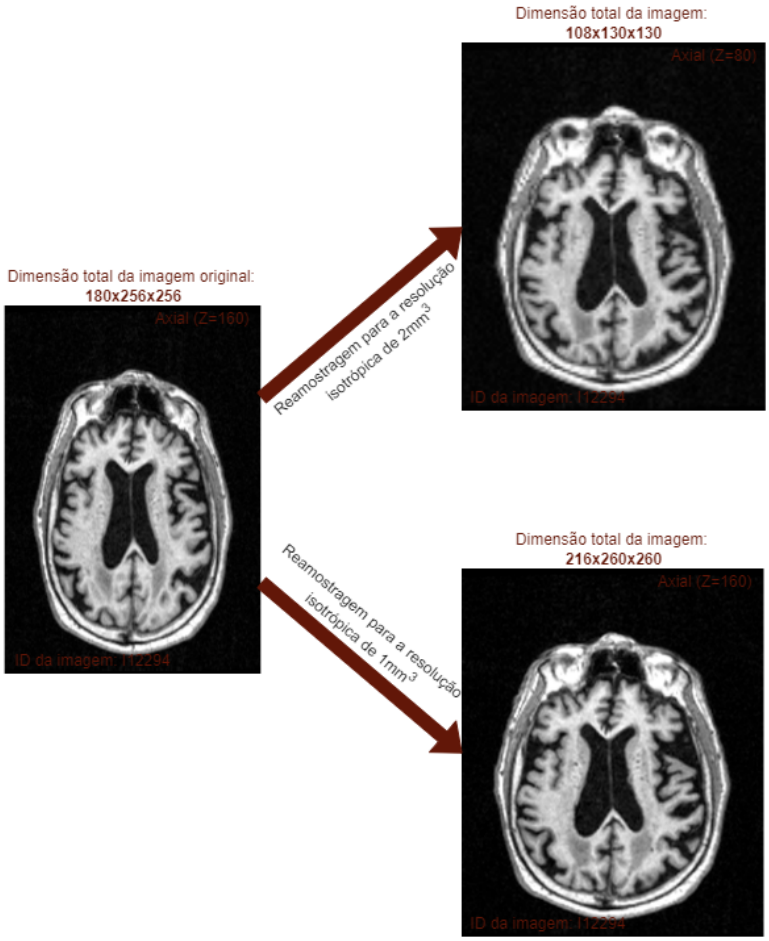


Figura 33: Processo de reamostragem das imagens.

Após a reamostragem foi realizado o registo das imagens médicas com recurso ao modelo MNI152[1] e à ferramenta FLIRT, disponível no *software* FSL[2]. O registo tem como finalidade alinhar as imagens de acordo com um determinado modelo, que neste caso é o MNI152.

A ferramenta FLIRT[3] é totalmente automatizada, robusta e precisa para o registo linear intra e inter-modal de imagens cerebrais, por forma a alinhar todas as imagens de acordo com um determinado modelo (Jenkinson et al., 2002)(Jenkinson and Smith, 2001)(Greve and Fischl, 2009).

Para o conjunto de dados simples, ao qual se aplicou a reamostragem para a resolução isotrópica de $2mm^3$, o registo utilizou o modelo MNI152 também com uma resolução isotrópica de $2mm^3$. Este modelo tem como dimensões $91 \times 109 \times 91$, pelo que após o registo

1 ICBM 2009c Nonlinear Symmetric template, McGill University, Canada.
2 Wellcome Center, University of Oxford, UK.
3 FMRIB's Linear Image Registration Tool.

todas as imagens do conjunto de dados simples passam a ter este tamanho. No conjunto de dados complexo, ao qual foi feita uma reamostragem para uma resolução isotrópica de $1mm^3$, o registo também utilizou o modelo MNI152 e com a resolução isotrópica de $1mm^3$. Este modelo tem como dimensões $182 \times 218 \times 182$, pelo que após o registo todas as imagens do conjunto de dados complexo passam a ter também este tamanho. A figura 34 ilustra todo o processo descrito, incluindo a mudança de dimensão das imagens MRI conforme o modelo usado.

Usando como exemplo o registo do conjunto de dados complexo, a listagem 3.2 apresenta o código desenvolvido para esta tarefa.

```
# bibliotecas relevantes para a etapa de registo
import nibabel as nib
import nipype
from nipype.interfaces import fsl
from subprocess import call

def registration(in_file, out_file, reference):
    fsl.FSLCommand.set_default_output_type('NIFTI')
    flt = fsl.FLIRT(bins=640, cost_func='mutualinfo')
    flt.inputs.in_file = in_file
    flt.inputs.out_file = out_file
    flt.inputs.reference = reference
    result = flt.run()

reference = 'MNI152_T1_1mm.nii.gz'

count = 0

for path, dirs, files in os.walk(DATABASE):
    if files:
        for file in files:
            try:
              print(count)
              count=count+1
              complete_file_path = os.path.join(path, file)
              target_file_path = os.path.join(REG_DB, file)
              registration(complete_file_path, target_file_path, reference)
            except RuntimeError:
                print('Exception with', os.path.join(path, file))
```

Listagem 3.2: Implementação do registo dasimagens.

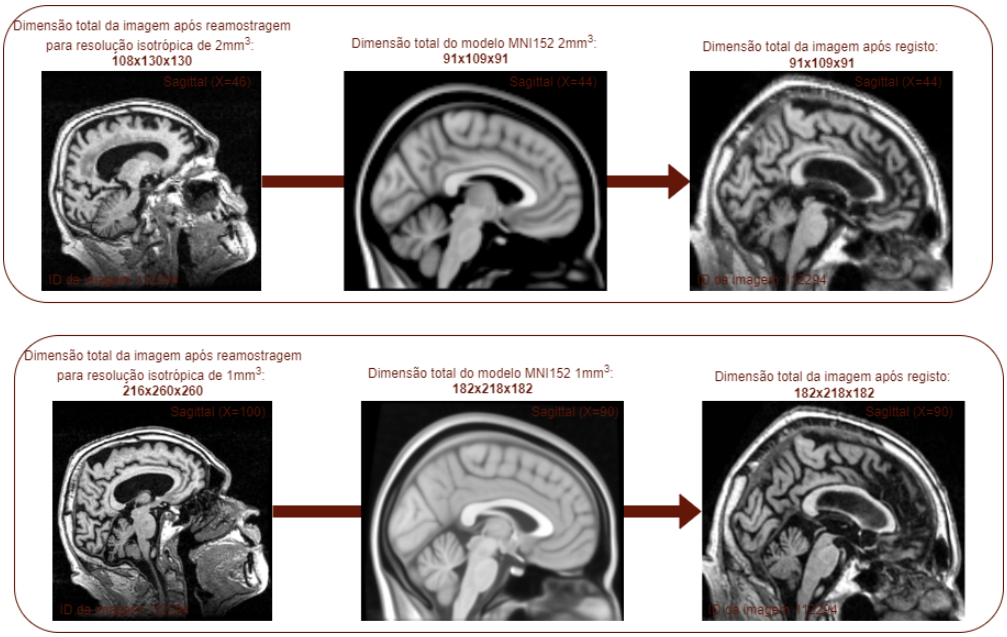


Figura 34: Processo de registo das imagens MRI.

3.2.2 *Remoção do crânio*

Após a normalização espacial e, uma vez que o objetivo da criação do conjunto de dados simples era remover o máximo possível de informação irrelevante, foi aplicada a este conjunto de dados uma técnica de remoção do crânio. Este passo permite remover informação irrelevante das imagens, deixando apenas tecido cerebral.

Para a realização deste processo foi utilizada a ferramenta *Brain Extraction Tool* (BET) (Smith, 2002), também disponível no pacote FSL. O principal desafio neste processo foi encontrar o valor adequado para o principal hiperparâmetro da ferramenta, o *fractional intensity threshold*, que basicamente mede a agressividade do algoritmo na remoção dos tecidos não cerebrais. Valores demasiado baixos deste parâmetro deixam alguma informação irrelevante nas imagens, enquanto que valores demasiado altos acabam por remover algum tecido cerebral.

Uma vez que as imagens já foram normalizadas espacialmente e ainda apresentam alguma heterogeneidade, encontrar um valor ideal para o *fractional intensity threshold*, que funcione na perfeição para a totalidade do conjunto de dados, não é viável. Como este passo é considerado fundamental para a remoção do máximo possível de informação desnecessária, realizaram-se múltiplas experiências e o valor de *threshold* que se obteve foi 0.4. Não sendo o valor ótimo em absoluto, este valor funciona bem na maioria das imagens. A figura 35 apresenta uma imagem após o processo descrito e onde se pode observar que ainda persistem alguns resíduos do crânio.

O código que permite remover o crânio encontra-se na listagem 3.3.

```
# bibliotecas importantes para a tarefa de remocao do cranio
import nibabel as nib
import nipype
from nipype.interfaces import fsl

def skull_strip_nii(original_img, destination_img, frac=0.3):
    ''' Realiza remocao do cranio a uma dada imagem e guarda o resultado numa nova
        magem em formato .nii
        Utiliza FSL-BET
        (https://fsl.fmrib.ox.ac.uk/fsl/fslwiki/BET/UserGuide#Main_bet2_options:)
        Parametros:
            original_img    -- Imagem original em formato .nii
            destination_img -- A nova imagem sem cranio
            frac            -- Fractional intensity threshold para BET
    '''

    btr = fsl.BET()
    btr.inputs.in_file = original_img
    btr.inputs.frac = frac
    btr.inputs.out_file = destination_img
    btr.cmdline
    res = btr.run()

count = 0

for path, dirs, files in os.walk(DATABASE):
    if files:
        for file in files:
            try:
              print(count)
              count=count+1
              complete_file_path = os.path.join(path, file)
              target_file_path = os.path.join(REG_DB, file)
              skull_strip_nii(complete_file_path, target_file_path, frac=0.4)
            except RuntimeError:
                print('Exception with', os.path.join(path, file))
```

Listagem 3.3: Implementação da remoção do crânio das imagens.

Neste momento existem dois conjuntos de dados com imagens 3D. Um simples ao qual foi aplicada reamostragem para uma resolução isotrópica de $2mm^3$ por vóxel, registo com o modelo MNI152 também com resolução isotrópica de $2mm^3$ por vóxel e ainda remoção do crânio. O outro conjunto de dados que possui imagens mais complexas, ao qual foi aplicada reamostragem para uma resolução isotrópica de $1mm^3$ por vóxel e registo com o modelo MNI152 também com uma resolução isotrópica de $1mm^3$ por vóxel.

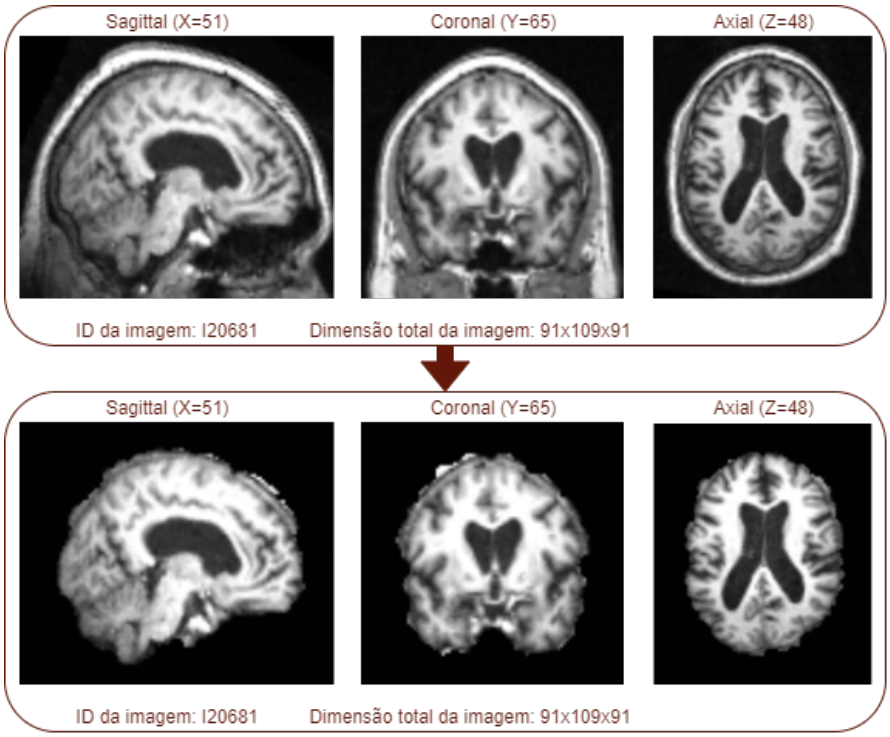


Figura 35: Processo de remoção de tecido não cerebral.

3.2.3 *Criação dos conjuntos de dados 2D*

As CNNs 2D são um tipo de DNN tipicamente desenhado para trabalhar com imagens RGB, contendo duas dimensões para a estrutura das imagens e uma terceira dimensão para a informação de cor. Por isso é impossível utilizar as imagens existentes neste momento, as quais contêm $91 \times 109 \times 91$ vóxeis, no caso do conjunto de dados simples, e $182 \times 218 \times 182$ vóxeis no conjunto de dados complexo.

Inspirado no procedimento descrito em Ding et al. (2019) e em Darias Plasencia (2019), para transformar as imagens 3D em imagens 2D foi implementado um algoritmo bastante simples. Este algoritmo consiste em fazer múltiplos cortes axiais (horizontais) e colocá-los no mesmo plano para construir uma imagem bidimensional. Mais especificamente, foram retiradas 16 fatias diferentes de cada imagem 3D e colocadas numa matriz de imagens 4×4, conforme mostra a figura 36. A imagem bidimensional resultante é replicada três vezes para respeitar as dimensões RGB.

Para o conjunto de dados simples foram retirados 16 cortes axiais com tamanho 91×109, resultando numa imagem final com tamanho $364 \times 436 \times 3$ (figura 37). Para o conjunto de dados complexo foram retirados 16 cortes axiais com tamanho 182×218, resultando numa imagem final com tamanho $728 \times 872 \times 3$ (figura 38). Estas duas figuras mostram claramente a diferença de informação presente em cada uma delas, quer seja no nível de detalhe das

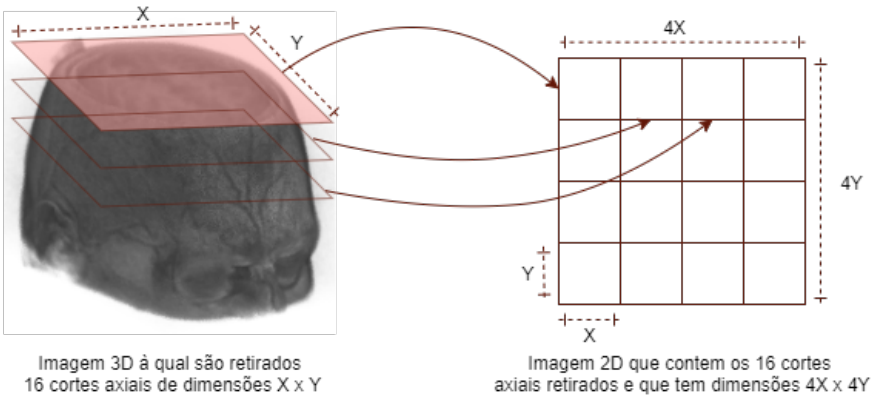


Figura 36: Processo de passagem de uma imagem 3D para uma imagem 2D.

imagens resultante da diferença de tamanho, quer seja na ausência de informação irrelevante devido à extração do crânio no conjunto de dados simples.

Na listagem 3.4 encontra-se um excerto de código que foi fundamental para a criação das imagens 2D. Neste caso é apresentado como exemplo a criação do conjunto de dados 2D simples.

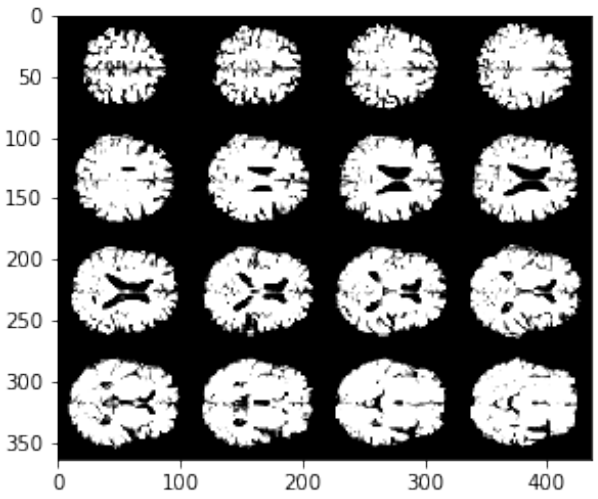


Figura 37: Uma imagem 2D simples.

```
def slices_matrix_2D(img):
  ''' Transforma uma MRI 3D numa imagem 2D atraves da obtencao de 16 fatias e
      colocando-as numa grelha bidimensional 4x4
      Os 16 cortes sao feitos ao longo da visao axial das imagens e sao selecionadas
      entre o 30 e 60 cortes da imagem original
      Parametros:
        img        -- np.ndarray com a imagem 3D
      Devolve:
        np.ndarray com a imagem 2D resultante
  '''

  # criar a imagem 2D
  image_2D = np.empty(IMG_2D_SHAPE)

  # definir os limites
  TOP = 60
  BOTTOM = 30
  STEP = 2
  N_CUTS = 16

  # iterador para os cortes
  cut_it = TOP
  # iterador para as linhas da imagem final 2D
  row_it = 0
  # iterador para as colunas da imagem final 2D
  col_it = 0

  for cutting_time in range(N_CUTS):
    # corte
    cut = img[cut_it, :, :]
    cut_it -= STEP
    if cutting_time in [4, 8, 12]:
      row_it = 0
      col_it += cut.shape[1]
    for i in range(cut.shape[0]):
      for j in range(cut.shape[1]):
        image_2D[i + row_it, j + col_it] = cut[i, j]
    row_it += cut.shape[0]

  # devolve uma imagem 2D, com 3 canais (necessario para treinar com grande parte
  # das redes pre-treinadas)
  return np.repeat(image_2D[None, ...], 3, axis=0).T
```

Listagem 3.4: Implementação da transformação das imagens 3D em imagens 2D.

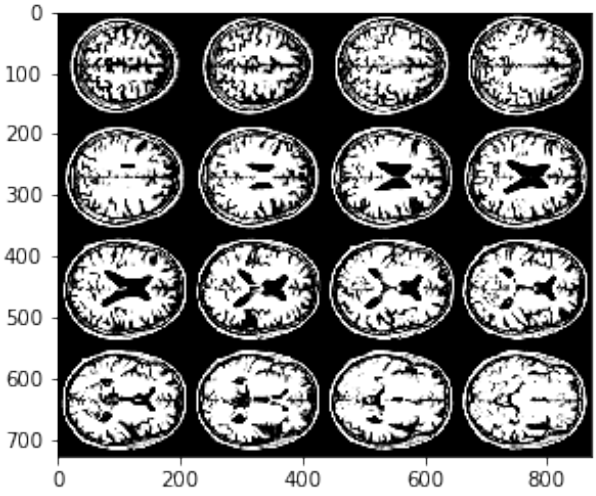


Figura 38: Uma imagem 2D complexa.

3.2.4 *TFRecords*

Os conjuntos de dados 3D são consideravelmente mais complexos que os conjuntos de dados 2D com imagens RGB. Consequentemente, o tamanho dos diferentes conjuntos de dados é bastante diferente. A máquina usada para treino, mesmo que possua uma GPU com bastante memória, dificilmente será suficiente para armazenar todo o conjunto de dados de uma vez só. Por isso, foi necessário arranjar uma solução diferente para alimentar os modelos.

A primeira solução encontrada, que é uma solução amplamente utilizada em Keras, passou por fazer uso dos geradores nativos do Python, os quais permitem gerar um objeto iterável.

A segunda alternativa consistiu em utilizar ficheiros `TFRecord`. O TensorFlow fornece aos programadores um formato de armazenamento persistente baseado em ficheiros `TFRecord`. Neste caso, os objetos são serializados e armazenados em ficheiros, num formato que permite a leitura rápida dessa informação que depois será guardada em memória, no formato dos tensores do TensorFlow. O formato `TFRecord` requer uma duplicação do espaço necessário para armazenar a mesma informação, mas permite uma leitura muito mais rápida do que os geradores nativos de Python. Este formato é compatível com os modelos da API Keras.

Após serem criados, os ficheiros `TFRecord` são lidos durante o treino e a informação é passada aos modelos *batch* a *batch*. Durante a leitura, o *whitening* é aplicado sobre as imagens, através do operador `tf.image.per_image_standardization`.

O código desenvolvido para criar os *TFRecords* com as imagens 3D pode ser consultado na listagem 3.5.

```
def load_image_3D(abs_path):
  ''' Carrega uma imagem (.nii) e a sua etiqueta, atraves do seu caminho absoluto.
      Parametros:
        abs_path -- Caminho absoluto, incluindo o nome do ficheiro
      Devolve:
        img      -- A imagem .nii, convertida num numpy array
        label    -- A etiqutea da imagem
  '''

  # obter a etiqueta usando o caminho (nome da ultima diretoria)
  split_path = abs_path.split('/')
  label = LABELS[split_path[-2]]
  # carregar a imagem com SimpleITK
  sitk_image = sitk.ReadImage(abs_path)
  # transformar num numpy array
  img = sitk.GetArrayFromImage(sitk_image)

  return img, label

def create_tf_record(img_filenames, tf_rec_filename):
  ''' Cria um ficheiro TFRecord
      parametros:
        img_filenames   -- Array com o caminho para cada imagem que vai
                           ser incluida no ficheiro TFRecord.
        tf_rec_filename -- Nome do ficheiro TFRecord a ser criado.
  '''

  # abrir o ficheiro
  writer = tf.io.TFRecordWriter(tf_rec_filename)

  # iterar sobre cada ficheiro .nii
  for meta_data in img_filenames:
    # carregar a imagem e a etiqueta
    img, label = load_image_3D(meta_data)
    # criar a 'feature'
    feature = {'label': _int64_feature(label),
               'image': _float_feature(img.ravel())}
    example = tf.train.Example(features=tf.train.Features(feature=feature))
    # Escrever no ficheiro
    writer.write(example.SerializeToString())

  writer.close()
```

Listagem 3.5: Criação de `TFRecords` com as imagens 3D.

```python
def load_image_2D(abs_path, labels):
  ''' Carrega uma imagem (.nii) e a sua etiqueta, atraves do seu caminho absoluto.
      Parametros:
        abs_path -- Caminho absoluto, incluindo o nome do ficheiro
        labels   -- Etiqueta
      Devolve:
        img      -- A imagem .nii, convertida num numpy array
        label    -- A etiqueta da imagem
  '''
  # obter a etiqueta usando o caminho (nome da ultima diretoria)
  label = labels[abs_path.split('/')[-2]]
  # carregar a imagem com SimpleITK
  sitk_image = sitk.ReadImage(abs_path)
  # transformar em numpy array
  img = sitk.GetArrayFromImage(sitk_image)
  # aplicar whitening
  img = preprocessing.whitening(img)
  # criar a imagem 2D
  img = slices_matrix_2D(img)

  return img, label

def create_tf_record_2D(img_filenames, tf_rec_filename, labels):
  '''
      Cria um ficheiro TFRecord apos converter as imagens numa grelha 2D
      Parametros:
        img_filenames   -- Array com o caminho para cada imagem que vai
                           ser incluida no ficheiro TFRecord.
        tf_rec_filename -- Nome do ficheiro TFRecord a ser criado.
        labels          -- Etiqueta
  '''

  # abrir o ficheiro
  writer = tf.io.TFRecordWriter(tf_rec_filename)
  # iterar por cada ficheiro .nii
  for meta_data in img_filenames:
    # carregar a imagem e a etiqueta
    img, label = load_image_2D(meta_data, labels)
    # criar a 'feature'
    feature = {'label': _int64_feature(label),
               'image': _float_feature(img.ravel())}
    example = tf.train.Example(features=tf.train.Features(feature=feature))
    # escrever no ficheiro
    writer.write(example.SerializeToString())

  writer.close()
```

Listagem 3.6: Implementação da criação de `TFRecords` com as imagens 2D.

Para as imagens 2D o procedimento é semelhante e a sua implementação pode ser analisada na listagem 3.6.

Convém explicar a forma como foram criados os `TFRecords` para os conjuntos de dados de treino, validação e teste. Antes da criação dos `TFRecords`, as imagens destes 3 conjuntos estavam guardados em pastas diferentes. Por isso foi possível obter a etiqueta associada a cada imagem a partir do caminho absoluto do ficheiro contendo a imagem. Foram utilizadas 70% das imagens para criar o conjunto de dados de treino, 15% das imagens para o conjunto de dados de validação e os restantes 15% para o conjunto de dados de teste. O conjunto de dados de treino fica assim com apenas 760 instâncias, o que é muito pouco, comparado com o que é necessário para construir bons modelos de DL.

Concluída a fase de pré-processamento dos dados, dispomos de quatro conjuntos de dados distintos: conjunto de dados simples bidimensional, conjunto de dados simples 3D, conjunto de dados complexo bidimensional e conjunto de dados complexo 3D, guardados em ficheiros do tipo `TFRrecord`, devidamente separados para treino, validação e teste dos modelos. Estão assim reunidas as condições para passar à implementação, treino e avaliação dos modelos.

4

MODELOS IMPLEMENTADOS

Esta capítulo começa com uma exposição das principais tecnologias a serem utilizadas na concretização das tarefas subsequentes. São também expostos os modelos 2D e 3D implementados, quais os parâmetros utilizados, a forma como foram treinados e os resultados obtidos.

Por fim, é feita uma análise geral dos resultados e são feitas as recomendações para profissionais que queiram implementar modelos relacionados com o doença de Alzheimer.

4.1 TECNOLOGIAS

Apresentam-se a seguir as principais tecnologias utilizadas durante o desenvolvimento da dissertação, mais especificamente as tecnologias utilizadas para desenvolver as redes neuronais profundas. Para cada tecnologia é apresentada uma introdução sobre a mesma, bem como alguns aspetos que fazem com que sejam importantes para a dissertação.

4.1.1 *Python*

Python é uma linguagem de programação criada pelo programador holandês Guido van Rossum e lançada pela primeira vez em 1991. Como grande fã da série de comédia "*Monty Python's Flying Circus*", criada pelo grupo de comédia britânico Monty Python, também conhecidos como "*The Pythons*", decidiu fazer um tributo aos seus comediantes favoritos tendo dado como nome à sua linguagem de programação "Python". A filosofia da linguagem enfatiza a legibilidade do código, sendo fácil de aprender e permite expressar ideias complexas com clareza. Atualmente é uma linguagem muito popular, de uso geral, e que pode ser utilizada para uma grande variedade de aplicações, nomeadamente em inteligência artificial e ciência de dados.

Python está a tornar-se cada vez mais popular em ML e *Big Data*, duas tecnologias emergentes que atraem muito interesse global. Isto explica ainda mais a crescente popularidade da linguagem.

A linguagem de programação Python caracteriza-se por ser interpretada, orientada para objetos, de alto nível com semântica dinâmica, compilada em tempo de execução, simples, compatível e para a qual existe uma grande quantidade de bibliotecas disponíveis. Inclui milhares de módulos de terceiros, disponíveis no Índice de Pacotes Python (PyPI). PyPI fornece padrões populares para diferentes especialidades, como Django para desenvolvimento Web, NumPy, Pandas, e Matplotlib para ciência de dados.

O código desenvolvido nesta dissertação está de acordo com a norma Python 3, lançado em 2008, que não é completamente retro compatível, e muito do código Python 2 não corre sem modificação em Python 3.

4.1.2 *TensorFlow*

O TensorFlow é uma biblioteca de software de código aberto lançada em 2015 pela Google para ajudar os desenvolvedores a conceberem, construírem, e treinarem modelos de aprendizagem profunda (Buduma, 2017). De uma forma abstrata, o TensorFlow é uma biblioteca em Python que utiliza grafos de fluxo de dados computacionais para representar a arquitetura das redes neuronais. Os nodos do grafo representam as unidades de computação, também chamadas de operações, enquanto que as arestas denotam os dados consumidos ou produzidos por uma unidade de computação. Além disso, os gradientes relevantes são armazenados em cada nodo e durante a retro propagação. Os gradientes dos vários nodos são combinados para obter os gradientes em relação a cada peso/viés[1] do modelo (Pattanayak, 2017). Os dados em TensorFlow são representados como tensores, que essencialmente são matrizes multidimensionais representando vetores com um tensor 1D, matrizes com um tensor 2D, etc.

Utilizar grafos de fluxo de dados para representar os cálculos, os quais são um modelo de programação amplamente utilizado em computação paralela, tem a vantagem de se poder executar os passos para a frente e para trás necessários para treinar um modelo paramétrico de aprendizagem profunda através do método da descida de gradiente[2], mas tem também outras vantagens associadas, como por exemplo (i) explorar o paralelismo, dado que a representação com grafos de fluxo de dados permite ao TensorFlow identificar operações que podem ser executadas em paralelo, e (ii) executar as operações de forma distribuída, uma vez que cada nodo do grafo pode ser colocado num dispositivo independente e numa máquina diferente, ficando o TensorFlow responsável pela gestão da comunicação entre os nós e por garantir que a execução do grafo é correta; a distribuição da computação torna possível treinar e executar redes neuronais grandes de forma eficiente, distribuindo os cálculos por potencialmente centenas de servidores multi-GPU (Géron, 2019).

1 *Bias*, na terminologia inglesa.
2 *Gradient descent*, na terminologia inglesa.

Para facilitar a utilização do TensorFlow foram construídas algumas APIs de alto nível que ajudam no desenvolvimento dos modelos de aprendizagem, sendo o Keras uma delas. A versão do TensorFlow usada nesta dissertação foi a 2.4.0.

4.1.3 *Keras*

O Keras é uma API de alto nível para ajudar em tarefas de DL, escrita em Python, que torna muito simples o treino e a gestão de redes neuronais e que está completamente integrada no TensorFlow, mas também funciona com Pytorch, Theano ou Microsoft Cognitive Toolkit como *backend*. O TensorFlow possui a sua própria implementação da API Keras, chamada `tf.keras`, que fornece suporte para algumas características avançadas do TensorFlow, como por exemplo, carregar dados de forma eficiente (Géron, 2019).

A API Keras suporta tanto computação em CPU, como GPU ou TPU, e é uma ótima ferramenta para prototipagem rápida de ideias (Ketkar, 2017). A versão do Keras utilizada nesta dissertação foi a 2.4.3.

4.2 MODELOS 2D

As primeiras CNNs implementadas neste trabalho são 2D. Seguindo a tendência de sucesso dos últimos anos de usar redes pré-treinadas com o conjunto de dados Imagenet, os dois conjuntos de dados 2D foram usados para reajustar um conjunto de diversas CNNs.

Todas as CNNs foram implementadas com recurso à API Keras, que permite obter as redes já com os pesos do treino com Imagenet. É importante referir que a última camada de cada uma das redes pré-treinadas tem que ser sempre ignorada, pois as dimensões das imagens dos conjuntos de dados aqui criados são diferentes do conjunto de dados Imagenet. Para além disso, no final de cada uma destas redes é adicionada uma camada `softmax` com três unidades de saída para que seja feita a classificação das três classes do nosso problema (AD/MCI/CN). As redes CNN 2D utilizadas foram `DenseNet121`, `DenseNet169`, `DenseNet201`, `Inceptionv3`, `InceptionResNetV2`, `MobileNet`, `MobileNetV2`, `ResNet50`, `ResNet101`, `ResNet152` e `Xception`.

Convém ainda referir que todas as CNNs foram treinadas seguindo o mesmo procedimento e perante as mesmas condições e hiperparâmetros, que serão enunciados de seguida. Este não é o procedimento ideal, pois idealmente teria de se procurar quais os melhores hiperparâmetros para cada CNN, mas como foi mencionado na seção 2.13 o tempo e poder de computação são limitações difíceis de contornar.

Em primeiro lugar, as imagens 2D foram carregadas dos `TFRecords` criados. O código desenvolvido para essa tarefa pode ser consultado na listagem 4.1, nomeadamente o código da função `decode`, que permite descodificar uma imagem guardada no formato `TFRecord`, e

o código da função `dataset_parser`, que cria uma um objeto iterável por todas as imagens presentes num `TFRecord`. No final, são criadas as duas variáveis necessárias ao treino dos modelos.

```
def decode(serialized_example):

  # descodifica as instancias guardadas no TFRecord
  feature = tf.io.parse_single_example(
      serialized_example,
      features = {
          'image': tf.io.FixedLenFeature(IMG_RGB_SHAPE, tf.float32),
          'label': tf.io.FixedLenFeature([], tf.int64)
      }
  )

  return feature['image'], tf.one_hot(feature['label'], n_classes)

def dataset_parser(filepath, batch_size, n_classes):
  ''' Este metodo cria um pipeline para alimentar os modelos com dados guardados em TFRecord.
      Parametros:
        filepath   -- O caminho para o ficheiro TFRecord
        batch_size -- tamanho do batch
        n_classes  -- numero de classes
      Devolve:
        image -- Tensor com as imagens
        label -- Tensor com as etiquetas
  '''

  dataset = tf.compat.v1.data.TFRecordDataset(filepath).map(decode)
  dataset = dataset.shuffle(SHUFFLE_BUFFER, seed=tf.random.set_seed(21))
  dataset = dataset.repeat()
  dataset = dataset.batch(batch_size)
  dataset = dataset.prefetch(1)

  return dataset

# tensor para treino
train_dataset=dataset_parser(training_tfrec, BATCH_SIZE, n_classes)

# tensor para validacao
val_tensor=dataset_parser(validation_tfrec, BATCH_SIZE, n_classes)
```

Listagem 4.1: Implementação das funções de descodificação e carregamento dos dados para memória.

O código presente na listagem 4.2 mostra como foram instanciados os modelos a partir da biblioteca Keras com os pesos do Imagenet, neste exemplo para o modelo `ResNet101`.

```
base_model = tf.keras.applications.resnet.ResNet101(
    input_shape=(IMG_2D_SHAPE[0], IMG_2D_SHAPE[1], 3),
    weights='imagenet',
    include_top=False,
    pooling='avg')

base_output = base_model.output

# criar a ultima camada de classificacao
output_layer = tf.keras.layers.Dense(n_classes, activation='softmax')(base_output)

# modelo final
model = tf.keras.models.Model(inputs=base_model.input, outputs=output_layer)
```

Listagem 4.2: Instanciação de modelos 2D.

Numa primeira fase, para treinar cada uma das CNNs foram congeladas as camadas base do modelo e treinam-se apenas as camadas adicionadas manualmente, que são camadas totalmente ligadas, também designadas por camadas de classificação. Para cada um dos modelos, o treino decorreu durante 200 épocas com o otimizador `Adam`, com uma taxa de aprendizagem inicial de 0.0001, e com uma constante de decaimento da taxa de aprendizagem de 0.001. O tamanho de *batch* utilizado foi de 8 para todas as redes treinadas com o conjunto de dados 2D simples e, 2 para as redes treinadas com o conjunto de dados 2D complexo. O excerto de código incluído na listagem 4.3 mostra como foi implementado o treino dos modelos.

```
# treinar apenas as camadas totalmente ligadas, ou seja,
# 'congelando' todas as camadas convolucionais
for layer in base_model.layers:
    layer.trainable = False

# compilar o modelo apos definir quais as camadas que nao serao treinadas
optimizer = tf.keras.optimizers.Adam(lr=0.0001, decay=1e-3)
model.compile(optimizer=optimizer, loss='categorical_crossentropy',
              metrics=['acc'])

# treinar o modelo
fc_training = model.fit(train_dataset,
                        epochs=200, steps_per_epoch=STEPS_PER_EPOCH,
                        validation_data=val_tensor,
                        validation_steps=VALIDATION_STEPS)
```

Listagem 4.3: Primeira fase de treino das CNNs 2D.

Depois deste treino inicial, foi realizado o ajuste fino[3] da totalidade das CNNs. Neste processo é também usado o otimizador `Adam` com uma taxa de aprendizagem igual a 0.0001

3 *Fine-tuning*, na terminologia inglesa.

e com uma taxa de decaimento de 0.000001. O tamanho de *batch* nesta fase é igual ao usado na primeira fase do treino.

A fase de ajuste fino da totalidade das CNNs decorreu durante 500 épocas. No treino foram usadas a *categorical cross-entropy* como função de perda e a acurácia como métrica de desempenho auxiliar. A listagem 4.4 mostra a implementação desta etapa.

```
# indicar que todas as camadas vao ser treinadas
for layer in model.layers:
  layer.trainable = True

optimizer = tf.keras.optimizers.Adam(lr=0.0001, decay=1e-6)
model.compile(optimizer=optimizer, loss='categorical_crossentropy',
              metrics=['acc'])

fine_tunning_history = model.fit(train_dataset,
                                 epochs=500, steps_per_epoch=STEPS_PER_EPOCH,
                                 validation_data=val_tensor,
                                 validation_steps=VALIDATION_STEPS)
```

Listagem 4.4: Segunda fase do treino das CNNs 2D.

As figuras 39 e 40 mostram a evolução da acurácia e da perda, respetivamente para os modelos `ResNet50` e `DenseNet201`, treinados com o conjunto de dados 2D simples e para a fase de ajuste fino. Para isso foi criada uma função que cria os gráficos, a qual pode ser consultada na listagem 4.5.

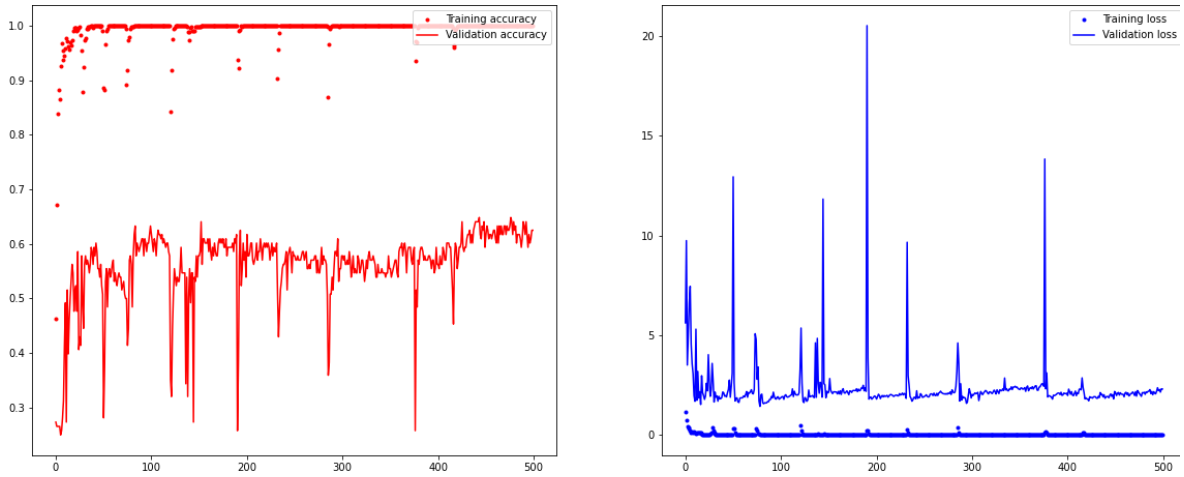


Figura 39: Acurácia e perda no treino da CNN 2D `ResNet50` com o conjunto de dados 2D simples.

```
def plot_training(history, title):
  '''
  Cria um grafico com a acuracia e a perda do treino e da validacao
  '''

  acc = history.history['acc']
  val_acc = history.history['val_acc']
  loss = history.history['loss']
  val_loss = history.history['val_loss']
  epochs = range(len(acc))

  plt.figure(figsize=(20, 8))
  plt.subplot(121)
  plt.plot(epochs, acc, 'r.')
  plt.plot(epochs, val_acc, 'r')
  plt.legend(('Training accuracy', 'Validation accuracy'),
             loc='upper right')
  plt.title(title + '- Accuracy')
  plt.subplot(122)
  plt.plot(epochs, loss, 'b.')
  plt.plot(epochs, val_loss, 'b')
  plt.legend(('Training loss', 'Validation loss'),
             loc='upper right')
  plt.title(title + '- Loss')

  export_file = '-'.join(title.split()) + '.png'
  plt.savefig(os.path.join(DB_MODELS, export_file))

  plt.show()

plot_training(fine_tunning_history, 'Title')
```

Listagem 4.5: Função para cria os gráficos que representam a evolução da acurácia e da perda, ao longo do treino e da validação.

As figuras 41 e 42 mostram a evolução da acurácia e da perda, respetivamente para os modelos `DenseNet121` e `MobileNetV2`, treinados com o conjunto de dados 2D complexo e para a fase de ajuste fino.

Um aspeto que salta à vista ao analisar as figuras 39, 40, 41 e 42 é a grande oscilação dos valores da acurácia e da perda. Esta oscilação deve-se ao facto dos conjuntos de dados serem muito pequenos, complicando principalmente a tarefa de validação. Contudo, e apesar de ser algo comum e esperado, constata-se que para o conjunto de treino a otimização dos modelos evolui de forma consistente.

O próximo passo consistiu em avaliar os modelos no conjunto de dados de teste e o código desenvolvido para essa fase encontra-se na listagem 4.6. Este código inclui como tarefa

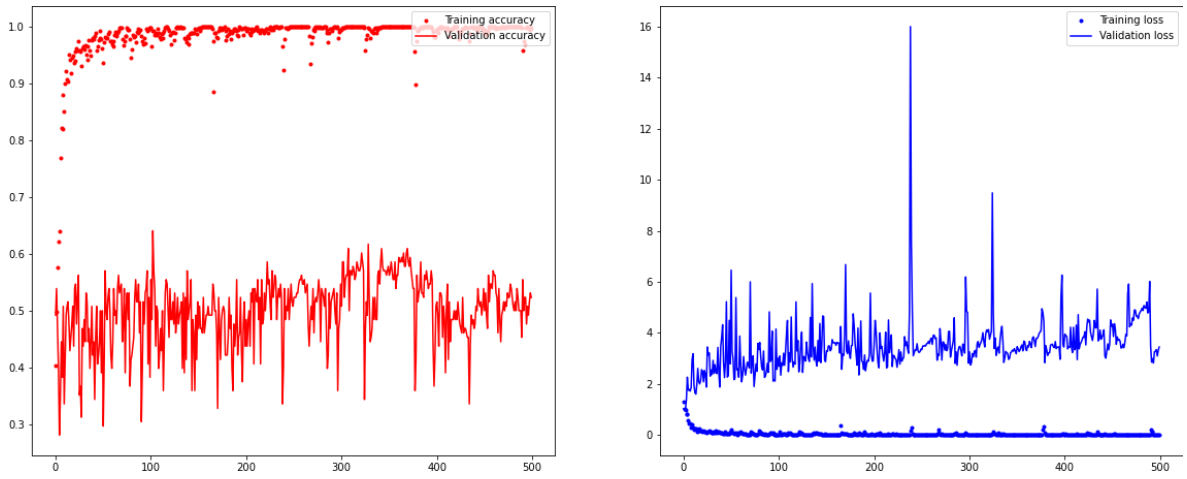


Figura 40: Acurácia e perda no treino da CNN 2D `DenseNet201` com o conjunto de dados 2D simples.

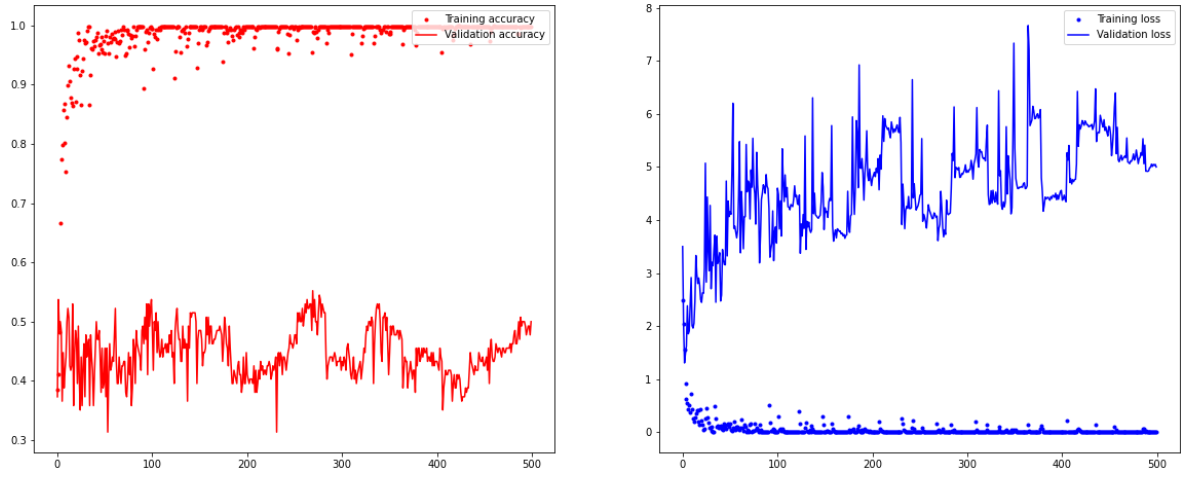


Figura 41: Acurácia e perda no treino da CNN 2D `DenseNet121` com o conjunto de dados 2D complexo.

principal carregar os dados de teste a partir de ficheiros `TFRecord` para memória e depois calcula as predições utilizando estes dados como entrada no modelo.

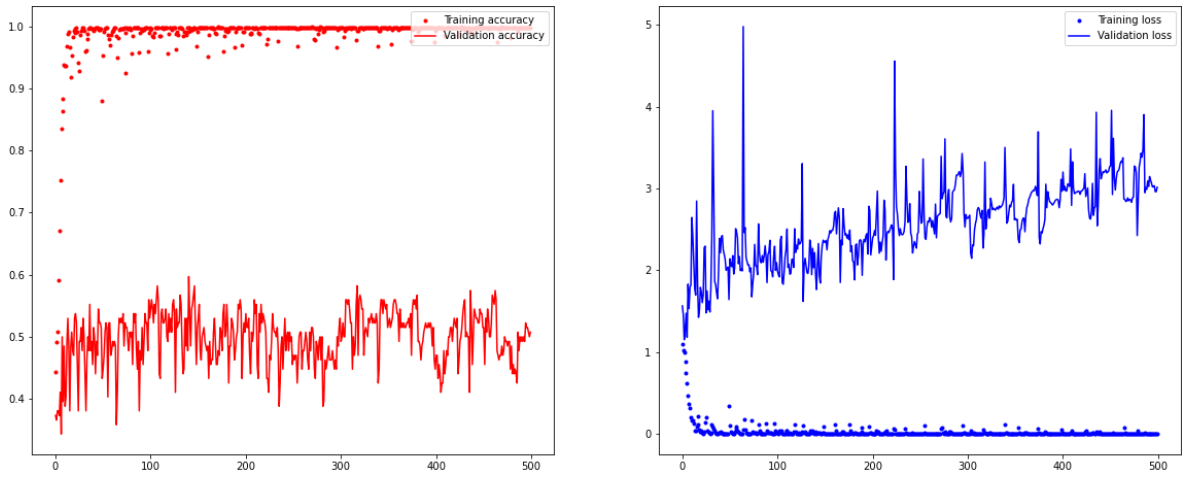


Figura 42: Acurácia e perda no treino da CNN 2D `MobileNetV2` com o conjunto de dados 2D complexo.

```python
def get_data_for_predict(tfrec_file, decoding_fn):
  ''' Ler e devolver o conteudo de um TFRecord.
      Parametros:
        tfrec_file  -- caminho para o ficheiro TFRecord
        decoding_fn -- funcao que descodifica os dados
      Devolve:
        Tuplo com as imagens e as etiquetas
  '''
  images = np.empty((n_test_samples, *IMG_RGB_SHAPE))
  labels = np.empty((n_test_samples))

  # criar o iterador para o TFRecord
  dataset = tf.compat.v1.data.TFRecordDataset(tfrec_file).map(decoding_fn)
  dataset = dataset.batch(1)
  dataset = dataset.prefetch(1)
  return dataset

test_dataset=get_data_for_predict(test_tfrec, decode)

# guardar as etiquetas
y_true = []
for x,y in test_dataset:
    y_true.append(y.numpy())
y_true = np.concatenate(y_true)

# calcular as predicoes e guardar os resultados
y_pred = model.predict(test_dataset)
rounded_probs = np.round(y_pred)
```

Listagem 4.6: Avaliar um modelo no conjunto de dados de teste.

```
def compute_auc(y_true, y_pred, labels, target_names, title):
  ''' Calcular a curva ROC e AUC, para um modelo e para o conjunto de dados de teste
      Parametros:
        model        -- O modelo treinado
        images       -- Imagens do conjunto de dados de teste
        labels       -- Etiquetas das imagens
        target_names -- Nome das etiquetas
  '''

  n_classes = len(target_names)
  y_true = y_true
  y_pred = y_pred

  fpr = dict()
  tpr = dict()
  roc_auc = dict()
  for i in range(n_classes):
    fpr[i], tpr[i], _ = skmetrics.roc_curve(y_true[:, i], y_pred[:, i])
    roc_auc[i] = skmetrics.auc(fpr[i], tpr[i])

  # calcular a curva micro-average ROC e a area ROC
  fpr['micro'], tpr['micro'], _ = skmetrics.roc_curve(y_true.ravel(), y_pred.ravel())
  roc_auc['micro'] = skmetrics.auc(fpr['micro'], tpr['micro'])

  # calcular a curva macro-average ROC e a area ROC
  # agregar a taxa de todos os falsos positivos
  all_fpr = np.unique(np.concatenate([fpr[i] for i in range(n_classes)]))

  # Interpolar as curvas ROC
  mean_tpr = np.zeros_like(all_fpr)
  for i in range(n_classes):
      mean_tpr += interp(all_fpr, fpr[i], tpr[i])

  # calcular a media e calcular a AUC
  mean_tpr /= n_classes

  fpr['macro'] = all_fpr
  tpr['macro'] = mean_tpr
  roc_auc['macro'] = skmetrics.auc(fpr['macro'], tpr['macro'])
  ...
```

Listagem 4.7: Função para calcular a curva ROC e a respetiva AUC (parte 1).

Como foi mencionado na seção 2.14, a acurácia não é uma boa métrica para avaliar o desempenho de um modelo no contexto de avaliação médica (Mazurowski et al., 2008), e foi aqui utilizada apenas como medida de avaliação auxiliar aquando do treino. Pelo contrário, a AUC é uma métrica comummente utilizada para avaliar o desempenho dos

modelos em cada uma das classes das amostras de entrada. Desta forma, e utilizando os métodos de avaliação disponibilizados pela biblioteca `scikit-learn` (Pedregosa et al., 2011), foi implementado um método para calcular a AUC para cada uma das classes, num modelo treinado com o conjunto de dados de teste. O código criado para esta tarefa é mostrado nas listagens 4.7 e 4.8.

```
...
# criar um grafico com todas as curvas ROC
plt.figure(figsize=(10, 10))
plt.plot(fpr['micro'], tpr['micro'],
         label='micro-average ROC curve (area = {0:0.2f})'
               ''.format(roc_auc['micro']),
         color='aqua', linestyle=':', linewidth=2)

plt.plot(fpr['macro'], tpr['macro'],
         label='macro-average ROC curve (area = {0:0.2f})'
               ''.format(roc_auc["macro"]),
         color='navy', linestyle=':', linewidth=2)

colors = cycle(['green', 'gold', 'darkred'])
for i, color in zip(range(n_classes), colors):
    plt.plot(fpr[i], tpr[i], color=color, lw=2,
             label='ROC curve of class {0} (area = {1:0.2f})'
             ''.format(target_names[i], roc_auc[i]))

plt.plot([0, 1], [0, 1], 'k--', lw=2)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title(title)
plt.legend(loc="lower right")

export_file = '-'.join(title.split()) + '.png'
plt.savefig(os.path.join(DB_MODELS, export_file))

plt.show()

compute_auc(y_true, y_pred, LABELS, ['CN','MCI', 'AD'], 'Title')
```

Listagem 4.8: Função para calcular a curva ROC e a respetiva AUC (parte 2).

A tabela 2 apresenta os resultados obtidos para cada um dos modelos implementados e para ambos os conjuntos de dados 2D.

Analisando a tabela 2 de resultados podemos tirar várias conclusões. Em primeiro lugar, em quase todos os modelos é notória a dificuldade em classificar corretamente a classe MCI.

	Conjunto de dados simples			Conjunto de dados complexo		
	CN	**MCI**	**AD**	**CN**	**MCI**	**AD**
DenseNet121	0.64	0.66	0.66	0.71	0.64	0.62
DenseNet169	0.60	0.63	0.64	0.72	0.58	0.56
DenseNet201	0.64	0.67	0.59	0.76	0.60	0.67
Inceptionv3	0.65	0.62	0.66	0.67	0.53	0.53
InceptionResNetV2	0.64	0.54	0.73	0.70	0.51	0.56
MobileNet	0.66	0.57	0.68	0.75	0.61	0.63
MobileNetV2	0.70	0.58	0.67	0.76	0.62	0.63
ResNet50	0.70	0.62	0.64	0.69	0.57	0.63
ResNet101	0.73	0.70	0.67	0.72	0.57	0.56
ResNet152	0.74	0.65	0.67	0.67	0.54	0.68
Xception	0.57	0.56	0.62	0.69	0.61	0.55

Tabela 2: Resultados obtidos com as CNNs 2D.

Na maioria dos casos, esta classe é a que possui menor valor de AUC, sendo que apenas para a CNN `DenseNet201` esta classe possui um valor de AUC superior às demais classes.

Ao comparar os resultados para os diferentes conjuntos de dados, simples e complexo, pode concluir-se que não há uma melhoria significativa nos valores de AUC da classe MCI - o principal objetivo desta investigação no que toca à tarefa de classificação - para o conjunto de dados complexo, quando comparado com o conjunto de dados simples. No entanto, genericamente houve uma ligeira melhoria na AUC da classe CN, mas em contrapartida houve uma ligeira descida nos valores da AUC para a classe AD.

A escolher um modelo entre os treinados até esta fase, a escolha seria o `ResNet101` treinado com o conjunto de dados simples, pois apresenta o melhor resultado de AUC para a classe MCI e valores aceitáveis para as restantes classes. O gráfico das curvas ROC para este modelo pode ser consultado na figura 43.

Convém notar que estes resultados não se aproximam dos apresentados na bibliografia atual, mas isto é algo que já se estaria à espera a partir do momento em que se decidiu utilizar apenas uma imagem por paciente que acabou por reduzir drasticamente o tamanho do conjunto de dados. Este é um procedimento diferente do que se encontra na bibliografia, contudo, com a convição que desta forma se atingem resultados que melhor demonstram a capacidade dos algoritmos. Uma forma de melhorar os resultados seria reduzir a classificação de três para duas classes, concretamente, passar o problema para uma classificação binária com as classes AD e CN.

4.3 Modelos 3D

Depois do treino dos modelos com os conjuntos de dados 2D, foram treinadas diversas CNNs 3D com os conjuntos de dados simples e complexo tridimensionais criados. Os

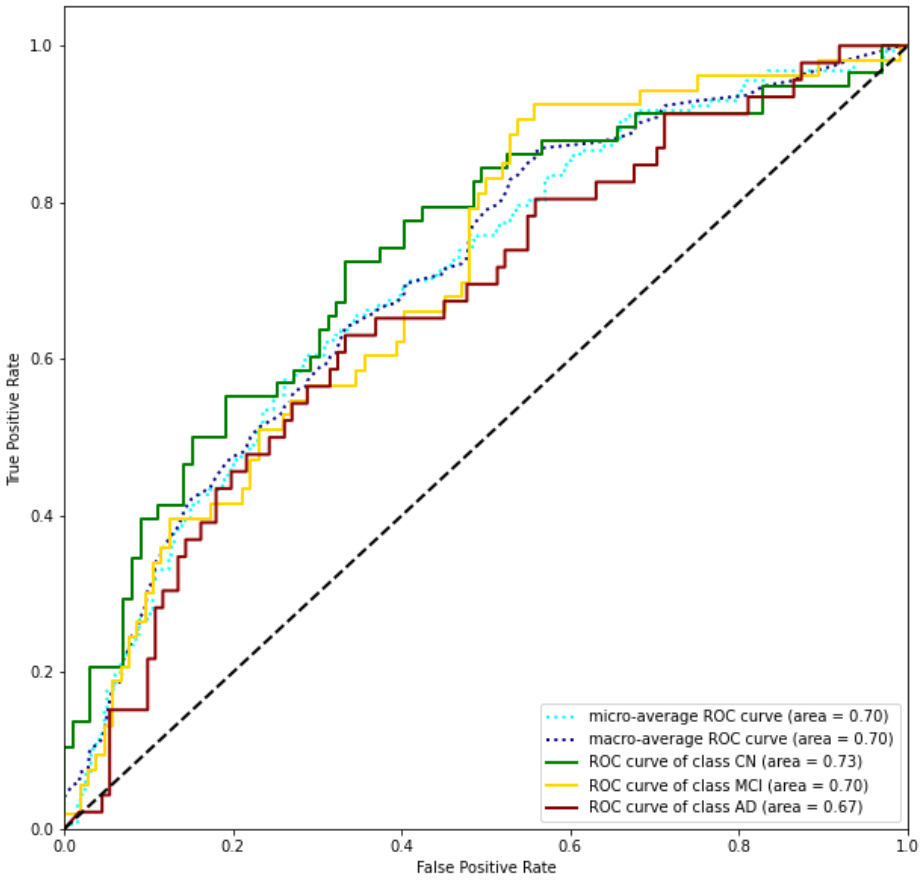


Figura 43: Gráfico ROC da CNN 2D `ResNet101` treinada com o conjunto de dados 2D simples.

modelos treinados são variantes 3D das populares CNNs 2D usadas em 4.2. Os modelos 3D foram desenvolvidos por Roman Solovyev (Solovyev et al., 2022), foram desenvolvidos em Keras e a sua implementação pode ser encontrada no seguinte repositório do GitHub https://github.com/ZFTurbo/classification_models_3D.

No trabalho presente foram treinados os modelos `DenseNet121`, `DenseNet169`, `DenseNet201`, `InceptionResNetV2`, `InceptionV3`, `ResNet18`, `ResNet34`, `ResNet50`, `ResNet101`, `SeNet154`, `SeResNet50` e `SeResNext50`. Mesmo antes de treinar estes modelos 3D já se previa que o treino fosse muito mais lento que o dos modelos 2D pois, para além do número de parâmetros ser superior nestes modelos, os dados de entrada são imagens 3D. Tanto os modelos 2D, mas ainda mais nos modelos 3D, o número de imagens disponíveis para o treino fica aquém do desejável. Isto ocorre porque quanto mais complexa é uma rede e quanto mais informação tiver que processar, maior é o número de observações que precisa para se otimizar os pesos e viés.

Mais uma vez, e não sendo o cenário perfeito como referido anteriormente, todos os modelos foram treinados seguindo o mesmo procedimento e perante as mesmas condições e hiperparâmetros. Ou seja, todos os modelos foram treinados ao longo de 300 épocas,

com o otimizador `Adam` e com uma taxa de aprendizagem inicial de 0.0001. O tamanho de *batch* utilizado foi 2 para todas as redes treinadas com o conjunto de dados 3D simples e foi 1 para as redes treinadas com o conjunto de dados 3D complexo, exceto na `ResNet18`, `ResNet34` e `ResNet50` onde foi possível treinar com um tamanho de *batch* igual a 2. Foram ainda utilizadas a *categorical cross-entropy* como função de perda e a acurácia como métrica de desempenho auxiliar.

O código desenvolvido para carregar os dados e instanciar os modelos pré-treinados é semelhante ao apresentado anteriormente, assim como as funções que criam os gráficos da acurácia e da perda para o treino e validação. O código inclui ainda o cálculo da curva ROC. O código referente a esta tarefa encontra-se na listagem 4.9, utilizando como exemplo o modelo `ResNet50` 3D.

```
# instanciar o modelo
ResNet50, preprocess_input = Classifiers.get('resnet50')

model = ResNet50(input_shape=(91, 109, 91, 1), weights=None, classes=N_CLASSES)

optimizer = tf.keras.optimizers.Adam(lr=0.0001)
model.compile(optimizer=optimizer,
              loss='categorical_crossentropy',
              metrics=['acc'])


# treinar o modelo
history = model.fit(train_dataset,
                    epochs=300, steps_per_epoch=STEPS_PER_EPOCH,
                    validation_data=val_tensor,
                    validation_steps=VALIDATION_STEPS,
                    )
```

Listagem 4.9: Instanciar e treinar o modelo `ResNet50` 3D.

As figuras 44 e 45 mostram a evolução da acurácia e da perda, respetivamente para os modelos `ResNet18` 3D e `SeResNext50` 3D, durante o treino e validação com o conjunto de dados 3D simples.

Por seu lado, as figuras 46 e 47 mostram a evolução da acurácia e da perda, respetivamente para os modelos `ResNet18` 3D e `ResNet150` 3D, durante o treino e validação com o conjunto de dados 3D complexo.

Analisando as figuras 44, 45, 46 e 47 e comparando com as figuras 39, 40, 41 e 42, é possível identificar uma diminuição da oscilação dos valores da acurácia e da perda. Tal como aconteceu no ajuste fino dos modelos 2D, observa-se que durante o treino as curvas apresentam um padrão mais suavizado. Uma vez mais, a AUC foi a métrica utilizada para avaliar os diversos modelos.

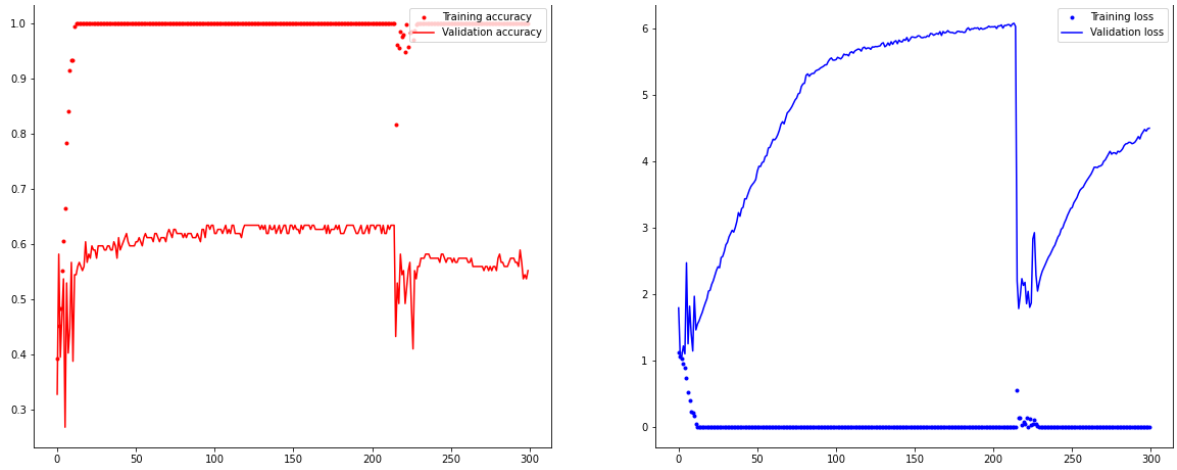


Figura 44: Treino da `ResNet18` 3D com o conjunto de dados 3D simples.

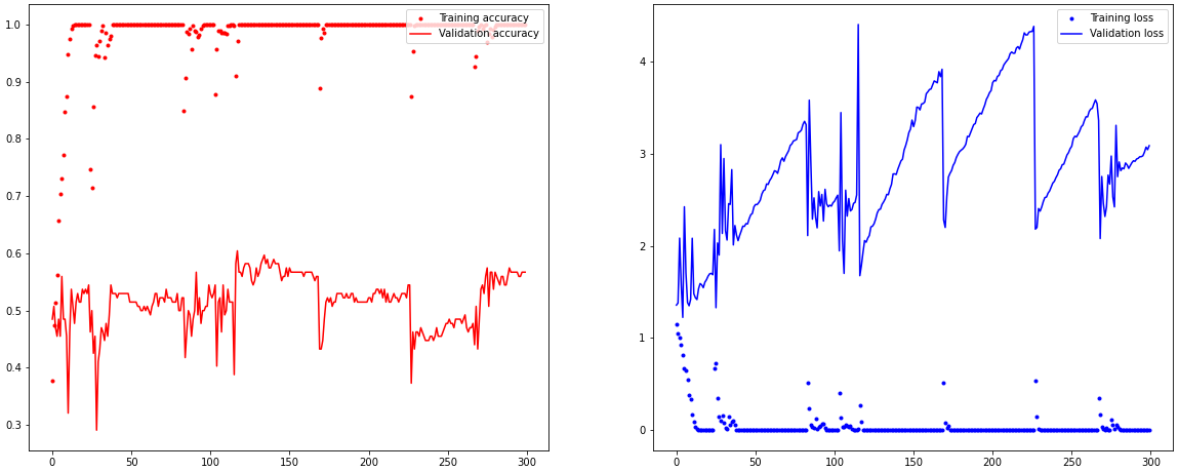


Figura 45: Treino da `SeResNext50` 3D com o conjunto de dados 3D simples.

A tabela 3 apresenta os resultados obtidos para cada um dos modelos e para ambos os conjuntos de dados 3D.

A análise da tabela 3 revela a ausência de resultados para os modelos `SeNet154` e `SeResNext50`. Isto deveu-se ao facto de não se dispor de recursos computacionais adequados para treinar estes modelos. Em quase todos os modelos, e tal com nos modelos 2D, também aqui se nota uma dificuldade em classificar corretamente as amostras da classe MCI quando comparamos com as outras duas classes.

A comparação de resultados para ambos os conjuntos de dados 3D, simples e complexo, pode concluir-se que os valores de AUC para a classe MCI são inferiores, enquanto que nas demais classes não existe uma melhoria que justifique a utilização do conjunto de dados complexo em detrimento do conjunto de dados simples. Comparando estes resultados com

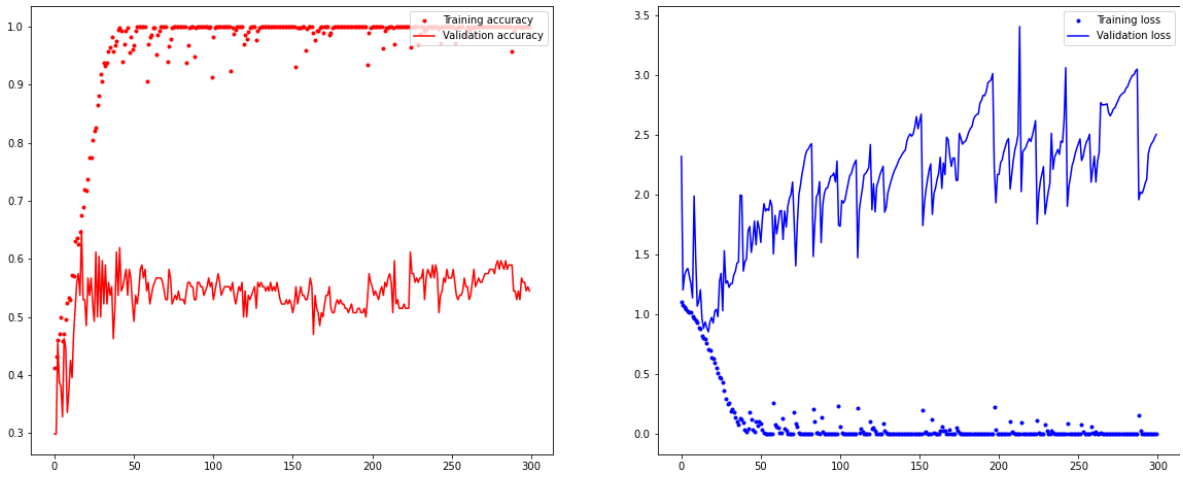


Figura 46: Treino da `ResNet18` 3D com o conjunto de dados 3D complexo.

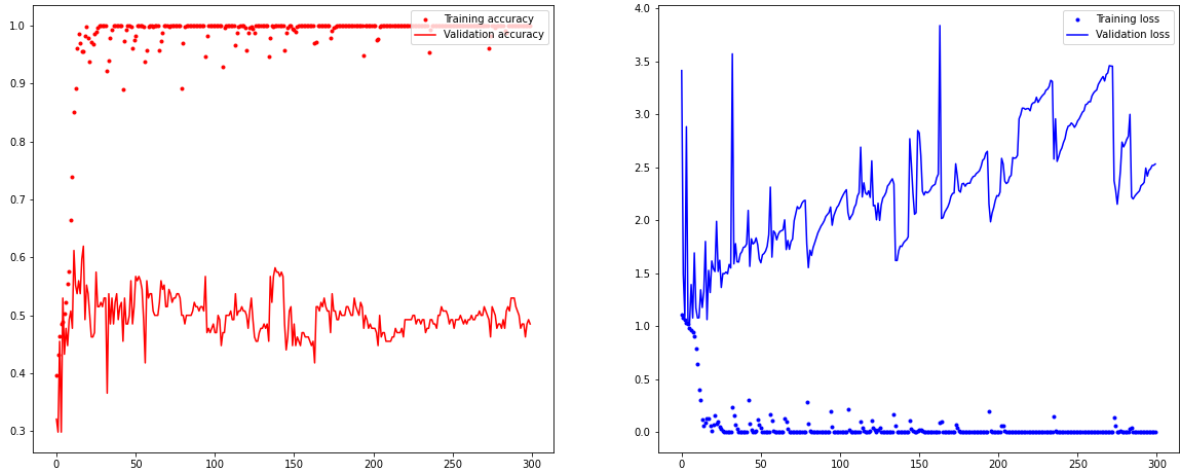


Figura 47: Treino da `ResNet50` 3D com o conjunto de dados 3D complexo.

os obtidos anteriormente nos modelos 2D, é possível observar que para o conjunto de dados 3D simples os modelos 3D apresentam melhores resultados em todas as classes.

A ter que selecionar um modelo entre os avaliados, a escolha recairia na `SeResNext50` treinada com o conjunto de dados simples, pois apresenta o melhor resultado de AUC para a classe MCI e valores bons para as restantes classes. O gráfico das curvas ROC para este modelo encontram-se na figura 48.

4.4 AUMENTO DE DADOS

Tal como foi referido na secção 2.12 o aumento de dados é uma técnica bastante utilizada para aumentar o conjunto de dados de treino e consequentemente reduzir o *overfitting*. Neste trabalho foram aplicados métodos tradicionais de aumento de dados, ou seja, foram

	Conjunto de dados simples			Conjunto de dados complexo		
	CN	**MCI**	**AD**	**CN**	**MCI**	**AD**
DenseNet121 3D	0.76	0.63	0.74	0.76	0.59	0.77
DenseNet169 3D	075	0.62	0.75	0.74	0.57	0.72
DenseNet201 3D	0.76	0.63	0.72	0.42	0.51	0.47
InceptionResNetV2 3D	0.76	0.66	0.66	0.67	0.54	0.75
InceptionV3 3D	0.37	0.68	0.53	0.48	0.34	0.47
ResNet18 3D	0.78	0.65	0.73	0.78	0.65	0.75
ResNet34 3D	0.77	0.65	0.74	0.69	0.55	0.64
ResNet50 3D	0.74	0.64	0.76	0.76	0.62	0.63
ResNet101 3D	0.71	0.65	0.66	0.65	0.51	0.59
SeNet154 3D	0.70	0.69	0.61	-	-	-
SeResNet50 3D	0.75	0.67	0.75	0.82	0.64	0.75
SeResNext50 3D	0.76	0.70	0.70	-	-	-

Tabela 3: Resultados obtidos com as CNNs 3D.

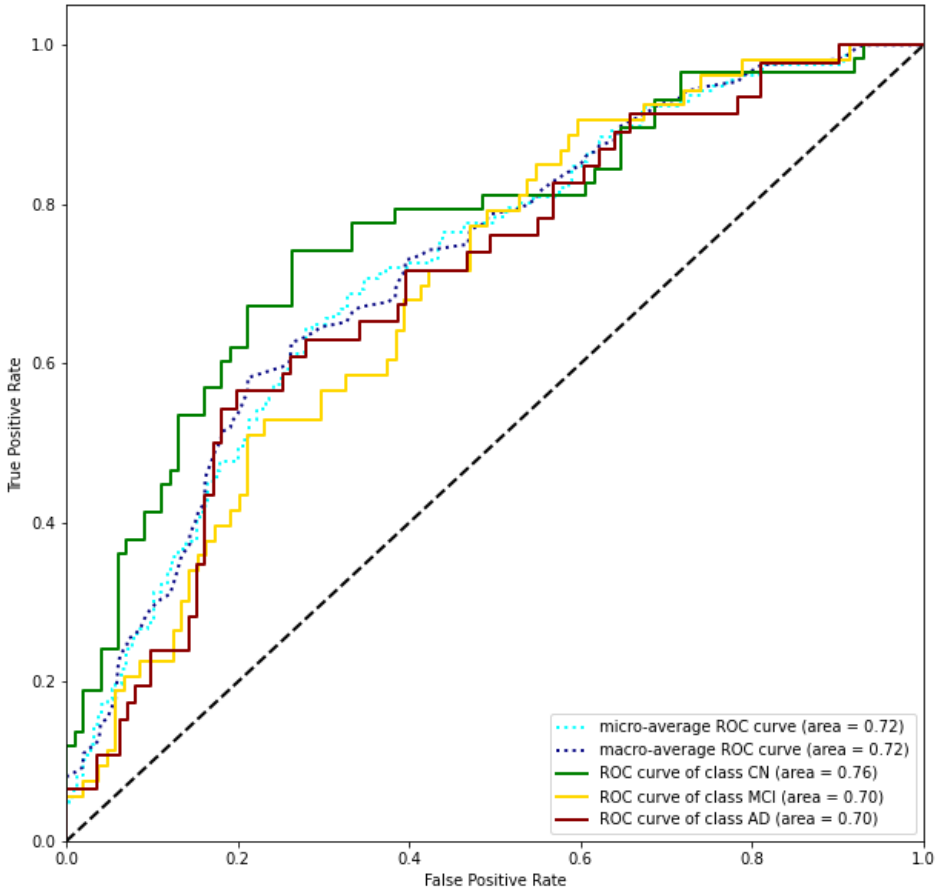


Figura 48: Curvas ROC da `SeResNext50` 3D treinada com o conjunto de dados 3D simples.

aplicadas algumas operações que alteram as imagens originais, como por exemplo rotações e deslocamentos, e desta forma adicionar as imagens resultantes ao conjunto de dados

de treino. O principal objetivo passa por tornar o modelo mais tolerante às variações adicionadas às imagens originais. A combinação das diversas transformações permite aumentar consideravelmente o tamanho do conjunto de dados de treino.

Justifica-se contudo mencionar que aumentar o número de amostras com variantes de uma mesma amostra original contraria de certa forma alguns princípios adotados no início do trabalho, nomeadamente a utilização de apenas uma amostra por paciente e o registo das imagens realizado durante a fase de pré-processamento. O aumento de dados gera novas imagens, aplicando por exemplo ligeiras rotações à imagem original, o que é o oposto do que se pretendia com o registo das imagens, ou seja, remover as diferenças nas imagens que resultavam por exemplo de a posição/ângulo da cabeça do paciente ser diferente.

Para aplicar o aumento de dados foi escolhido o conjunto de dados 3D simples, pois foi com ele que se obtiveram os melhores resultados. Para aplicar esta técnica foi adotada a biblioteca `volumentations-3D` (Solovyev et al., 2022), a qual permite aplicar 25 tipos de transformações às imagens 3D. Algumas destas transformações não fazem sentido no contexto de imagens MRI do cérebro, como por exemplo a rotação de 90º de acordo com um determinado eixo. Por isso, foi feita uma seleção de quatro transformações para aplicar: adição de ruído Gaussiano, transformação elástica, rotação e *grid dropout*. O código da implementação do aumento de dados pode ser consultado na listagem 4.10.

O ruído Gaussiano é um ruído estatístico com uma função de densidade de probabilidade igual à distribuição normal, também conhecida como Distribuição Gaussiana. A transformação elástica consiste em alterar a forma de um material, neste caso de uma imagem, através da aplicação de uma força dentro do limite elástico desse material. Tal como o nome indica, uma rotação consiste em rodar os eixos da imagem. Para as rotações foi definido que a cada eixo pode ser aplicada uma rotação com um valor aleatório entre -15º e 15º. A transformação *grid dropout* aplica cortes à imagem, ou seja, são removidas partes da imagem.

Para cada imagem original do conjunto de dados de treino foram criadas 6 novas imagens, existindo uma probabilidade igual a 50% de se aplicar cada uma das 4 transformações na geração de uma nova imagem. Depois de aplicar as transformações o conjunto de dados, que anteriormente era composto por 760 imagens, passou a ter 4560 imagens. Na figura 49 podemos ver quatro fatias de uma imagem 3D antes e depois de serem aplicadas transformações. As fatias da imagem original encontram-se na linha superior da figura, as imagens incluídas na linha central foram criadas adicionado ruído, e as imagens incluídas na linha inferior foram obtidas aplicando cortes e rotações.

Após criar o novo conjunto de dados de treino, foi treinado modelo `SeResNext50` 3D, dado que foi o que obteve melhores resultados nos testes anteriores. No que toca às condições em que este treino foi realizado, adotaram-se as mesmas condições e hiperparâmetros utilizados anteriormente, ou seja, a rede neuronal foi treinada ao longo de 200 épocas, com o otimizador `Adam` e uma taxa de aprendizagem inicial de 0.0001. Utilizou-se um tamanho

de *batch* igual a 2, a função de perda foi a *categorical cross-entropy* e a métrica de desempenho auxiliar usada foi a acurácia. Como seria de esperar, a AUC foi a métrica utilizada para avaliar o modelo. A figura 50 mostra a evolução da acurácia e da perda durante o treino e validação do modelo `SeResNext50` 3D.

```
from volumentations import Compose
from volumentations import augmentations as ai

def build_CustomMRI():
    # Inicializar a lista de transformacoes
    transforms = []
    # Preencher a lista de transformacoes
    tf = ai.GaussianNoise(var_limit=(30.0, 60.0), p=0.5)
    transforms.append(tf)
    tf = ai.ElasticTransform(p=0.5)
    transforms.append(tf)
    tf = ai.Rotate(p=0.5)
    transforms.append(tf)
    tf = ai.GridDropout(p=0.5)
    transforms.append(tf)
    return Compose(transforms)

writer = tf.io.TFRecordWriter(train_aug_tfrec)

for record in tf.compat.v1.io.tf_record_iterator(training_tfrec):
    one_rec = decode(record)
    img = one_rec[0].numpy()
    img = np.squeeze(img, axis=3)
    feature = {'label': _int64_feature(one_rec[1]), 'image': _float_feature(img.ravel())}
    example = tf.train.Example(features=tf.train.Features(feature=feature))
    writer.write(example.SerializeToString())
    for x in range(5):
        # Inicializar as Volumentations
        data_aug = build_CustomMRI()
        # Aplicar as transformacoes
        img_augmented = data_aug(image=img)["image"]
        # Criar uma nova 'feature'
        feature = {'label': _int64_feature(one_rec[1]), 'image': _float_feature(img_augmented.
    ravel())}
        example = tf.train.Example(features=tf.train.Features(feature=feature))
        # Adicionar a 'feature' ao novo TFRecord
        writer.write(example.SerializeToString())

writer.close()
```

Listagem 4.10: Implementação do aumento de dados.

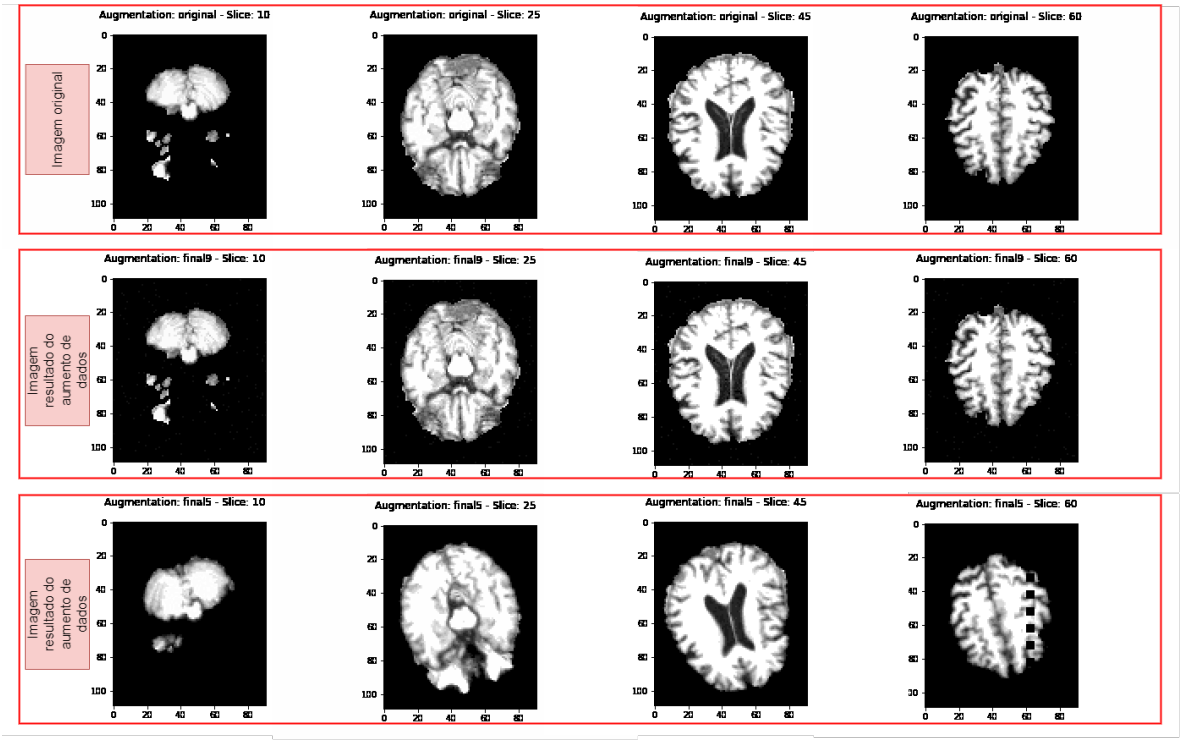


Figura 49: Aplicação de aumento de dados a MRIs.

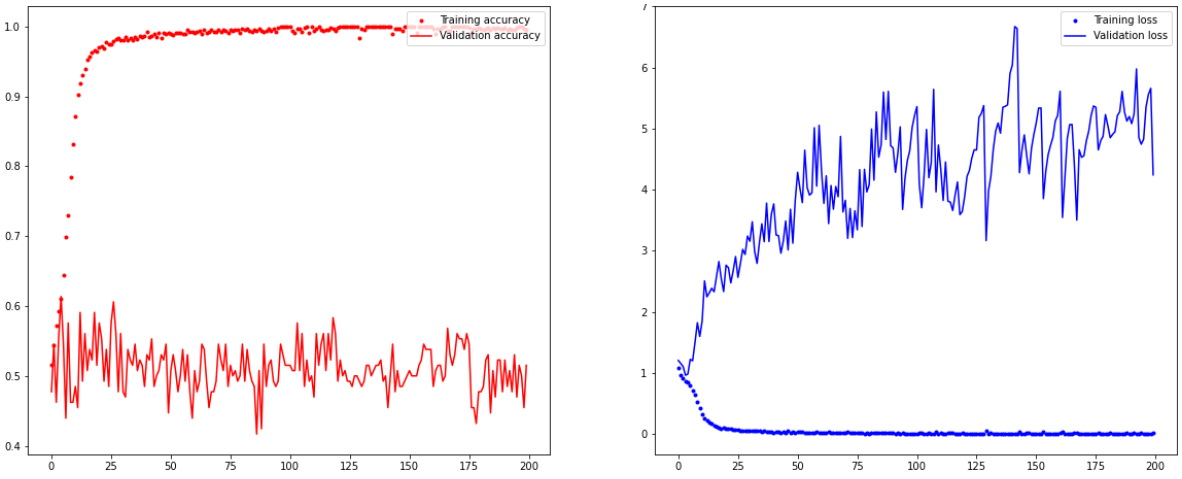


Figura 50: Treino da `SeResNext50` 3D com o conjunto de dados 3D simples aumentado.

O gráfico das curvas ROC para este modelo pode ser visto na figura 51. Podemos ver que os valores da AUC para cada uma das classes são 0.70 para a classe AD, 0.65 para a classe MCI e 0.73 para a classe CN.

Comparando estes resultados com os obtidos anteriormente pelo mesmo modelo, treinado sem aumento de dados, conclui-se que os valores da AUC desceram para as classes CN e

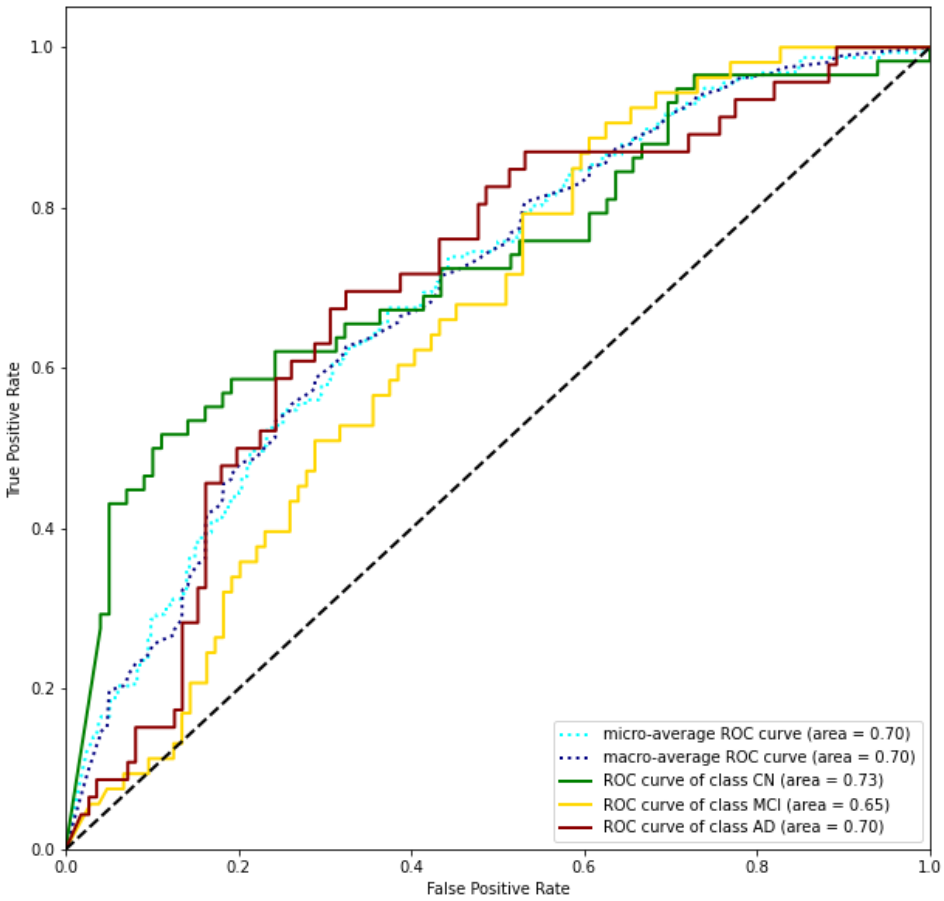


Figura 51: Curvas ROC da `SeResNext50` 3D treinada com o conjunto de dados 3D simples aumentado.

MCI e manteve-se para a classe AD. Globalmente, conclui-se que este modelo apresenta resultados piores.

4.5 Análise de Resultados

A análise dos resultados foi sendo incluída no documento à medida que os resultados foram apresentados. Esta seção apresenta um breve resumo e uma análise global dos resultados parcelares obtidos. De forma geral, os resultados apresentados neste trabalho são inferiores aos documentados na literatura, contudo esta conclusão era esperada a partir do momento que se tomou a decisão de reduzir drasticamente o tamanho do conjunto de dados.

Na maior parte dos casos, os modelos 3D apresentaram melhores resultados do que os modelos 2D correspondentes. Se por um lado seria expectável que isso acontecesse devido à utilização da informação completa das imagens, por outro lado, seria também de esperar que fossem necessários mais dados para treinar as CNNs 3D. No entanto, de-se referir que não há grande diferença entre os melhores resultados obtidos com modelos 3D e os melhores

obtidos com modelos 2D. Este resultado não deixa de ser surpreendente, uma vez que o algoritmo de conversão de imagens 3D para 2D não recorre a qualquer critério clínico ou científico, simplesmente extrai planos horizontais localizados à mesma distância uns dos outros, dentro de um intervalo estimado manualmente.

Outro resultado a destacar é a maior dificuldade dos modelos em classificar corretamente as amostras da classe MCI. Em certa medida, esta conclusão também era expectável dado a doença de Alzheimer ser de espectro contínuo e a desordem MCI ser uma fase intermédia na progressão da doença.

Por fim, o aumento de dados não trouxe melhorias aos resultados, o que indica que a técnica aplicada produziu imagens muito correlacionadas com as originais.

Embora seja verdade que os resultados apresentados ficam abaixo dos presentes na literatura atual, devido em primeiro lugar ao tamanho do conjunto de dados utilizado, deve ter-se em consideração que foi seguida uma abordagem baseada na simplicidade de processos. Os procedimentos de registo das imagens MRIs e de extração do crânio, realizados durante o pré-processamento, não foram muito sofisticados, tendo-se optado por utilizar uma ferramentas de código aberto. Por exemplo a simplificação do processo de extração do crânio, onde se poderiam obter melhorias consideráveis na qualidade da informação de treino, pode ter sido particularmente penalizador. Aqui, em vez de se usar um *fractional intensity threshold* fixo para todas as imagens, poderia ter-se utilizado um valor ajustado a cada imagem individual. Conclui-se que a utilização de imagens MRI, em conjunto com modelos de DL, é uma opção viável para diagnosticar a doença de Alzheimer, devendo por isso ser mais explorada, especialmente à medida que mais dados estejam disponíveis para uso geral dos investigadores.

Apesar dos resultados não serem extraordinários, deve-se destacar o modelo `SeResNext50` 3D, treinado com o conjunto de dados 3D simples, por ter sido o melhor a identificar o Défice Cognitivo Ligeiro, obtendo uma AUC igual a 0.70.

4.6 RECOMENDAÇÕES

Concluída a análise geral dos resultados, pode-se concretizar o último objetivo deste trabalho, elaborar um conjunto de recomendações para um cientista de dados sem conhecimentos médicos profundos e que pretenda construir um sistema de diagnóstico CADx da DA.

Para começar, há a mencionar que a utilização de CNNs é uma opção muito válida. No capítulo 2, foi mostrada a superioridade destes modelos em relação a outras opções, principalmente em relação aos tradicionais métodos de ML, tais como as SVMs. De facto, em geral, as CNNs são responsáveis por quase todas as aplicações na análise da imagem médica, e não apenas no diagnóstico da DA.

No que toca ao conjunto de dados, se for tomada uma decisão radical como a que adotou nesta dissertação e que consistiu em utilizar apenas uma imagem por paciente, o conjunto de dados pode ficar demasiado pequeno e insuficiente para construir bons modelos. Contudo, considera-se que utilizar mais de uma imagem do mesmo paciente, especialmente quando as recolhas são feitas apenas a intervalos de meses, não é uma boa prática e produz resultados de treino melhores do que aqueles que se obterão quando o modelo estiver a ser utilizado num cenário não laboratorial.

O pré-processamento das imagens exige experimentar e afinar várias técnicas, e as funcionalidades de pré-processamento disponibilizadas pelas ferramentas de código aberto nem sempre estão otimizadas. Por exemplo para extrair o crânio, onde seria aconselhável tratar cada imagem de forma personalizada, para o *fractional intensity threshold* (no caso da FSL BET) mais adequado a cada imagem. O problema é que a única forma de verificar se o valor do *fractional intensity threshold* é o mais correto é observar diretamente o resultado da extração, algo que não é exequível em conjuntos de imagens além de pequenos. Por outro lado, as ferramentas de registo de imagens funcionam bastante bem e são suficientes para tratar as imagens automaticamente, desde que estejam registadas num atlas da mesma modalidade (T1 ponderada, no caso deste trabalho) e todas as imagens sejam normalizadas com a mesma resolução espacial.

Quanto maior for a resolução espacial, maiores são as imagens e menos informação é representada por cada vóxel, o que sugere que uma resolução espacial superior seria melhor. Contudo, apesar de gerar imagens maiores e tornar o treino dos modelos mais lento, vimos que os modelos treinados com as imagens resultantes da aplicação de uma resolução isotrópica maior ($1mm^3$) não produzem melhores resultados do que os modelos treinados com as imagens resultantes da aplicação de uma resolução espacial inferior ($2mm^3$). Portanto, a recomendação passa por não usar resoluções espaciais muito grandes se não estiver ao dispor poder computacional para treinar os modelos com imagens de grandes dimensões.

No que diz respeito à opção entre imagens 2D ou imagens 3D, não há uma recomendação a ser feita. Apesar de nesta investigação os modelos treinados com imagens 3D terem obtido resultados ligeiramente superiores, há trabalhos onde acontece o oposto (Darias Plasencia, 2019). Por isso, se possível devem ser testadas as duas alternativas. Caso se esteja num cenário de poder computacional e tempo de cálculo muito limitados, sem dúvida que a opção deve recair nas imagens 2D pois, para além de serem imagens mais pequenas, as CNNs 2D são muito mais simples e treinam muito mais rapidamente e existe uma grande variedade de modelos avançados sobre os quais se pode aplicar o ajuste fino.

Em termos de tipo de imagem a utilizar, a recomendação vai para as imagens MRI. Apesar dos trabalhos que utilizam PET apresentarem também bons resultados e de não ser necessária a remoção do crânio das imagens, a sua obtenção é muito mais cara e prejudicial

para o paciente. Em contraste, as imagens MRI são exames baratos, inofensivos para os pacientes, mas requerem um pré-processamento mais cuidadoso.

Relativamente ao treino dos modelos, avaliar muitos modelos 2D e 3D como se fez nesta dissertação, pode ser contraproducente porque alarga o espaço de procura pelos melhores hiperparâmetros. A recomendação passa por selecionar um conjunto pequeno de modelos e fazer a procura dos melhores hiperparâmetros para os mesmos, por forma a extrair os melhores resultados que cada um podem oferecer.

Fica também em aberto a questão se se deve realizar algum pré-treino recorrendo a *autoencoders* convolucionais, uma técnica ligeiramente mencionada na seção 2.5. Esta abordagem é referida em alguma literatura, mas a recomendação passa por a usar apenas se dispusermos de um conjunto de dados muito grande e não etiquetado. Neste caso, o *autoencoder* permitiria extrair características úteis das imagens que seriam utilizadas para as etiquetar, podendo as imagens etiquetadas ser depois utilizadas num treino supervisionado de outros modelos classificadores. Este não era o cenário da presente dissertação.

Por fim, deve ser realçado que para obter melhores resultados poderia ter-se reduzido a classificação a uma classificação binária entre os grupos AD e CN. Contudo, esta simplificação não permitiria alcançar o principal objetivo da dissertação, a deteção precoce da DA. Por isso, deve optar-se por uma classificação num número de classes que esteja de acordo com os objetivos da investigação.

5

CONCLUSÃO

As conclusões retiradas do trabalho realizado são resumidas de seguida com base nos objetivos definidos em 1.2. Estes objetivos foram sendo abordados ao longo do documento.

O primeiro objetivo passava por compreender os principais conceitos relacionados com o tema em investigação e foi concretizado através da leitura de artigos de revista, leitura de livros e artigos publicados na Internet.

O próximo passo consistiu em encontrar os modelos mais relevantes do estado da arte. Esta tarefa foi abordada em detalhe no capítulo 2, concluindo-se que as CNNs são a escolha natural, com apenas havendo variações na forma como são treinadas. A este respeito, três métodos predominam na literatura: (1) treino direto das redes, utilizando inicialização de pesos com métodos clássicos, sem pré-treino; (2) treino direto, com inicialização de pesos com métodos clássicos, mas utilizando *autoencoders* para pré-treinar as redes. Esta opção foi descartada para atividades posteriores devido à sua perda de popularidade e ao facto de normalmente não demonstrar melhores resultados do que a opção anterior; (3) utilização de redes pré-treinadas, o que requer um pré-processamento das imagens para as converter em duas dimensões. São utilizadas arquiteturas que tenham tido sucesso noutras tarefas, tais como no concurso *ImageNet Large Scale Visual Recognition Challenge*.

A recolha dos dados, assim como o seu pré-processamento, foram objetivos bem concretizados e explicados em detalhe no capítulo 3. O pré-processamento consistiu no regido das imagens utilizando uma abordagem simples e uma ferramentas de código abertos, tendo-se revelado uma tarefa relativamente complexa.

Um dos principais objetivos desta dissertação consistia em treinar e avaliar modelos, com o objetivo de comprovar que a sua implementação é possível por profissionais sem extensos e profundos conhecimentos médicos. De entre as três abordagens de treino de redes mencionadas foram utilizadas a primeira e terceira abordagens para o treino com os conjuntos de dados 3D e para os conjuntos de dados 2D criados, respetivamente. Estas são as abordagens mais comuns e mais simples de implementar e não diferem muito dos procedimentos seguidos noutros domínios de aplicação. A obtenção de resultados excelentes é uma tarefa bastante complicada devido à natureza da imagiologia médica e da própria DA. Portanto, este objetivo passava apenas por implementar os modelos selecionados num

cenário de prototipagem para testar a viabilidade da sua implementação por profissionais sem conhecimentos médicos profundos. Este objetivo foi tratado ao longo do capítulo 4 e demonstrou-se que a implementação dos modelos utilizando as ferramentas disponíveis é viável e consegue criar-se bons modelos, mesmo sem possuir um conhecimento profundo da doença de Alzheimer.

Os procedimentos adotados para implementar os modelos foram os comummente usados nas diversas áreas de aplicação de DL, não exigindo grandes conhecimentos médicos. Contudo, para efetuar corretamente o pré-processamento dos dados é necessário adquirir alguns conhecimentos sobre imagiologia médica para utilizar as ferramentas de registo de imagens médicas e para implementar a extração do crânio.

O objetivo seguinte consistiu na análise dos resultados obtidos e na escolha do melhor modelo para tratar o problema em análise. Em primeiro lugar, destaca-se o superior desempenho das CNNs 3D em relação às CNNs 2D. No que toca aos modelos destaca-se a CNN 3D *ResNext50* treinada com o conjunto de dados 3D simples como a escolhida entre os modelos treinados e avaliados.

No passo seguinte aplicaram-se técnicas para melhorar o desempenho do modelo selecionado, concretamente o aumento de dados. Foram aplicadas estratégias habituais para aumentar o tamanho do conjunto de dados de treino do conjunto de dados 3D simples, como adição de ruído, rotações e cortes das imagens originais. Após treinar o modelo *ResNext50* 3D com aumento de dados não se notou melhoria nos resultados, pelo contrário diminuíram ligeiramente. Isto indica que o processo de aumento de dados produziu fundamentalmente imagens muito correlacionadas que não trouxeram nada de novo no que toca à aprendizagem de características por parte da rede.

Por fim, era proposto elaborar um conjunto de recomendações para profissionais que pretendam implementar sistemas CADx para a DA, assunto abordado em detalhe na seção 4.6. Podem destacar-se como principais recomendações a utilização de imagens MRI, selecionar um método de extração do crânio que se adapte às várias imagens processadas e, em caso de limitações de capacidade computacional, optar por processar as imagens com uma resolução isotópica inferior e utilizar modelos 2D.

Tendo em conta a sensibilidade deste domínio de aplicação, e de acordo com as experiências realizadas neste trabalho, a introdução destes modelos em produção requer cuidados redobrados. Resultados de AUC da ordem de 0.7 não são suficientes para desenvolver aplicações de diagnóstico médico completamente automáticas, mas funcionam bem como auxiliares do diagnóstico manual por profissionais médicos. Fica a crença que à medida que mais imagens etiquetadas forem ficando disponíveis, valores mais elevados de AUC serão alcançados. O desempenho dos modelos também pode ser avaliado de outra forma não absoluta. Com base na ideia de que a utilização da IA no domínio médico visa reduzir o erro humano, poder-se-ia avaliar o desempenho dum modelo por comparação com o

dos profissionais médicos. Neste cenário, um modelo passa a ser válido desde que o seu desempenho iguale ou supere o dos humanos.

5.1 TRABALHO FUTURO

A análise de imagens médicas com técnicas de IA, principalmente com modelos de DL, para diagnosticar a DA é um campo muito amplo e complexo que oferece muito espaço para explorar e investigar. Em particular, a investigação aqui realizada levanta uma série de tópicos que merecem ser abordados em trabalhos futuros.

Em primeiro lugar seria interessante realizar uma extração do crânio mais cuidadosa do que aquela realizada nesta dissertação, principalmente no que toca ao limiar de intensidade fracionária a aplicar a cada imagem individual. Com uma extração individualizada para cada imagem acabaríamos por obter um conjunto de dados mais limpo.

Uma outra linha de investigação poderia ser a recolha de imagens de MRI sem etiquetas para criar um conjunto de dados maior. Com esse conjunto de dados poderia realizar-se pré-treino de uma CNN, utilizando *autoencoders* convolucionais.

Outra extensão possível ao trabalho realizado seria a utilização de outras técnicas de aumento de dados além das aplicadas. Seria interessante comparar os resultados obtidos por diferentes técnicas de aumentos de dados e até mesmo utilizar as diversas técnicas para criação de um único conjunto de dados de treino mais complexo.

As imagens utilizadas foram obtidas com scanners de 1,5 Tesla. Mesmo a base de dados ADNI já possui imagens 3T, embora que em número muito inferior. O valor em Tesla determina a resolução com que as imagens podem ser obtidas, existindo scanners com 7T, 10T e mesmo 21,1T. Por conseguinte, este trabalho poderia ser atualizado recorrente a imagens de maior resolução. É importante notar que uma resolução mais alta fornece mais detalhe, mas também torna o treino mais lento.

Por fim, é importante ter em conta os avanços de DL. Esta é uma área em constante evolução, pelo que isso deve ser tido em consideração quando se começa um trabalho de dissertação. São constantemente publicadas novas técnicas e novas arquiteturas, que melhoram as anteriores, que permitem o treino de redes mais profundas, com convergência mais rápida e que atingem melhores resultados.

BIBLIOGRAFIA

Josh Patterson Adam Gibson. *Deep Learning: A Practitioner's Approach*. OREILLY MEDIA, 2017. ISBN 1491914254. URL https://www.ebook.de/de/product/23640784/adam_gibson_josh_patterson_deep_learning_a_practitioner_s_approach.html.

Tameem Adel, Taco Cohen, Matthan Caan, and Max Welling. 3d scattering transforms for disease classification in neuroimaging. *NeuroImage: Clinical*, 14:506–517, 2017. doi: 10.1016/j.nicl.2017.02.004.

Sanjeev Arora, Aditya Bhaskara, Rong Ge, and Tengyu Ma. Provable bounds for learning some deep representations. 2013.

Yaniv Bar, Idit Diamant, Lior Wolf, and Hayit Greenspan. Deep learning with non-medical training used for chest pathology identification. In Lubomir M. Hadjiiski and Georgia D. Tourassi, editors, *SPIE Proceedings*. SPIE, mar 2015. doi: 10.1117/12.2083124.

Silvia Basaia, Federica Agosta, Luca Wagner, Elisa Canu, Giuseppe Magnani, Roberto Santangelo, and Massimo Filippi. Automated classification of alzheimer's disease and mild cognitive impairment using a single MRI and deep neural networks. *NeuroImage: Clinical*, 21:101645, 2019. doi: 10.1016/j.nicl.2018.101645.

Pushpa B.R, Nayana.P.Kamal, and Amal P.S. A comparative study on different segmentation and classification methods for the diagnosis of alzheimer disease. *International Journal of Research in Pharmaceutical Sciences*, 10(3):2058–2070, jul 2019. doi: 10.26452/ijrps.v10i3.1422.

Nikhil Buduma. *Fundamentals of deep learning : designing next-generation machine intelligence algorithms*. O'Reilly Media, 1st edition, 2017. ISBN 9781491925614.

Davide Castelvecchi. Can we open the black box of AI? *Nature*, 538(7623):20–23, oct 2016. doi: 10.1038/538020a.

François Chollet. Xception: Deep learning with depthwise separable convolutions. 2016.

Óscar Darias Plasencia. Medicina personalizada: Comparativa de técnicas para el diagnóstico automático del alzheimer. Master's thesis, Universidad Internacional de La Rioja (UNIR), 2019.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, jun 2009. doi: 10.1109/cvpr.2009.5206848.

Yiming Ding, Jae Ho Sohn, Michael G. Kawczynski, Hari Trivedi, Roy Harnish, Nathaniel W. Jenkins, Dmytro Lituiev, Timothy P. Copeland, Mariam S. Aboian, Carina Mari Aparici, Spencer C. Behr, Robert R. Flavell, Shih-Ying Huang, Kelly A. Zalocusky, Lorenzo Nardo, Youngho Seo, Randall A. Hawkins, Miguel Hernandez Pampaloni, Dexter Hadley, and Benjamin L. Franc. A deep learning model to predict a diagnosis of alzheimer disease by using sup18/supf-FDG PET of the brain. *Radiology*, 290(2):456–464, feb 2019. doi: 10.1148/radiol.2018180958.

Baskar Duraisamy, Jayanthi Venkatraman Shanmugam, and Jayanthi Annamalai. Alzheimer disease detection from structural MR images using FCM based weighted probabilistic neural network. *Brain Imaging and Behavior*, 13(1):87–110, feb 2018. doi: 10.1007/s11682-018-9831-2.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *J. Mach. Learn. Res.*, 11:625–660, mar 2010. ISSN 1532-4435.

Soheil Esmaeilzadeh, Dimitrios Belivanis, Kilian Pohl, and Ehsan Adeli. End-to-end alzheimer's disease diagnosis and biomarker identification. volume 11046, pages 337–345, 09 2018. ISBN 978-3-030-00918-2. doi: 10.1007/978-3-030-00919-9_39.

Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4): 193–202, apr 1980. doi: 10.1007/bf00344251.

Adrian Galdran, Aitor Alvarez-Gila, Maria Ines Meyer, Cristina L. Saratxaga, Teresa Araújo, Estibaliz Garrote, Guilherme Aresta, Pedro Costa, A. M. Mendonça, and Aurélio Campilho. Data-driven color augmentation techniques for deep skin image analysis. 2017.

Serge Gauthier, Barry Reisberg, Michael Zaudig, Ronald C Petersen, Karen Ritchie, Karl Broich, Sylvie Belleville, Henry Brodaty, David Bennett, Howard Chertkow, Jeffrey L Cummings, Mony de Leon, Howard Feldman, Mary Ganguli, Harald Hampel, Philip Scheltens, Mary C Tierney, Peter Whitehouse, and Bengt Winblad. Mild cognitive impairment. *The Lancet*, 367(9518):1262–1270, apr 2006. doi: 10.1016/s0140-6736(06)68542-5.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. 2014.

Hayit Greenspan, Bram van Ginneken, and Ronald M. Summers. Guest editorial deep learning in medical imaging: Overview and future promise of an exciting new technique. *IEEE Transactions on Medical Imaging*, 35(5):1153–1159, may 2016. doi: 10.1109/tmi.2016.2553401.

Douglas Greve and Bruce Fischl. Accurate and robust brain image alignment using boundary-based registration. *NeuroImage*, 48:63–72, 07 2009. doi: 10.1016/j.neuroimage.2009.06.060.

Michael Grundman. Mild cognitive impairment can be distinguished from alzheimer disease and normal aging for clinical trials. *Archives of Neurology*, 61(1):59, jan 2004. doi: 10.1001/archneur.61.1.59.

Ashish Gupta, Murat Seçkin Ayhan, and A. Maida. Natural image bases to represent neuroimaging data. In *ICML*, 2013.

Aurélien Géron. *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly UK Ltd., 2019. ISBN 1492032646. URL https://www.ebook.de/de/product/33315532/aurelien_geron_hands_on_machine_learning_with_scikit_learn_keras_and_tensorflow.html.

Changhee Han, Kohei Murao, Tomoyuki Noguchi, Yusuke Kawata, Fumiya Uchiyama, Leonardo Rundo, Hideki Nakayama, and Shin'ichi Satoh. Learning more with less: gan-based medical image augmentation. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. ACM, nov 2019. doi: 10.1145/3357384.3357890.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. 2015.

Karl Herrup. Commentary on "recommendations from the national institute on aging-alzheimer's association workgroups on diagnostic guidelines for alzheimer's disease." addressing the challenge of alzheimer's disease in the 21st century. *Alzheimer's &amp Dementia*, 7(3):335–337, may 2011. doi: 10.1016/j.jalz.2011.04.002.

Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, jul 2006. doi: 10.1162/neco.2006.18.7.1527.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. 2012.

Ehsan Hosseini-Asl, Georgy Gimel'farb, and Ayman El-Baz. Alzheimer's disease diagnostics by a deeply supervised adaptable 3d convolutional network. 2016a.

Ehsan Hosseini-Asl, Robert Keynton, and Ayman El-Baz. Alzheimer's disease diagnostics by adaptation of 3d convolutional network. In *2016 IEEE International Conference on Image Processing (ICIP)*. IEEE, sep 2016b. doi: 10.1109/icip.2016.7532332.

Jie Hu, Li Shen, Samuel Albanie, Gang Sun, and Enhua Wu. Squeeze-and-excitation networks. 2017.

D. H. Hubel. Single unit activity in striate cortex of unrestrained cats. *The Journal of Physiology*, 147(2):226–238, sep 1959. doi: 10.1113/jphysiol.1959.sp006238.

D. H. Hubel and T. N. Wiesel. Receptive fields of single neurones in the cat's striate cortex. *The Journal of Physiology*, 148(3):574–591, oct 1959. doi: 10.1113/jphysiol.1959.sp006308.

D. H. Hubel and T. N. Wiesel. Receptive fields and functional architecture of monkey striate cortex. *The Journal of Physiology*, 195(1):215–243, mar 1968. doi: 10.1113/jphysiol.1968.sp008455.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. 2015.

Jyoti Islam and Yanqing Zhang. Brain MRI analysis for alzheimer's disease diagnosis using an ensemble system of deep convolutional neural networks. *Brain Informatics*, 5(2), may 2018. doi: 10.1186/s40708-018-0080-3.

Mark Jenkinson and Stephen Smith. A global optimisation method for robust affine registration of brain images. *Medical Image Analysis*, 5(2):143–156, 2001. ISSN 1361-8415. doi: https://doi.org/10.1016/S1361-8415(01)00036-6. URL https://www.sciencedirect.com/science/article/pii/S1361841501000366.

Mark Jenkinson, Peter Bannister, Michael Brady, and Stephen Smith. Improved optimization for the robust and accurate linear registration and motion correction of brain images. *NeuroImage*, 17(2):825—841, October 2002. ISSN 1053-8119. doi: 10.1016/s1053-8119(02)91132-8. URL https://doi.org/10.1006/nimg.2002.1132.

Justin Ker, Lipo Wang, Jai Rao, and Tchoyoson Lim. Deep learning applications in medical image analysis. *IEEE Access*, 6:9375–9389, 2018. doi: 10.1109/access.2017.2788044.

Nikhil Ketkar. *Deep Learning with Python*. Springer-Verlag GmbH, 2017. ISBN 9781484227664. URL https://www.ebook.de/de/product/28972222/nikhil_ketkar_deep_learning_with_python.html.

S. Kloppel, C. M. Stonnington, J. Barnes, F. Chen, C. Chu, C. D. Good, I. Mader, L. A. Mitchell, A. C. Patel, C. C. Roberts, N. C. Fox, C. R. Jack, J. Ashburner, and R. S. J. Frackowiak. Accuracy of dementia diagnosis–a direct comparison between radiologists and a computerized method. *Brain*, 131(11):2969–2974, jun 2008. doi: 10.1093/brain/awn239.

Gregory Koch, Richard Zemel, Ruslan Salakhutdinov, et al. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2, page 0. Lille, 2015.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural*

Information Processing Systems - Volume 1, NIPS'12, page 1097–1105, Red Hook, NY, USA, 2012. Curran Associates Inc.

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

Joon Lee and Roger G Mark. An investigation of patterns in hemodynamic data indicative of impending hypotension in intensive care. *BioMedical Engineering OnLine*, 9(1):62, 2010. doi: 10.1186/1475-925x-9-62.

Fan Li and Manhua Liu. Alzheimer's disease diagnosis based on multiple cluster dense convolutional networks. *Computerized Medical Imaging and Graphics*, 70:101–110, dec 2018. doi: 10.1016/j.compmedimag.2018.09.009.

Feng Li, Loc Tran, Kim-Han Thung, Shuiwang Ji, Dinggang Shen, and Jiang Li. A robust deep model for improved classification of AD/MCI patients. *IEEE Journal of Biomedical and Health Informatics*, 19(5):1610–1616, sep 2015. doi: 10.1109/jbhi.2015.2429556.

Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. 2013.

Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen A. W. M. van der Laak, Bram van Ginneken, and Clara I. Sánchez. A survey on deep learning in medical image analysis. 2017a. doi: 10.1016/j.media.2017.07.005.

Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen A.W.M. van der Laak, Bram van Ginneken, and Clara I. Sánchez. A survey on deep learning in medical image analysis. *Medical Image Analysis*, 42:60–88, dec 2017b. doi: 10.1016/j.media.2017.07.005.

Feng Liu, Heung-Il Suk, Chong-Yaw Wee, Huafu Chen, and Dinggang Shen. High-order graph matching based feature selection for alzheimer's disease identification. In *Advanced Information Systems Engineering*, pages 311–318. Springer Berlin Heidelberg, 2013. doi: 10.1007/978-3-642-40763-5_39.

Feng Liu, Chong-Yaw Wee, Huafu Chen, and Dinggang Shen. Inter-modality relationship constrained multi-modality multi-task feature selection for alzheimer's disease and mild cognitive impairment identification. *NeuroImage*, 84:466–475, jan 2014a. doi: 10.1016/j.neuroimage.2013.09.015.

Siqi Liu, Sidong Liu, Weidong Cai, Sonia Pujol, Ron Kikinis, and Dagan Feng. Early diagnosis of alzheimer's disease with deep learning. In *2014 IEEE 11th International Symposium on Biomedical Imaging (ISBI)*. IEEE, apr 2014b. doi: 10.1109/isbi.2014.6868045.

Yan Liu, Strahinja Stojadinovic, Brian Hrycushko, Zabi Wardak, Steven Lau, Weiguo Lu, Yulong Yan, Steve B. Jiang, Xin Zhen, Robert Timmerman, Lucien Nedzi, and Xuejun Gu. A deep convolutional neural network-based automatic delineation strategy for multiple brain metastases stereotactic radiosurgery. *PLOS ONE*, 12(10):e0185844, oct 2017. doi: 10.1371/journal.pone.0185844.

Donghuan Lu, , Karteek Popuri, Gavin Weiguang Ding, Rakesh Balachandar, and Mirza Faisal Beg. Multimodal and multiscale deep neural networks for the early diagnosis of alzheimer's disease using structural MR and FDG-PET images. *Scientific Reports*, 8(1), apr 2018. doi: 10.1038/s41598-018-22871-z.

H. Matsuda. Role of neuroimaging in alzheimer's disease, with emphasis on brain perfusion SPECT. *Journal of Nuclear Medicine*, 48(8):1289–1300, aug 2007. doi: 10.2967/jnumed.106.037218.

Maciej A. Mazurowski, Piotr A. Habas, Jacek M. Zurada, Joseph Y. Lo, Jay A. Baker, and Georgia D. Tourassi. Training neural network classifiers for medical decision making: The effects of imbalanced datasets on classification performance. *Neural Networks*, 21(2-3):427–436, mar 2008. doi: 10.1016/j.neunet.2007.12.031.

Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133, dec 1943. doi: 10.1007/bf02478259.

G. McKhann, D. Drachman, M. Folstein, R. Katzman, D. Price, and E. M. Stadlan. Clinical diagnosis of alzheimer's disease: Report of the NINCDS-ADRDA work group under the auspices of department of health and human services task force on alzheimer's disease. *Neurology*, 34(7):939–939, jul 1984. doi: 10.1212/wnl.34.7.939.

Guy M. McKhann, David S. Knopman, Howard Chertkow, Bradley T. Hyman, Clifford R. Jack, Claudia H. Kawas, William E. Klunk, Walter J. Koroshetz, Jennifer J. Manly, Richard Mayeux, Richard C. Mohs, John C. Morris, Martin N. Rossor, Philip Scheltens, Maria C. Carrillo, Bill Thies, Sandra Weintraub, and Creighton H. Phelps. The diagnosis of dementia due to alzheimer's disease: Recommendations from the national institute on aging-alzheimer's association workgroups on diagnostic guidelines for alzheimer's disease. *Alzheimer's &amp Dementia*, 7(3):263–269, apr 2011. doi: 10.1016/j.jalz.2011.03.005.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. 2013.

Jakub Nalepa, Michal Marcinkiewicz, and Michal Kawulok. Data augmentation for brain-tumor segmentation: A review. *Frontiers in Computational Neuroscience*, 13, dec 2019. doi: 10.3389/fncom.2019.00083.

James P. B. O'Connor, Eric O. Aboagye, Judith E. Adams, Hugo J. W. L. Aerts, Sally F. Barrington, Ambros J. Beer, Ronald Boellaard, Sarah E. Bohndiek, Michael Brady, Gina Brown, David L. Buckley, Thomas L. Chenevert, Laurence P. Clarke, Sandra Collette, Gary J. Cook, Nandita M. Desouza, John C. Dickson, Caroline Dive, Jeffrey L. Evelhoch, Corinne Faivre-Finn, Ferdia A. Gallagher, Fiona J. Gilbert, Robert J. Gillies, Vicky Goh, J. R. Griffiths, Ashley M. Groves, Steve Halligan, Adrian L. Harris, David J. Hawkes, Otto S. Hoekstra, Erich P. Huang, Brian F. Hutton, Edward F. Jackson, Gordon C. Jayson, Andrew Jones, Dow-Mu Koh, Denis Lacombe, Philippe Lambin, Nathalie Lassau, Martin O. Leach, Ting-Yim Lee, Edward L. Leen, Jason S. Lewis, Yan Liu, Mark F. Lythgoe, Prakash Manoharan, Ross J. Maxwell, Kenneth A. Miles, Bruno Morgan, Steve Morris, Tony Ng, Anwar R. Padhani, Geoff J. M. Parker, Mike Partridge, Arvind P. Pathak, Andrew C. Peet, Shonit Punwani, Andrew R. Reynolds, Simon P. Robinson, Lalitha K. Shankar, Ricky A. Sharma, Dmitry Soloviev, Sigrid G. Stroobants, Daniel C. Sullivan, Stuart A. Taylor, Paul S. Tofts, Gillian M. Tozer, Marcel B. van Herk, Simon Walker-Samuel, James Wason, Kaye J. Williams, Paul Workman, Thomas E. Yankeelov, Kevin M. Brindle, Lisa M. McShane, Alan Jackson, and John C. Waterton. Imaging biomarker roadmap for cancer studies. *Nature Reviews Clinical Oncology*, 14(3):169–186, March 2017. ISSN 1759-4774. doi: 10.1038/nrclinonc.2016.162.

Kanghan Oh, Young-Chul Chung, Ko Woon Kim, Woo-Sung Kim, and Il-Seok Oh. Classification and visualization of alzheimer's disease using volumetric convolutional neural network and transfer learning. *Scientific Reports*, 9(1), dec 2019. doi: 10.1038/s41598-019-54548-6.

Mark Palatucci, Dean Pomerleau, Geoffrey E Hinton, and Tom M Mitchell. Zero-shot learning with semantic output codes. In Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 22. Curran Associates, Inc., 2009. URL https://proceedings.neurips.cc/paper/2009/file/1543843a4723ed2ab08e18053ae6dc5b-Paper.pdf.

Santanu Pattanayak. *Pro Deep Learning with TensorFlow*. Springer-Verlag GmbH, 2017. ISBN 9781484230961. URL https://www.ebook.de/de/product/31061192/santanu_pattanayak_pro_deep_learning_with_tensorflow.html.

Nick Pawlowski, Sofia Ira Ktena, Matthew C. H. Lee, Bernhard Kainz, Daniel Rueckert, Ben Glocker, and Martin Rajchl. Dltk: State of the art reference implementations for deep learning on medical images. 2017.

Adrien Payan and Giovanni Montana. Predicting alzheimer's disease: a neuroimaging study with 3d convolutional neural networks. 2015.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python. *J. Mach. Learn. Res.*, 12 (null):2825–2830, nov 2011. ISSN 1532-4435.

Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2014. doi: 10.3115/v1/d14-1162.

Sergio Pereira, Adriano Pinto, Victor Alves, and Carlos A. Silva. Brain tumor segmentation using convolutional neural networks in MRI images. *IEEE Transactions on Medical Imaging*, 35(5):1240–1251, may 2016. doi: 10.1109/tmi.2016.2538465.

Shangran Qiu, Prajakta S Joshi, Matthew I Miller, Chonghua Xue, Xiao Zhou, Cody Karjadi, Gary H Chang, Anant S Joshi, Brigid Dwyer, Shuhan Zhu, Michelle Kaku, Yan Zhou, Yazan J Alderazi, Arun Swaminathan, Sachin Kedar, Marie-Helene Saint-Hilaire, Sanford H Auerbach, Jing Yuan, E Alton Sartor, Rhoda Au, and Vijaya B Kolachalama. Development and validation of an interpretable deep learning framework for alzheimer's disease classification. *Brain*, 143(6):1920–1933, may 2020. doi: 10.1093/brain/awaa137.

Muhammad Imran Razzak, Saeeda Naz, and Ahmad Zaib. Deep learning for medical image processing: Overview, challenges and the future. In *Lecture Notes in Computational Vision and Biomechanics*, pages 323–350. Springer International Publishing, nov 2017. doi: 10.1007/978-3-319-65981-7_12.

E. D. Roberson and L. Mucke. 100 years and counting: Prospects for defeating alzheimer's disease. *Science*, 314(5800):781–784, nov 2006. doi: 10.1126/science.1132813.

Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, USA, 3rd edition, 2009. ISBN 0136042597.

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. One-shot learning with memory-augmented neural networks. 2016.

Saman Sarraf, Danielle D. DeSouza, John Anderson, and Ghassem Tofighi and. DeepAD: Alzheimer's disease classification via deep convolutional neural networks using MRI and fMRI. aug 2016. doi: 10.1101/070441.

Ling Shao, Fan Zhu, and Xuelong Li. Transfer learning for visual categorization: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 26(5):1019–1034, may 2015. doi: 10.1109/tnnls.2014.2330900.

Hoo-Chang Shin, Neil A. Tenenholtz, Jameson K. Rogers, Christopher G. Schwarz, Matthew L. Senjem, Jeffrey L. Gunter, Katherine P. Andriole, and Mark Michalski. Medical image synthesis for data augmentation and anonymization using generative adversarial networks. In *Simulation and Synthesis in Medical Imaging*, pages 1–11. Springer International Publishing, 2018. doi: 10.1007/978-3-030-00536-8_1.

Connor Shorten and Taghi M. Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1), jul 2019. doi: 10.1186/s40537-019-0197-0.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. 2014.

Stephen M. Smith. Fast robust automated brain extraction. *Human Brain Mapping*, 17(3): 143–155, 2002. doi: https://doi.org/10.1002/hbm.10062. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/hbm.10062.

Roman Solovyev, Alexandr A. Kalinin, and Tatiana Gabruseva. 3d convolutional neural networks for stalled brain capillary detection. *Computers in Biology and Medicine*, 141: 105089, feb 2022. doi: 10.1016/j.compbiomed.2021.105089.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. URL http://jmlr.org/papers/v15/srivastava14a.html.

Kyle Strimbu and Jorge Tavel. What are biomarkers? *Current opinion in HIV and AIDS*, 5: 463–6, 11 2010. doi: 10.1097/COH.0b013e32833ed177.

Heung-Il Suk and Dinggang Shen. Deep learning-based feature representation for AD/MCI classification. In *Advanced Information Systems Engineering*, pages 583–590. Springer Berlin Heidelberg, 2013. doi: 10.1007/978-3-642-40763-5_72.

Heung-Il Suk, , Seong-Whan Lee, and Dinggang Shen. Latent feature representation with stacked auto-encoder for AD/MCI diagnosis. *Brain Structure and Function*, 220(2):841–859, dec 2013. doi: 10.1007/s00429-013-0687-3.

Heung-Il Suk, Seong-Whan Lee, and Dinggang Shen. Hierarchical feature representation and multimodal fusion with deep learning for AD/MCI diagnosis. *NeuroImage*, 101:569–582, nov 2014. doi: 10.1016/j.neuroimage.2014.06.077.

Shobhita Sundaram and Neha Hulkund. Gan-based data augmentation for chest x-ray classification. 2021.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. 2014.

Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. 2016.

Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1701–1708, 2014.

A. M. Turing. *Computing Machinery and Intelligence*, page 11–35. MIT Press, Cambridge, MA, USA, 1995. ISBN 0262560925.

Karl Weiss, Taghi M. Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big Data*, 3(1), may 2016. doi: 10.1186/s40537-016-0043-6.

Yongqin Xian, Christoph H. Lampert, Bernt Schiele, and Zeynep Akata. Zero-shot learning – a comprehensive evaluation of the good, the bad and the ugly. 2017.

Daoqiang Zhang, Yaping Wang, Luping Zhou, Hong Yuan, and Dinggang Shen. Multimodal classification of alzheimer's disease and mild cognitive impairment. *NeuroImage*, 55(3): 856–867, apr 2011. doi: 10.1016/j.neuroimage.2011.01.008.