Universidade da Beira Interior
Departamento de Informática

Nº 313 - 2021/2022: *Modelo baseado em deep learning para a classificação de doenças pulmonares em imagens raio-X do tórax*

Elaborado por:

Alexandre Salcedas Monteiro

Orientador:

Professor Doutor João Carlos Raposo Neves

2 de julho de 2022

Agradecimentos

A conclusão deste trabalho não teria sido possível sem a ajuda do Professor Doutor João Neves que me orientou na realização deste projeto, bem como ao Cristiano Patrício, aluno de doutoramento, por toda a disponibilidade e compreensão que me demonstraram.

Aos meus pais, por todos os sacrifícios e esforços que me permitiram alcançar os meus objetivos académicos e, futuramente, profissionais. Sem eles nada teria sido possível.

À demais família por todo o apoio, paciência e conselhos que me mostraram o caminho certo.

À minha companheira, namorada e melhor amiga por me ter ajudado a ser quem sou hoje, por aturar-me em todas as situações, por me guiar e motivar e por estar sempre ao meu lado, independentemente da situação, por toda a força e apoio nos momentos difíceis.

Um obrigado aos meus amigos de curso pelas noites mal dormidas, pela diversão e, sobretudo, por me ensinarem que um curso não se faz só.

Aos meus amigos de longa data por todas as gargalhadas, brincadeiras e momentos bem passados.

A todas as outras pessoas que não mencionei, mas que direta ou indiretamente me auxiliaram neste trajeto que culmina na elaboração deste projeto.

A todos o meu muito obrigada!

Conteúdo

Lista de Figuras

Lista de Tabelas

Acrónimos

ROC curve	*Receiver Operating Characteristic Curve*
ADAM	*Adaptive Moment Estimation*
API	*Application Programming Interface*
AUC	*Area Under the ROC Curve*
CAM	*Class Activation Mapping*
CSV	*Comma-separated values*
FPR	*False Positive Rate*
IDE	*Integrated Development Environment*
RNC	Rede Neuronal Convolucional
TPR	*True Positive Rate*
DL	*Deep Learning*
IA	Inteligência Artificial
ML	*Machine Learning*
RN	Rede Neuronal

Capítulo

1

Introdução

1.1 Enquadramento e Motivação

A medicina tradicional, nomeadamente no diagnóstico de patologias pulmonares e relacionadas, requer a utilização de meios complementares de diagnóstico, como, por exemplo, o RAIO-X. Devido à elevada complexidade destas patologias, estas são difíceis de distinguir, pelo que existe a necessidade de encontrar novos métodos no auxílio dos especialistas formados atualmente.

Os especialistas são dotados de erro e, para além da sua escassez, diferentes especialistas poderão ter diferentes opiniões levando a uma falha no diagnóstico, impactando a qualidade de vida do paciente.

Assim, a implementação de novas tecnologias, na área de imagem médica, irá permitir os radiologistas beneficiar destes recursos de modo a exercerem melhor o sua profissão.

Neste projeto desenvolveram-se vários modelos que poderão solucionar o problema descrito, com recurso à área de Inteligência Artificial (IA).

1.2 Objetivos

O desenvolvimento deste projeto, permitir-me-á:

- Compreender o conceito de *Deep Learning* (DL) e a sua aplicabilidade;
- Entender as principais patologias identificáveis numa radiografia do tórax, através de uma IA.

- Perceber quais os métodos existentes e, ainda, tentar melhorar os mesmos na medida em que é necessário interpretar e entender corretamente as imagens apresentadas;
- Reconhecer as principais vantagens de uma IA aplicada à área da medicina.

1.3 Organização do Documento

De modo a refletir o trabalho que foi feito, este documento encontra-se estruturado da seguinte forma:

1. O primeiro capítulo – **Introdução** – iniciado na página 1, refere-se ao enquadramento e motivação para o projeto, os seus objetivos e a respetiva organização do documento.
2. O segundo capítulo – **Estado da Arte** – iniciado na página 3, descreve os conceitos mais importantes no âmbito deste projeto, bem como os métodos estado da arte.
3. O terceiro capítulo – **Métodos** – iniciado na página 19, retrata as diferentes tecnologias e ferramentas utilizadas no desenvolvimento do projeto, bem como todos os dados e procedimentos aplicados.
4. O quarto capítulo – **Teste e resultados obtidos** – iniciado na página 27, detalha os diversos resultados obtidos após a implementação e sua comparação e interpretação em termos visuais.
5. O quinto capítulo – **Conclusões** – iniciado na página 45, expõe as principais conclusões obtidas na realização do projeto.

Capítulo

2

Estado da Arte

2.1 Introdução

Assim que foi possível analisar e fornecer imagens médicas, para um computador, construiram-se sistemas para análise automática.

Inicialmente, entre 1970 e 1990, cada uma das imagens eram examinadas, de forma sequencial, píxel a píxel, recorrendo-se a filtros de deteção de arestas, também conhecidos como *handcrafted features*. Este método foi usado para construir sistemas baseados em regras compostas de forma a resolver problemas e tarefas particulares. Durante algum tempo, as *handcrafted features* eram essenciais no desenho destes sistemas, no entanto, tinham que ser feitos por humanos.

O DL tem-se tornado numa área alvo de grande investigação e está a ser usada, regularmente, para classificação e previsão de tarefas, como o reconhecimento de imagem/voz ou de texto.

A sua popularidade tem vindo a aumentar desde 2012, onde a arquitetura do DL ultrapassou significativamente as técnicas habituais que dependiam das *handcrafter features* (como uma Rede Neuronal (RN)) para a classificação de imagem.

O sucesso deveu-se à disponibilidade de dados em maior quantidade, ao poder computacional eficiente e acessível e aos avanços técnicos, tais como:

- Pré-treino: Veio resolver o treino ineficiente de uma RN com muitas camadas.

- Convoluções: Implementação de menos parâmetros numa Rede Neuronal Convolucional (RNC), tornando a rede mais eficiente.

A base técnica do DL é, na sua maioria, à volta de uma RN(secção 2.2) e das suas extenções, como a RNC(secção 2.3).

2.2 Redes Neuronais

A RN tem como objetivo inferir a relação entre um conjunto de dados, por um processo que replica o funcionamento do cérebro humano. Desta maneira, uma RN pode ser referida como um mecanismo biológico ou artificial como ilustra a figura 2.1.

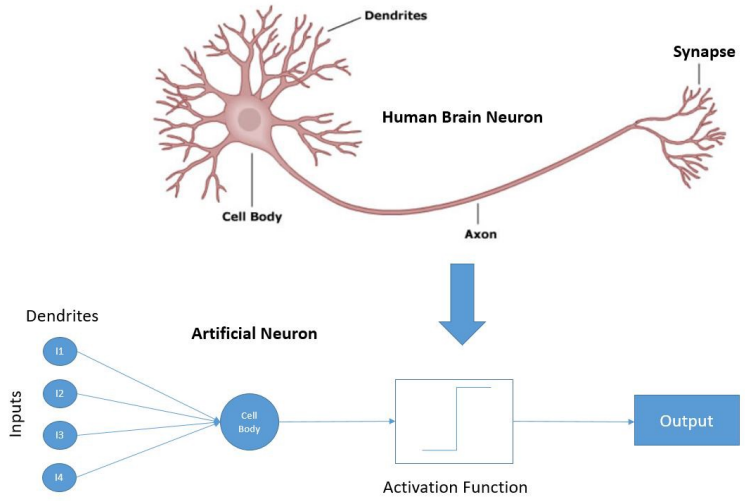


Figura 2.1: Diagrama esquemático de um neurónio biológico e artificial [1]. Na parte superior da imagem, encontra-se a representação de um neurónio do cérebro humano, enquanto na parte inferior mostra-se o neurónio artificial.

A RN não é mais do que um grupo de neurónios artificiais, estruturados de maneira a conseguirem gerar o melhor *output* possível, sem ser preciso alterar a sua constituição, independentemente do *input*. Cada um desses neurónios contém, um conjunto de pesos (W) e um *bias* (b). Esses dois valores com os dados de *input* formam uma reta linear, como mostra na figura 2.2.

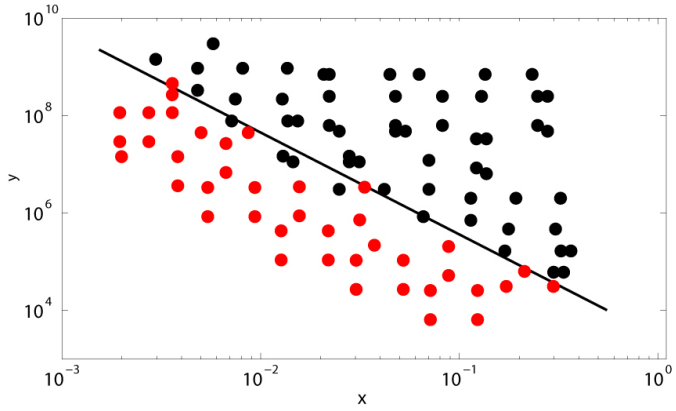


Figura 2.2: Diagrama esquemático de uma reta linear criado por um neurónio artificial (separação de dados).

Em exemplos mais complexos, torna-se difícil conseguir uma reta linear para agrupar os dados, com maior precisão. A figura 2.3 ilustra a inferência de dois neurónios sobre um conjunto de dados. Como cada neurónio cria a sua própria reta linear, então, dois ou mais neurónios criam várias retas e, desta forma, evitam que certos dados sejam mal interpretados, dado que existe uma maior separabilidade, como na figura 2.2.

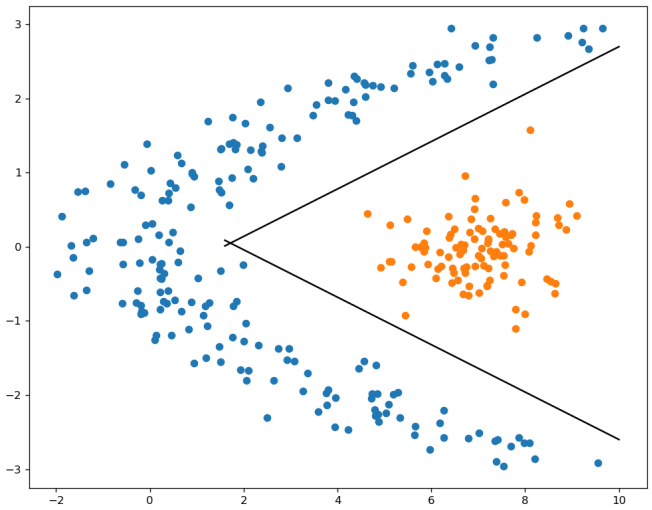


Figura 2.3: Diagrama esquemático de retas lineares obtidas por dois neurónios, que, no seu conjunto, separam adequadamente os dados [2].

Na imagem 2.4, encontra-se demonstrado como pode ser estruturada uma RN. Essa mesma é formada por três categorias de camadas:

- *Input*
- *Output*
- *Hidden*

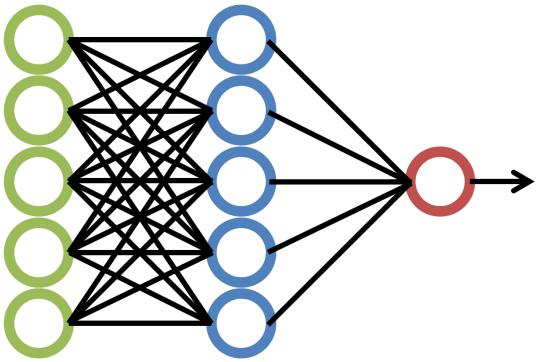


Figura 2.4: Diagrama esquemático de uma RN, onde os neurónios assinalados a verde pertencem à camada de *input*, os azuis à camada *hidden* e, por fim, o vermelho à camada de *output* [3].

Cada *output* de uma camada é o *input* da próxima, tornando a RN mais complexa ao adicionar mais camadas *hidden*. Esse *output* de uma camada é calculado através do uso de uma função de ativação (secção 2.2.1), preferivelmente não linear entre as camadas *hidden* e linear na camada de *output*.

Chegando à camada de output, aquando do treino, deve-se medir o erro entre o valor previsto e o valor real, como mencionado na secção 2.2.2.

Estes dois passos estão definidos como *forwardpropagation*. O mesmo consiste em avançar na RN, desde a camada de *input* até à camada de *output*.

O erro obtido, ou perda, é usado, com a descida do gradiente, para procurar os melhores pesos e *bias*, para se obter o menor erro possível, como mencionado na secção 2.2.3.

Para se conseguir atualizar os pesos e *bias* usa-se o *backpropagation*, onde o mesmo é referenciado na secção 2.2.4. Estes dois métodos são o fundamento para que a RN consiga aprender.

2.2.1 Função de Ativação

Uma das etapas principais é a utilização de uma função de ativação. Cada neurónio usa uma função de ativação, para decidir se o próprio deve ou não

ativar, conforme a seguinte fórmula:

$$Y = F(b + \sum_{i=0}^{n} X_i W_i).$$

A passagem da multiplicação do *input* (X) com os pesos (W) atribuídos, com a soma de um *bias* (b), pela função de ativação (F) resulta num *output* (Y), que será usado como *input* na próxima camada.

Como se pode observar na figura 2.5, existem várias funções de ativação, cada uma das quais com diferentes funcionalidades.

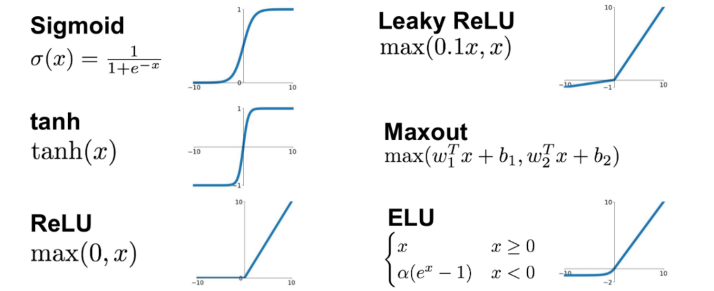


Figura 2.5: Diagrama com as funções de ativação mais usadas [4].

O uso de cada função depende do problema em questão, no entanto, as mais usadas são as funções diferenciáveis, para que, posteriormente, seja possível aplicar o *backpropagation*.

2.2.2 Função de Perda

Após obtenção do *output*, este é introduzido numa função de perda. Desta forma, quando aplicada a um conjunto de dados, a função indica o quão preciso é esse *output* consoante o resultado verdadeiro.

À semelhança das funções de ativação, existem várias funções de perda tipificadas em, principalmente, duas classes. Destas, a mais usual assume a seguinte equação:

$$L = -\frac{1}{m} \sum_{i=1}^{m} (Y_i \log(A_i) + (1 - Y_i) \log(1 - A_i)).$$

Nesta função, aplica-se a entropia entre o resultado previsto (A_i) pela RN e o verdadeiro resultado (Y_i).

2.2.3 Descida do Gradiente

A descida do gradiente é de extrema importância para a RN, pelo que a sua utilização pretende minimizar o erro. Para o efeito, pretende-se escolher os valores dos pesos (W) e do *bias* (b) mais apropriados na direção oposta do gradiente da função (figura 2.6).

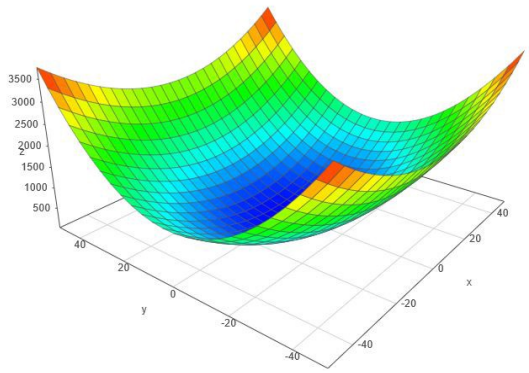


Figura 2.6: Diagrama com um gradiente convexo e apenas um mínimo global [5].

Idealmente, é preferível obter uma função de custo convexa, que possua somente um mínimo e que este seja o global. Caso não se trate de um gradiente convexo, então, o objetivo principal passará por identificar e localizar o mínimo global, evitando os mínimos locais.

A taxa de aprendizagem define o tamanho dos passos efetuados para atingir o mínimo global. Na eventualidade da existência de mínimos locais, devem-se evitar os mesmos e encontrar um mínimo global, como mostra na figura 2.7.

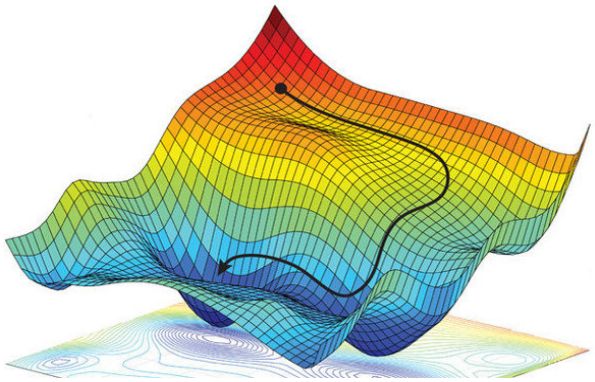


Figura 2.7: Diagrama com um gradiente não convexo, onde o mesmo tem vários mínimos locais e apenas um global [6].

2.2.4 Backpropagation

O *backpropagation* é essencial para treinar uma RN. Esta prática é usada para atualizar os pesos e o *bias*, com base na perda obtida. O ajuste adequado dos pesos assegura taxas de erro mais baixas, tornando a RN fiável ao aumentar a sua generalização.

Este processo consiste em efetuar as derivadas parciais (figura 2.8), em ordem aos parâmetros a atualizar, isto é, os pesos e *bias*. Deste modo, é possível chegar ao início da RN e o recomeçar o processo desde a camada de *input*, agora com pesos e *bias* atualizados, conduzindo a um menor erro .

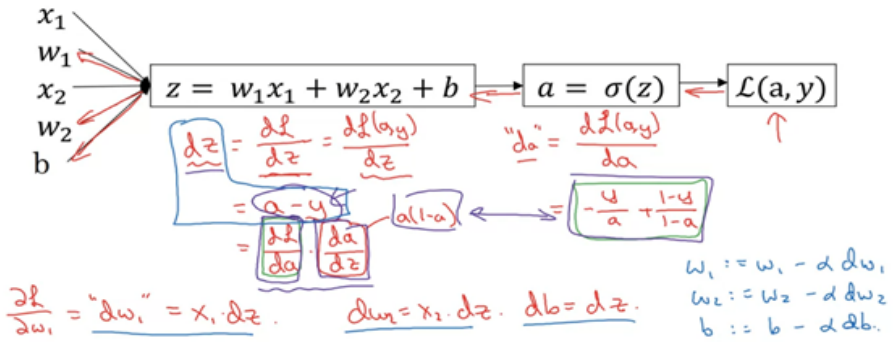


Figura 2.8: Diagrama esquemático sobre as derivadas parciais usadas no *backpropagation* [7].

2.3 Redes Neuronais Convolucionais

Ao contrário das *handcrafted image features*, as *learned features* são inferidas de um conjunto de dados de imagem, através de um procedimento de treino, de modo a cumprir uma determinada tarefa.

Uma RNC é um exemplo de uma *deep neuronal network* usada para extrair *learned features*. Esta categoria de rede possui características que, comparativamente à habitual RN, lhe permite classificar melhor imagens através do uso de:

- Filtros (secção 2.3.1) — Diminuição na quantidade de parâmetros;
- *Padding* (secção 2.3.2) — Menos perda de informação;
- *Stride* (secção 2.3.3) — Redução da imagem.

Numa RNC, estas características são implementadas em duas categorias de camadas:

- Convolucional;
- *Pooling*.

Nessas duas camadas, aplica-se o método de convolução (secção 2.3.4). Esse método é a forma de modificar o *input* (neste caso imagens), através da aplicação de filtros.

Assim, a aplicação nas imagens permite que estas sejam reduzidas para que as várias categorias de filtros usados possam aprender diferentes porções de imagens.

2.3.1 Filtros

Numa RNC, os filtros detetam padrões, através de bordas e/ou mudanças na intensidade de valores do *input*. Um filtro é usado em todo o *input*, de tal forma que não são necessários tantos parâmetros quanto uma RN, permitindo reduzir significativamente o custo e tempo computacional.

Na figura 2.9, pode observar-se o uso de um filtro, com pesos aprendidos, num dado *input* para conseguir o melhor *output* possível.

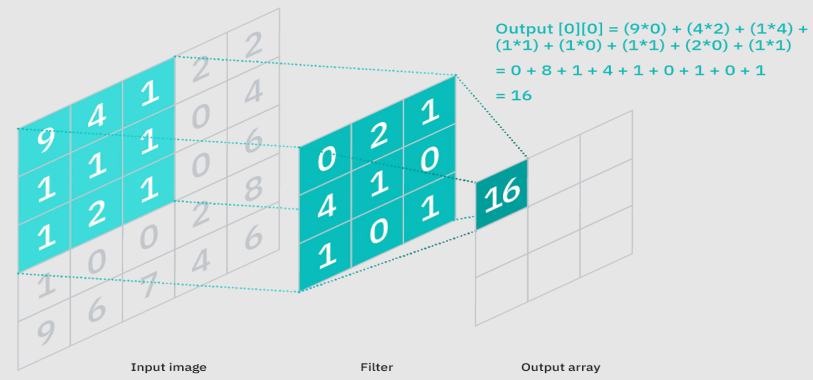


Figura 2.9: Diagrama da aplicação de um filtro [8].

2.3.2 *Padding*

Por definição, o *padding* adiciona, em torno do *input*, colunas e linhas extras de zeros.

Este é usado, juntamente com os filtros, para reduzir o *input*. Por vezes, a aplicação da convolução sem *padding* resulta na perda de informação e/ou reduz em demasia o *input*. Para evitar esta situação, e com base na figura 2.10, pode observar-se que, após a implementação do *padding*, o *output* mantém o tamanho sem perder tanta informação.

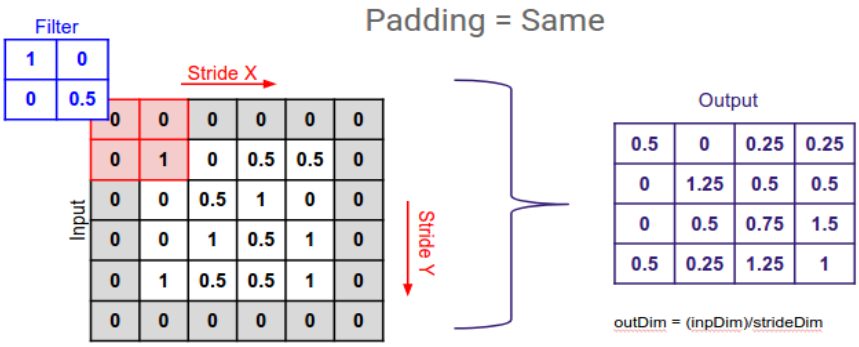
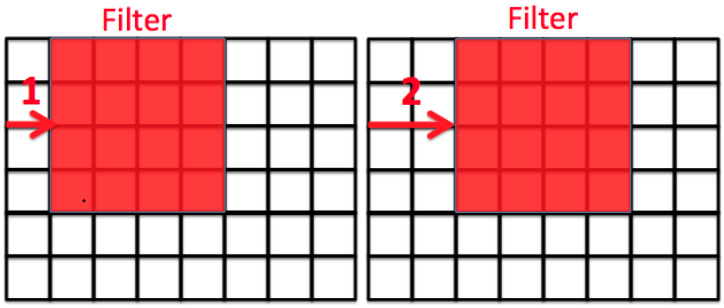


Figura 2.10: Diagrama de uma aplicação de *padding* [9].

2.3.3 *Stride*

O *stride* é o número de deslocamentos de píxeis num dado *input*. Quando o mesmo tem o valor 1 (figura 2.11a), então o filtro a aplicar mover-se-á um píxel.



(a) Diagrama de uma aplicação de *stride* de um píxel.

(b) Diagrama de uma aplicação de *stride* de dois píxeis.

Figura 2.11: Duas figuras com diferentes categorias de *stride* [10].

Caso seja dois (figura 2.11b), o filtro a aplicar move-se dois píxeis. Esta implementação pode reduzir, significativamente, o *input*.

2.3.4 Convolução

Uma convolução é uma aplicação simples de um filtro num dado *input* que resulta numa ativação. Um *feature map* é o resultado de várias aplicações do mesmo filtro num dado input (figura 2.12). Este *feature map* será usado como *input* na próxima camada.

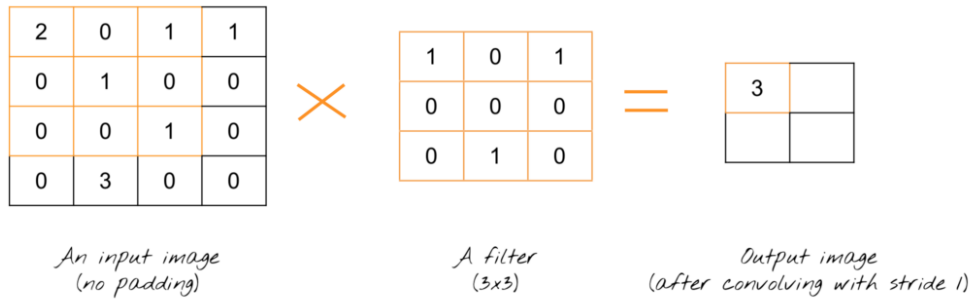


Figura 2.12: Diagrama de uma convolução, com apenas um filtro [11].

2.3.5 *Pooling*

A camada de *pooling* é usada para reduzir a imagem, tanto em altura como em largura.

A mesma tem dois tipos, que aquando da aplicação do filtro se obtém:

- *Max-pooling*: O maior valor como *output* (figura 2.13);
- *Average-pooling*: A média dos valores como *output* (figura 2.14).

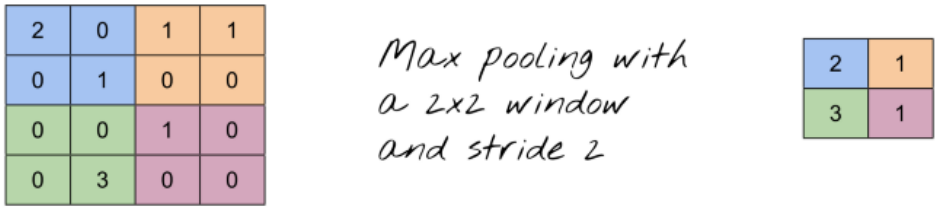


Figura 2.13: Diagrama de uma camada *max-pooling* [12].

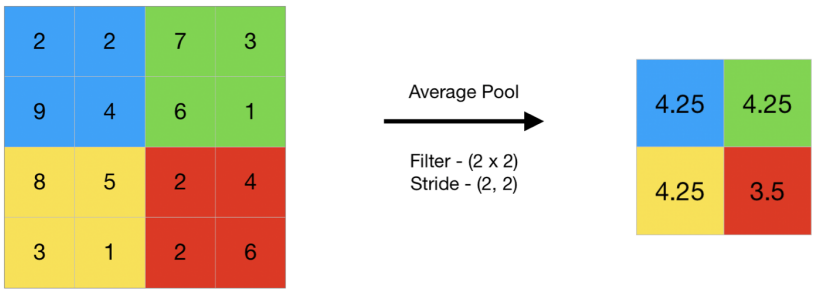


Figura 2.14: Diagrama de uma camada *average-pooling* [13].

Segundo a figura 2.15, após a aplicação de *pooling* (neste caso *max-pooling*) com um filtro (2x2) e *stride* (2x2), pode verificar-se que o *output* é o valor mais alto. Assim, esta camada dá mais importância aos valores que se sobrepõem.

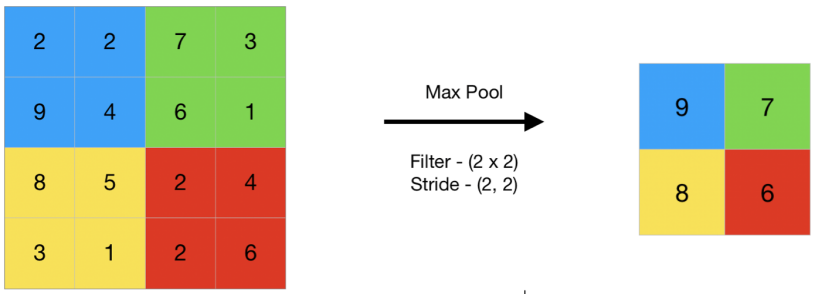


Figura 2.15: Diagrama de uma camada *max-pooling* [14].

2.4 *ImageNet*

O *ImageNet* é um *dataset* público, organizado para ajudar na investigação na área de visão computacional [21]. O *ImageNet* providencia os valores iniciais para os pesos de um modelo.

2.5 Imagem médica

Diariamente, a medicina convencional enfrenta novos desafios, sejam estes o surgimento novas doenças, redução de custos ou necessidade de rápida atuação. A DL consegue desempenhar um papel importante relativamente à tomada de decisões céleres, tentando ultrapassar a precisão dos especialistas da área. Os mesmos necessitam de muitas horas para analisar certas patologias, enquanto um modelo 2.5.1 necessita de menos tempo para o efetuar.

Alguns problemas a considerar ao aplicar modelos em medicina são os seguintes:

- *Class Imbalance* — Não haver um número igual de exemplos de pacientes saudáveis e não saudáveis;
- *Multitask* — Em vez de um determinado paciente pertencer apenas ao binómio "ser saudável/padecer de patologia", este pode sofrer de várias patologia em simultâneo;
- *Patient Overlap* — Por vezes, com conjuntos de dados muito grandes, é possível ter o mesmo paciente em todos os conjuntos (treino, valida-

ção e teste), sendo provável o modelo memorizar, tornando-se inútil aquando da generalização.

2.5.1 Modelos

Com o rápido desenvolvimento de vários modelos para análise de imagem médica, estes revelaram resultados equiparáveis aos obtidos por especialistas na área.

Como mencionado em [22, 23], alguns dos métodos mais usados:

Modelo	Ano	Característica principal
DenseNet	2016	Interligação densa de todas as camadas
Inception-v3	2016	Convoluções fatorizadas, mais pequenas e assimétricas
Inception-ResNet-v2	2016	Classificação de imagens em 1000 categorias de objetos

Tabela 2.1: Tabela com os três modelos usados durante a implementação.

2.5.2 *DenseNet*

A *DenseNet* é uma arquitetura moderna de uma RNC para reconhecimento visual de objetos, que adquiriu um estado da arte com menos parâmetros. A mesma, com os seus atributos concatenados, ou seja, o *output* de cada camada torna-se o *input* das camadas seguintes (como demonstra a figura 2.16), pretende resolver o problema de reconhecimento visual de objetos ao interligar densamente todas as camadas.

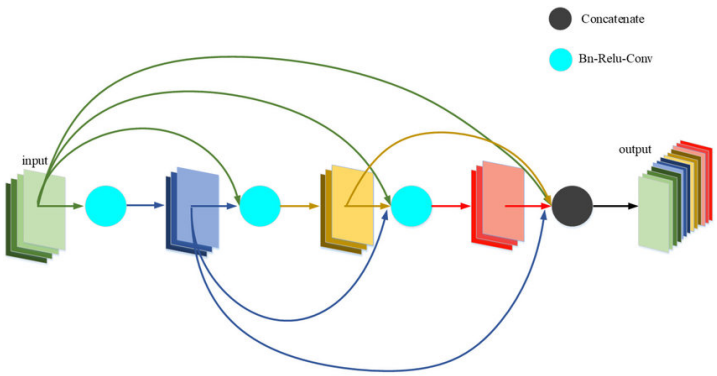


Figura 2.16: Diagrama de convoluções densamente conectadas da arquitetura *DenseNet*, retirada de [15].

Este método foi desenvolvido para resolver o problema do *vanishing gradient*, no qual a informação vai-se perdendo sucessivamente consoante o tamanho da rede em questão, em RNC mais profundas.

A *DenseNet* possui várias arquiteturas, desde a *DenseNet121* até à *DenseNet256*. A *DenseNet121* é composta por 121 camadas (5 — (6 – 12 – 24 – 16) x 2):

- 5 camadas de convolução e *pooling*;
- 3 camadas de transição (6 – 12 – 24);
- 1 camada de classificação (16);
- 2 camadas densamente conectadas;

Geralmente, uma RNC calcula o *input* de uma camada (n) usando uma transformação não linear no *input* da camada anterior ($n-1$), regida pela seguinte expressão:

$$X_n = H_n(X_n - 1).$$

No entanto, uma *DenseNet* recebe as características das camadas anteriores, de modo a melhorar o fluxo de informação. Este processo é demonstrado pela seguinte fórmula:

$$X_n = H_n(X_0, X_1, ..., X_n - 1).$$

2.5.3 *Inception-v3*

A *Inception-v3* foca-se em gastar menos recursos computacionais, ao modificar as arquiteturas *Inception* anteriores [16]. A arquitetura ilustrada na figura 2.18 da mesma contém os seguintes passos:

- Convoluções fatorizadas — Ajuda a reduzir o número de parâmetros envolvidos;
- Convoluções reduzidas — A troca de convoluções maiores por menores, leva a um treino mais rápido;
- Convoluções assimétricas — A troca de convoluções de $N * N$ por $N * 1$ e seguido por $1 * N$;
- Classificador auxiliar — Durante o treino, uma pequena RNC inserida entre camadas emite uma perda, sendo que esta é incluída na perda principal da *Inception-v3*;

- Redução da grelha — Para melhorar a eficiência computacional, usa-se um método que divide a dimensão do *input* em 2 e duplica a quantidade de filtros 2.17.

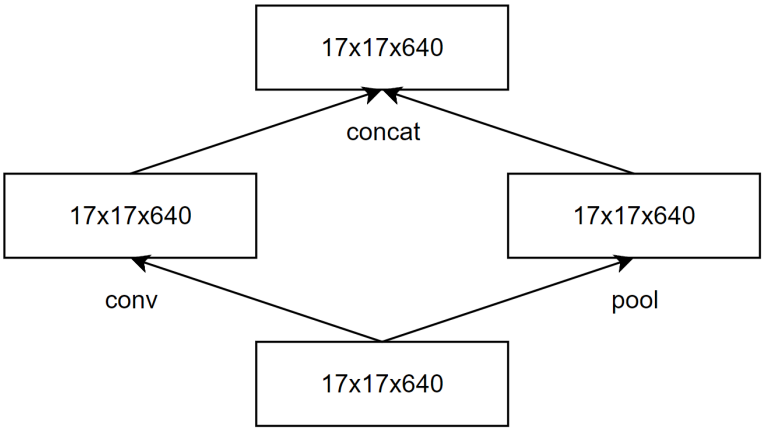


Figura 2.17: Método usado para diminuir o *input* na *Inception-V3*, retirado de [16].

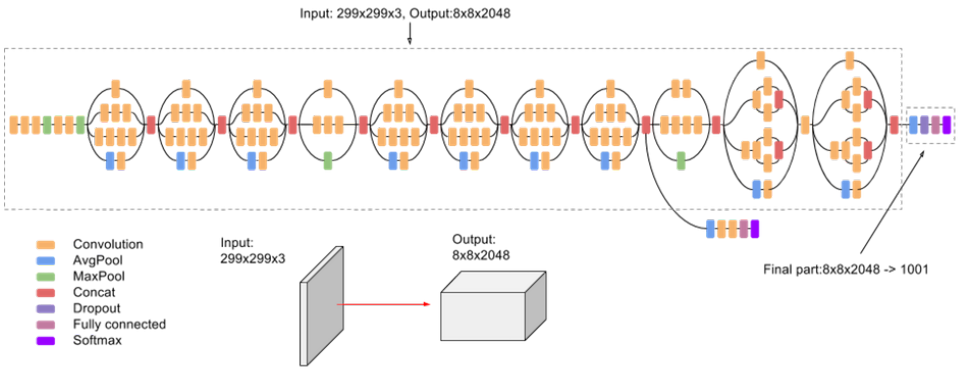


Figura 2.18: Arquitetura da *Inception-V3*, retirada de [16].

2.5.4 *Inception-ResNet-v2*

A *Inception-ResNet-v2* é um modelo pré-treinado (com mais de 1 milhão de imagens) que consegue classificar até 1000 categorias de objetos dentro de uma imagem [17]. Como consequência, o modelo conseguiu aprender várias características dentro de uma grande quantidade de imagens.

A mesma possui uma arquitetura de 164 camadas, como está representada na figura 2.19.

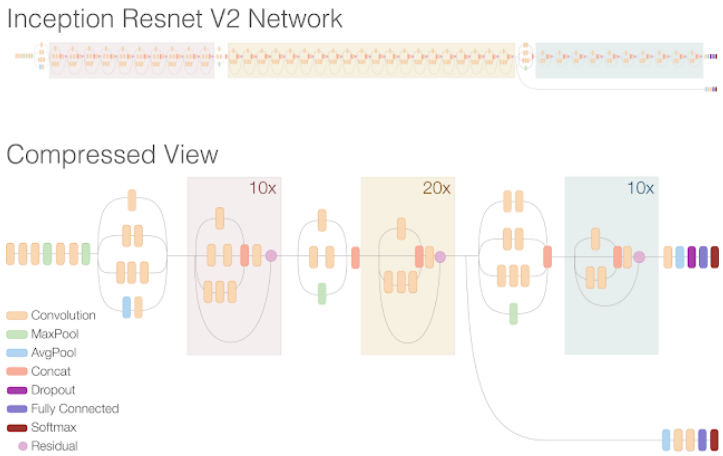


Figura 2.19: Arquitetura da *Inception-ResNet-v2* retirada de [17]

2.6 *CheXNet*

Como está referido em [24], o *CheXNet* é um método que implementa uma RNC com um classificador específico (ver secção 3.4.2), de modo a obter os melhores resultados possíveis com o *dataset* do *CheXpert* [20].

A implementação deste método é fundamental para analisar anomalias em imagens radiográficas do tórax e pode, potencialmente, expandir o acesso dos diagnósticos.

2.7 Conclusões

Com este capítulo, foi possível perceber o que existe atualmente na área do DL. No entanto, a sua dificuldade de implementação torna-se um desafio para qualquer jovem investigador.

Deste modo, o desenvolvimento e investigação da IA permite a sua aplicação e expansão para outros campos para além da Informática. Neste relatório foi abordado principalmente a sua aplicabilidade em imagem médica e/ou clínica, mas é igualmente possível a sua implementação noutros âmbitos tais como *face recognition* e *object detection*.

Esta é uma ferramenta muito útil no auxílio do diagnóstico, permitindo, em caso de redundância do mesmo, uma maior certeza relativamente à patologia em questão.

Capítulo

3

Métodos

3.1 Introdução

As primeiras etapas deste projeto implicam avaliar os dados mais relevantes, bem como o seu tratamento. Para tal, neste capítulo estão mencionados os passos necessárias para o efeito.

A secção 3.2 contém as tecnologias e ferramentas utilizadas no contexto do projeto. Na secção 3.3 encontra-se mencionado o *dataset* usado e a sua organização. A secção 3.4, refere-se às várias implementações, desde:

- Um classificador específico, com o objetivo de equilibrar, entre categorias, as que se encontrem em desvantagem numérica (secção 3.4.2);
- Um algoritmo de pré-processamento sobre o conjunto de treino de forma a melhorar resultados (secção 3.4.3);
- Um otimizador eficiente que consiga resolver problemas não convexos (secção 3.4.4);
- Uma estrutura de dados para facilitar a implementação dos modelos a partir da *framework keras* (secção 3.4.5).

Por fim, na secção 3.5 aborda-se um método de interpretação visual de modelos.

3.2 Tecnologias e Ferramentas Utilizadas

As ferramentas utilizadas no desenvolvimento deste projeto são:

3.2.1 PyCharm

Integrated Development Environment (IDE) para a linguagem *Python*, que permite uma grande variedade de ferramentas dentro de um ambiente virtual convenientemente usado para o desenvolvimento na área de ciência de dados [25].

3.2.2 *TensorFlow*

O *tensorflow* é uma biblioteca *open-source* usada na área de *Machine Learning* (ML) que providencia uma vasta quantidade de ferramentas, bibliotecas e ajuda comunitária. [26]

3.2.3 *Keras*

O *keras* é uma *Application Programming Interface* (API) *open-source* usada no DL simultaneamente com o *tensorflow* 3.2.2, permitindo um desenvolvimento mais facilitado ao programador [27].

3.3 Dados

Para treinar foi usado o *CheXpert*, citado em [20], um *dataset* público com mais de 224.000 radiografias do tórax onde as mesmas pertencem a mais de 65 mil pacientes.

Os dados foram selecionados de estudos radiográficos torácicos do Hospital de *Stanford*, entre 2002 e 2017. Dentro desses dados prevaleceram 14 categorias de observações como demonstra na tabela 3.1. Sem essa seleção por parte do *CheXpert*, seria necessário um especialista na área de radiografia do tórax para conseguir organizar e categorizar as várias imagens.

3.4 Treino

Antes de efetuar esta fase teve-se em conta os problemas referidos em 2.5. Nos mesmos, o *Patient Overlap* mostrou uma grande importância na divisão do *dataset* pelos vários conjuntos de dados. A tabela 3.1 mostra que existe uma possibilidade de incerteza dentro do *dataset*, pelo que nesta fase

ignoraram-se os dados incertos. Com esta informação, o *dataset* foi formatado, à *priori*, como está referido em [28], para usar cerca de 112.000 imagens obtendo as categorias de treino presentes na tabela 3.2.

Patologia	**Quantidade**(%)	**Incerteza**(%)	**Negativos**(%)
No Finding	8,86	0	91,14
Enlarged Cardiom.	4,81	5,41	89,78
Cardiomegaly	12,26	3,52	84,23
Lung Lesion	3,65	0,57	95,78
Lung Opacity	49,39	2,31	48,3
Edema	26,06	6,17	67,77
Consolidation	6,78	12,78	80,44
Pneumonia	2,44	8,34	89,22
Atelectasis	15,63	15,66	68,71
Pneumothorax	9,23	1,42	89,35
Pleural Effusion	40,34	5,02	54,64
Pleural Other	1,3	0,94	97,76
Fracture	3,87	0,26	95,87
Support Devices	56,4	0,48	43,12

Tabela 3.1: Tabela com as patologias a analisar presentes no *dataset*, citado em [20]

Implementou-se, também, o *dataset* 3.3 com três tipos de RNC, concretamente *DenseNet121*, *Inception-v3* e *Inception-ResNet-v2*. Para estas arquiteturas obterem os melhores resultados possíveis, o *dataset* foi divido em três conjuntos de dados 3.4.1:

- Treino — cerca de 100.000;
- Validação — cerca de 6.300;
- Teste — cerca de 1.500.

Patologia
Atelectasis
Cardiomegaly
Consolidation
Edema
Effusion
Emphysema
Fibrosis
Hernia
Infiltration
Mass
Nodule
Pleural Thickening
Pneumonia
Pneumothorax

Tabela 3.2: Tabela com as patologias a analisar presentes no *dataset*, citado em [20]

3.4.1 Divisão do *dataset*

Na divisão do *dataset*, o *CheXpert* [20] teve em conta o *Patient Overlap* e disponibiliza ficheiros no formato *Comma-separated values* (CSV) e, assim, elimina-se o problema referido, evitando que cada RNC usada para treinar não sofra de *overfitting* com tanta relevância.

3.4.2 Equilíbrio entre as observações

Outro problema referido em 2.5 é a *Class Imbalance*. Para o resolver utiliza-se um classificador ($\mathscr{L}$) específico para o problema, onde x_i é o valor a observar:

$$\mathscr{L} = -\frac{1}{N}(\sum freq_p \log(f(x_i)) + \sum freq_n \log(1 - f(x_i))).$$

$$freq_p = \frac{\text{Número de casos positivos}}{\text{Número total de casos}}$$

$$freq_n = \frac{\text{Número de casos negativos}}{\text{Número total de casos}}$$

Ao aplicar esta expressão, uma categoria que esteja em menor quantidade não será tão afetada nesta fase de treino.

3.4.3 Pré-processamento

Primeiramente, antes da adoção dos modelos, é necessário pré-processar os dados para obter os melhores resultados. Desta forma, aplicaram-se as seguintes técnicas:

- Normalização — Conversão dos dados para os valores entre [0, 1];
- *Augmentation* — Com a pouca quantidade de dados, usou-se um algoritmo de multiplicação de dados, ao mudá-los ligeiramente de posição na imagem. Assim, é possível obter mais do que 1 imagem a partir de uma original e tratar as mesmas como sendo diferentes no processo de treino;
- *Shuffle* de imagens — Neste caso, o *shuffle* adiciona aleatoriedade nos dados de treino e, deste modo, o modelo não está tão sujeito a *overfitting*.

3.4.4 Otimizador

Na implementação de um modelo é sempre necessário um algoritmo/otimizador que permita efetuar a descida do gradiente (ver secção 2.2.3). Neste caso, o otimizador usado é o *Adaptive Moment Estimation* (ADAM) [29] que permite resolver problemas não convexos, pois possui como propriedades:

- Fácil implementação;
- Eficiência computacional;
- Requisito de memória baixo;
- Apropriado nos problemas que envolvem dados e/ou parâmetros em abundância;
- Adequado em problemas com gradientes esparsos e/ou com ruído.

3.4.5 Modelos

Cada um dos modelos é construido a partir de uma *factory*. A mesma contém a estrutura necessária para cada modelo:

- Tamanho de *input* de cada imagem;
- Nome do modelo;

- Nome da última camada convolucional (usada no *Class Activation Mapping* (CAM) 3.5);
- Função de tratamento do modelo.

Nesta última, o *factory* permite instanciar modelos do *keras*, mencionado em 3.2.3. Deste modo, é possível treinar cada um dos modelos da seguinte forma:

1. Implementam-se os pesos da *ImageNet*, referido em 2.4, como base;
2. Definem-se as dimensões desejadas;
3. Colocam-se as imagens de treino e validação, mencionado na secção 3.4.3;
4. Implementa-se, sucessivamente, o otimizador e o classificador, referidos nas secções 3.4.4 e 3.4.2, respetivamente;
5. Criam-se *checkpoints* para guardar as melhores épocas de treino;
6. Usa-se um ficheiro com todas as configurações necessárias para o treino, desde a localização dos ficheiros de *output* aos hiperparâmetros(consultar tabela 3.3).
7. Por fim, instancia-se um modelo a partir do *factory* e começa-se o treino com um histórico (secção 3.4.6) de *output* no final.

3.4.6 Histórico

A relação entre input e respetivo output pode necessitar de outras variáveis para a sua compreensão pelo modelo. Assim, para poder avaliá-lo, é emitido um historial à *posteriori* com os seguintes resultados:

1. Perda do classificador aplicado ao conjunto de treino;
2. Perda do classificador aplicado ao conjunto de validação;
3. Previsão das 14 categorias, como ilustra a tabela 3.1.

Tendo estes dados estruturados, é possível construir um gráfico com:

- O cruzamento dos dados 1 e 2.
- Uma *Receiver Operating Characteristic Curve* (ROC curve) (consultar secção 4.3.1) com os dados em 3.

Nome	Descrição
output_directory	Diretoria final onde o *output* vai ser organizado;
image_source_directory	Diretoria das imagens usadas tanto em treino, teste e validação;
base_model_name	Nome do modelo a aplicar em treino e teste;
class_names	Nome das observações feitas no contexto das radiografias do tórax;
Epochs	Número das épocas de treino: 100;
batch_size	Divisão do conjunto em fatias para um processamento mais limpo e leve, evita-se um *overflow* de imagens: 8 ou 16;
learning_rate	Taxa de aprendizagem do modelo: 0.001, é mudado consoante o otimizador;
min_learning_rate	Taxa minima de aprendizagem, desta maneira o otimizador não irá diminuir em demasia: $1x10^{-8}$;
image_dimensions	Dimensão que as imagens precisam de ser processadas para serem inseridas no modelo: 224 ou 299;

Tabela 3.3: Tabela de configurações para a fase de treino.

3.5 *CAM*

Ao final do treino de um modelo consegue-se extrair os resultados do mesmo. No entanto, não se extrai nada mais do que números. Apesar de os valores se apresentarem como efetivamente corretos, há sempre a necessidade de depreender a forma de como um modelo consegue inferir os resultados e a razão desse acontecimento. O CAM [30] é uma técnica usada para interpretar esses resultados de uma forma visual através de um *heatmap* sobre a imagem original.

Na figura 3.1 à direita encontra-se a imagem radiográfica do tórax original e, após a aplicação do algoritmo CAM, este indica, visualmente, o que um dado modelo consegue prever. Assim, na imagem radiográfica da esquerda encontra-se destacado o problema em questão.

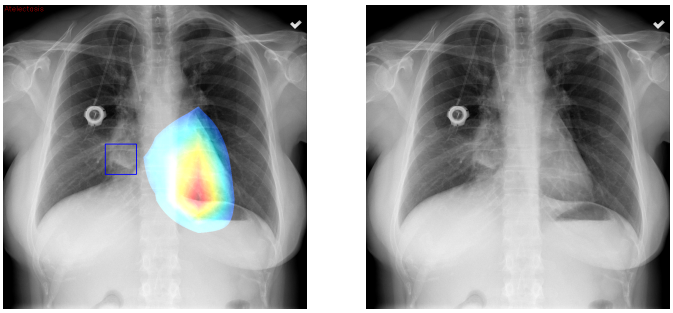


Figura 3.1: Duas imagens originais iguais. A imagem da esquerda tem o CAM implementado, enquanto a imagem da direita não foi manipulada.

3.6 Conclusões

Neste capítulo, foi possível demonstrar as várias ferramentas e tecnologias usadas na implementação de métodos na área de IA.

Demonstrou-se, também, que um *dataset* bem organizado consegue simplificar bastante o trabalho e, ainda, a importância que uma boa estruturação de algoritmos tem perante um problema tão complexo que este projeto apresenta.

Capítulo

4

Testes e resultados obtidos

4.1 Introdução

Neste capítulo são mencionados e discutidos os diversos resultados, bem como a sua melhoria perante a comparação com o método estado da arte *CheXNet* [24].

Na secção 4.2 estão referidos os modelos em questão e, ainda, o seu propósito de implementação, as métricas de avaliação e a comparação dos resultados finais com o método estado da arte.

Por fim, na secção 4.4, relativamente ao problema que este projeto expõe, discute-se a eficácia e rapidez de cada um dos modelos, em termos de reprodutibilidade e fiabilidade.

4.2 Modelos

O *CheXNet* 2.6, retirado de [24], é um método estado da arte usado para tentar classificar 14 categorias de patologia, referido em 3.4. Os pesos pré-treinados, usados na fase de testes, foram extraídos de um repositório do *GitHub* [31].

Na tentativa de superar os resultados do método do *CheXNet*, foram implementados três modelos: *DenseNet121*; *Inception-v3*; *Inception-ResNet-v2*. Portanto, houve a necessidade de comparar tanto as configurações, como os resultados de cada um dos modelos usados.

Os três modelos referidos foram escolhidos com base na leitura da secção 2.5.1 que confirma serem os modelos mais usados para o propósito deste projeto.

4.2.1 Configurações dos modelos

Para os vários modelos as configurações-base foram:

Modelo	***DenseNet121***	***Inception-v3***	***Inception-ResNet-v2***	***CheXNet DenseNet121***
Pesos-base	*ImageNet* 2.4	*ImageNet* 2.4	*ImageNet* 2.4	*ImageNet* 2.4
Batch size	16	8	8	8
Epoch	100	100	100	-
Learning Rate	0.001	0.001	0.001	0.001
Otimizador	*Adam*	*Adam*	*Adam*	*Adam*
Dimensões do *input*	224	299	299	224
Normalização	Sim	Sim	Sim	Sim
Ativação	*Sigmoid*	*Sigmoid*	*Sigmoid*	*Sigmoid*

Tabela 4.1: Tabela de configurações dos modelos mencionados.

Como se pode observar na tabela 4.1, o modelo *DenseNet121* treinado é usado também no método do *CheXNet*. Foi usada a mesma arquitetura para tentar replicar ou melhorar um método estado da arte. Tendo isso em conta, o modelo treinado contém hiperparâmetros diferentes na procura dessa melhoria, dos quais os resultados são discutidos na secção 4.3.

Para os outros dois modelos, o objetivo é o mesmo referido anteriormente, mas com arquiteturas diferentes do *CheXNet*.

4.3 Resultados dos modelos

Sempre que um modelo é treinado exige uma avaliação do seu desempenho. Para tal, existem métricas perfeitas para o efeito, como a ROC curve e a *Area Under the ROC Curve* (AUC) explicadas na secção 4.3.1.

Também é importante mostrar o progresso de treino de um dado modelo, de modo a confirmar se o mesmo treino foi efetivo. Nas secções 4.3.2, 4.3.4 e 4.3.6 estão ilustrados os gráficos de treino dos modelos *DenseNet121*, *Inception-v3* e *Inception-ResNet-v2*, respetivamente.

4.3.1 *ROC curve* e *AUC*

Como é mencionado em [32], uma ROC curve é uma métrica de avaliação para problemas binários. Neste caso, tratando-se de um problema de múlti-

plas categorias, então, foram aplicados os dados de conjunto de teste, presentes na tabela 4.2.

Patologia	Quantidade
Atelectasis	139
Cardiomegaly	57
Consolidation	76
Edema	29
Effusion	199
Emphysema	21
Fibrosis	27
Hernia	3
Infiltration	310
Mass	73
Nodule	91
Pleural Thickening	52
Pneumonia	24
Pneumothorax	47

Tabela 4.2: Tabela com a quantidade dados de teste para cada patologia

		Real Values	
		Positive	**Negative**
Predicted Values	**Positive**	*True Positive (TP)*	*False Positive (FP)*
	Negative	*False Negative (FN)*	*True Negative (TN)*

Tabela 4.3: Matriz de confusão usada na ROC curve.

Através de uma matriz de confusão, representada na tabela 4.3, conseguem-se extrair métricas importantes para a implementação de uma ROC curve. De forma a compreender a mesma é necessário introduzir os seguintes conceitos:

- *True Positive Rate* (TPR) — Percentagem de dados considerados positivos e bem avaliados pelo modelo, numa dada categoria, através da formula $TPR = \frac{TP}{TP+FN}$;
- *False Positive Rate* (FPR) — Percentagem de dados considerados negativos e bem avaliados pelo modelo, numa dada categoria, através da expressão $FPR = \frac{FP}{TN+FP}$.

A ROC curve segue uma variação num *threshold*, o qual varia entre [0, 1]. Quanto mais inclinada se encontra a curva para o canto superior esquerdo, melhor é a capacidade de um modelo generalizar uma certa categoria. O *threshold* é a margem de um dado ser classificado como positivo ou negativo. Esses valores, ao serem usados para calcular o TPR e FPR, formam uma coordenada numa ROC curve, como ilustra a figura 4.1.

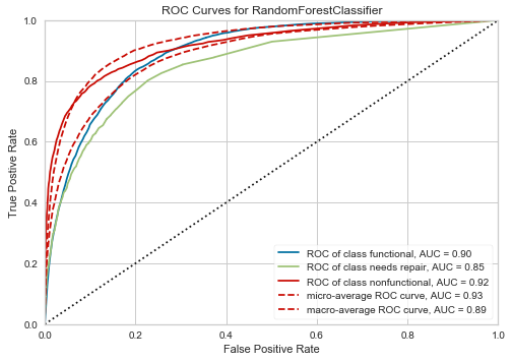


Figura 4.1: Representação da ROC curve, retirada de [18].

A AUC [19] permite usar os valores associados à ROC curve para medir uma classificação de desempenho entre todos os *thresholds*, como está ilustrado na figura 4.2. Essa medição demonstra uma boa ou má generalização do modelo avaliado, consoante o valor apresentado, sendo 0 uma péssima previsão e 1 uma ótima previsão.

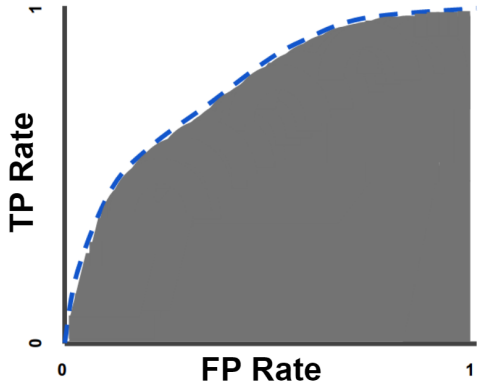


Figura 4.2: Representação da AUC, retirada de [19].

4.3.2 Histórico da *DenseNet121*

No gráfico 4.3 estão, lado a lado, os dados resultantes de 100 épocas de treino no modelo *DenseNet121*. À esquerda, encontram-se os resultados do classificador, referido em 3.4.2, no conjunto de treino e, à direita, o resultado do classificador no conjunto de validação. Consegue-se analisar, também, que o modelo começa a estagnar na aprendizagem, alcançando o melhor resultado na época 40.

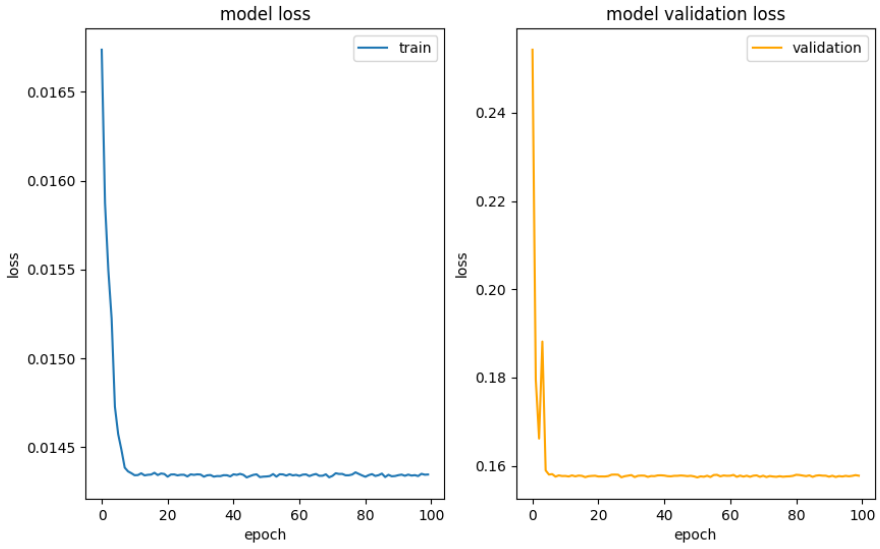


Figura 4.3: Gráfico de histórico de treino do modelo *DenseNet121*.

4.3.3 AUC-ROC curve da *DenseNet121*

O gráfico AUC-ROC curve, representado pela figura 4.4 do modelo em questão, foi construído da mesma maneira como está referido na secção 4.3.1. É de notar, também, que devido à falta de dados presentes no conjunto de treino, como a tabela 4.2 ilustra, patologias como a *Hernia* ou *Pneumonia* demonstram uma anomalia na representação da categoria na ROC curve. Estes dados são discutidos na secção 4.3.9.

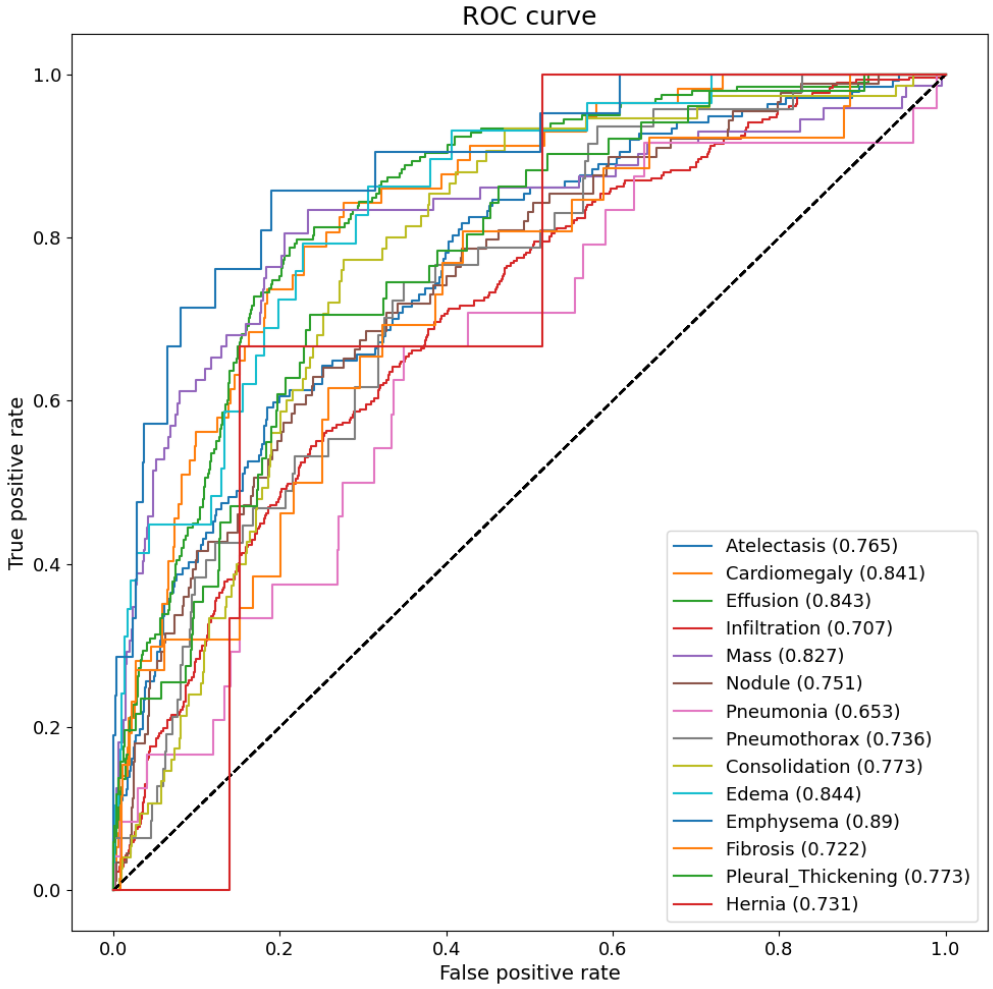


Figura 4.4: Gráfico com a AUC-ROC curve do modelo *DenseNet121*.

4.3.4 *Inception-v3*

No gráfico 4.5 estão, lado a lado, os dados resultantes de 100 épocas de treino no modelo *Inception-v3*. À esquerda, encontram-se os resultados do classificador, referido em 3.4.2, no conjunto de treino e, à direita, o resultado do classificador no conjunto de validação. Consegue-se analisar, também, que o modelo começa a estagnar na aprendizagem, alcançando o melhor resultado na época 36.

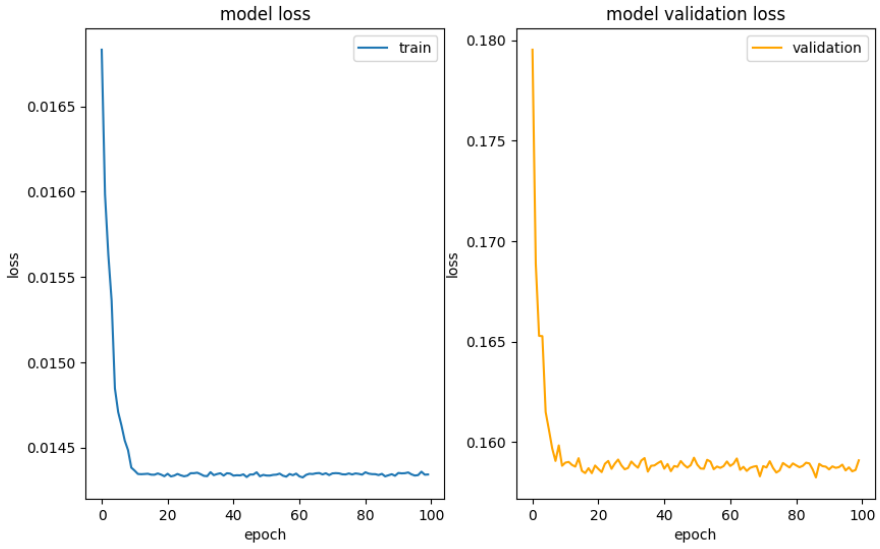


Figura 4.5: Gráfico de histórico de treino do modelo *Inception-v3*.

4.3.5 AUC-ROC curve da *Inception-v3*

O gráfico AUC-ROC curve, representado pela figura 4.6 do modelo em questão, foi construído da mesma maneira como está referido na secção 4.3.1. É de notar, também, que devido à falta de dados presentes no conjunto de treino, como a tabela 4.2 ilustra, patologias como a *Hernia* ou *Pneumonia* demonstram uma anomalia na representação da categoria na ROC curve. Estes dados são discutidos na secção 4.3.9.

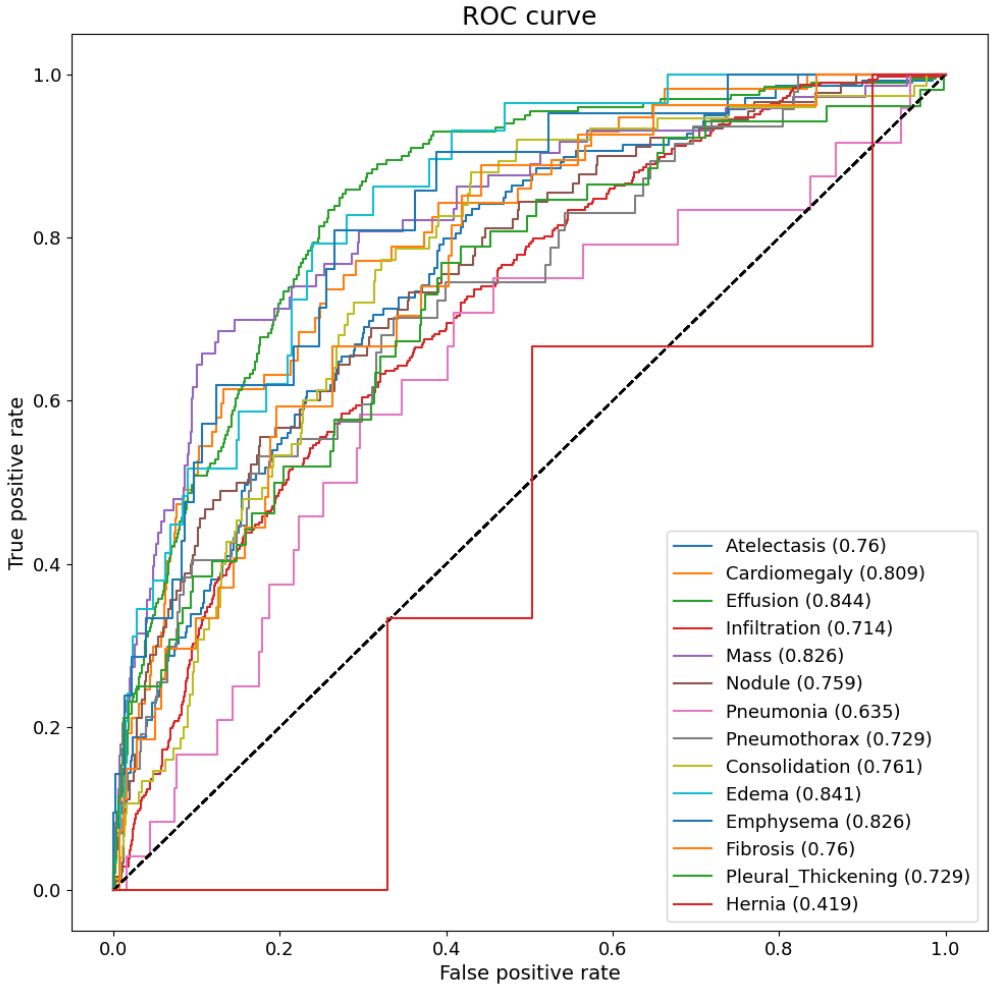


Figura 4.6: Gráfico com a AUC-ROC curve do modelo *Inception-v3*.

4.3.6 *Inception-ResNet-v2*

No gráfico 4.7 estão, lado a lado, os dados resultantes de 100 épocas de treino no modelo *Inception-ResNet-v2*. À esquerda, encontram-se os resultados do classificador, referido em 3.4.2, no conjunto de treino e, à direita, o resultado do classificador no conjunto de validação. Consegue-se analisar, também, que o modelo começa a estagnar na aprendizagem, alcançando o melhor resultado na época 73.

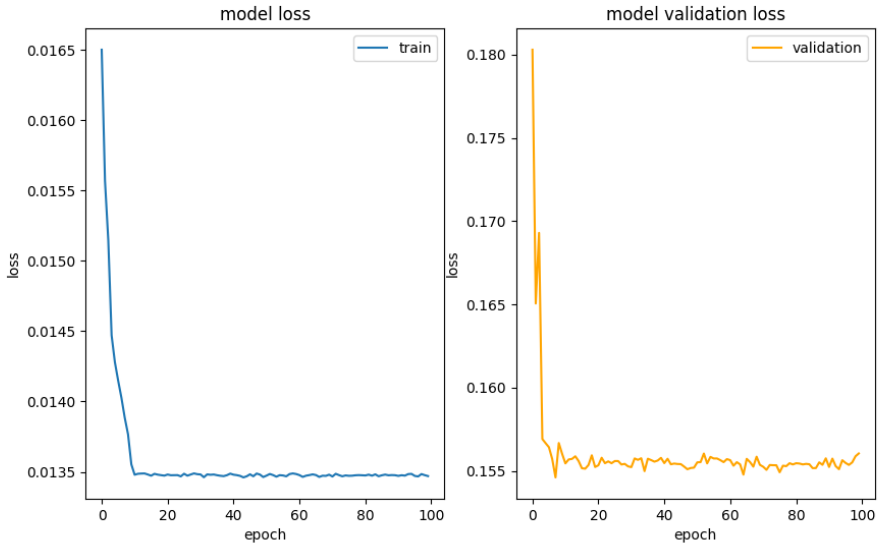


Figura 4.7: Gráfico de histórico de treino do modelo *Inception-ResNet-v2*.

4.3.7 AUC-ROC curve da *Inception-ResNet-v2*

O gráfico AUC-ROC curve, representado pela figura 4.8 do modelo em questão, foi construído da mesma maneira como está referido na secção 4.3.1. É de notar, também, que devido à falta de dados presentes no conjunto de treino, como a tabela 4.2 ilustra, patologias como a *Hernia* ou *Pneumonia* demonstram uma anomalia na representação da categoria na ROC curve. Estes dados são discutidos na secção 4.3.9.

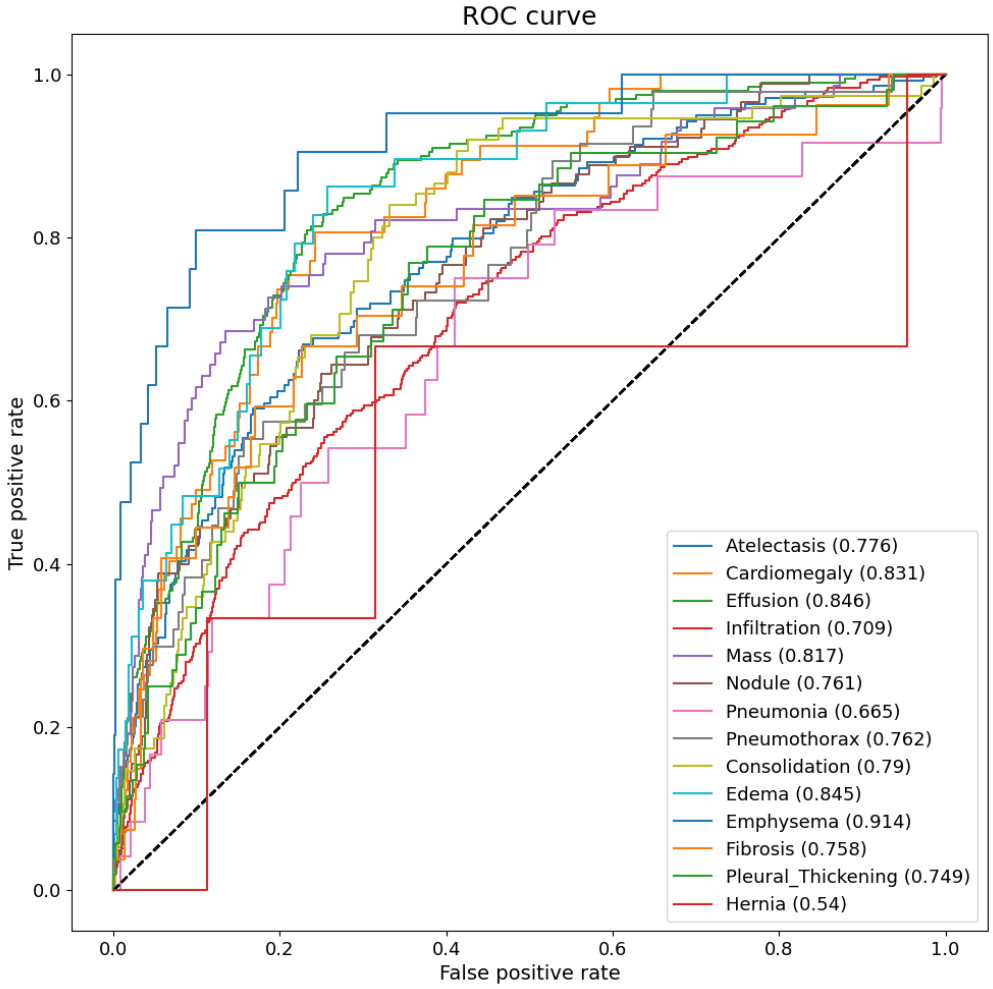


Figura 4.8: Gráfico com a AUC-ROC curve do modelo *Inception-ResNet-v2*.

4.3.8 *CheXNet*

No caso do método *CheXNet*, os dados de teste, que estão demonstrados no gráfico 4.9, ilustram a generalização que o método estado da arte consegue atingir nas 14 patologias. É de notar, também, que devido à falta de dados presentes no conjunto de treino, como a tabela 4.2 ilustra, patologias como a *Hernia* ou *Pneumonia* demonstram uma anomalia na representação da categoria na ROC curve, sendo que os resultados são discutidos na secção 4.3.9.

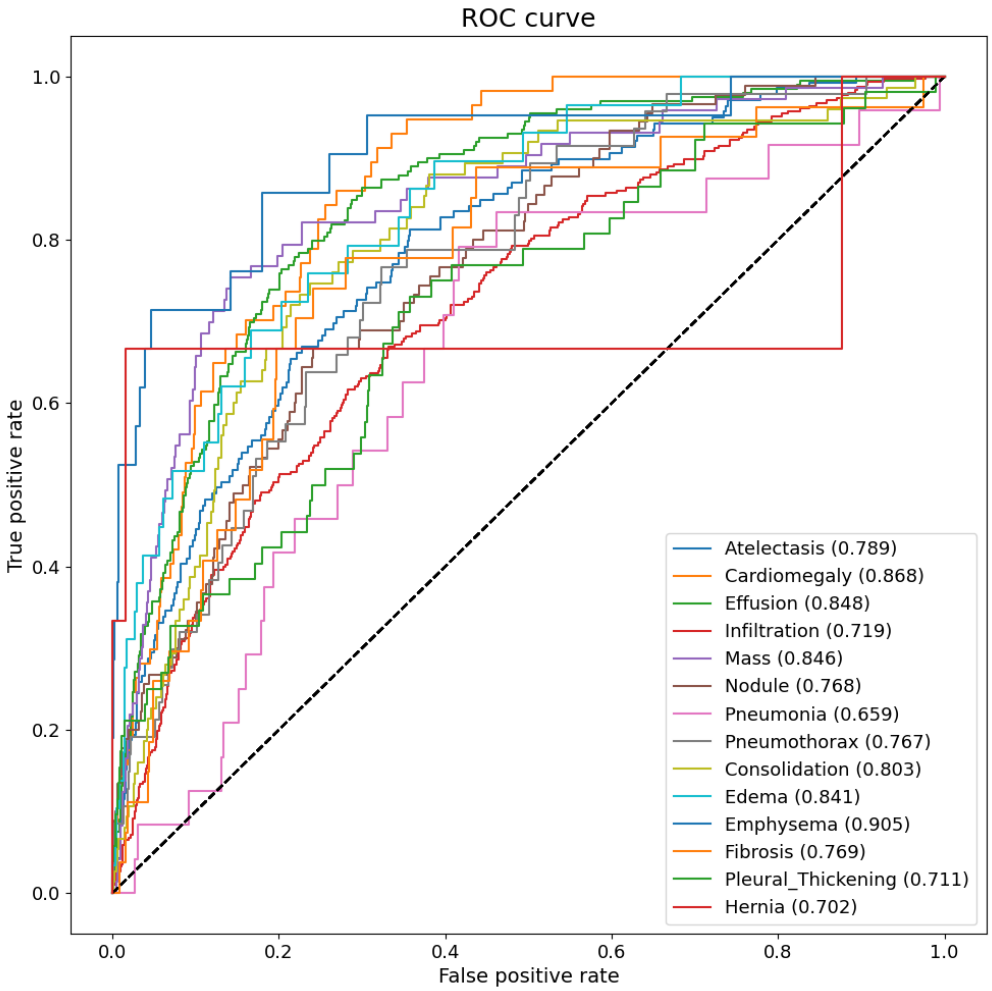


Figura 4.9: Gráfico com a AUC-ROC curve do modelo *CheXNet*.

4.3.9 Discussão dos Resultados

Patologia	*CheXNet*	*DenseNet121*	*Inception-v3*	*Inception-ResNet-v2*
Atelectasis	**0.789**	0.765	0.760	0.776
Cardiomegaly	**0.868**	0.841	0.809	0.831
Effusion	**0.848**	0.843	0.844	0.846
Infiltration	**0.719**	0.707	0.714	0.709
Mass	**0.846**	0.827	0.826	0.817
Nodule	**0.768**	0.751	0.759	0.761
Pneumonia	0.659	0.653	0.635	**0.665**
Pneumothorax	**0.767**	0.736	0.729	0.762
Consolidation	**0.803**	0.773	0.761	0.790
Edema	0.841	0.844	0.841	**0.845**
Emphysema	0.905	0.890	0.826	**0.914**
Fibrosis	**0.769**	0.722	0.760	0.758
Pleural Thickening	0.711	**0.773**	0.729	0.749
Hernia	0.702	**0.731**	0.419	0.54

Tabela 4.4: Comparação do desempenho dos modelos *DenseNet121*, *Inception-v3* e *Inception-ResNet-v2* e do método *CheXNet* na classificação 14 patologias. Os valores apresentados dizem respeito à AUC.

Na tabela 4.4 podem-se notar diferenças significativas nos valores da AUC de cada um dos modelos. O método *CheXNet* teve um melhor desempenho do que os outros modelos em 9 patologias e pior nas outras 5. Nas patologias *Atelectasis*, *Cardiomegaly*, *Effusion*, *Infiltration*, *Mass*, *Nodule*, *Pneumothorax*, *Consolidation* e *Fibrosis* alcançou uma AUC de 0.789, 0.868, 0.848, 0.719, 0.846, 0.768, 0.767, 0.803 e 0.769, respetivamente.

Em comparação ao *CheXNet*, o modelo *Inception-ResNet-v2* obteve resultados superiores ao método estado da arte nas patologias *Pneumonia*, *Edema* e *Emphysema* com uma AUC de 0.665, 0.845 e 0.914, respetivamente, enquanto que o modelo *DenseNet121* conseguiu superar em duas patologias, nomeadamente *Pleural Thickening* e *Hernia* com uma AUC de 0.773 e 0.731, respetivamente.

É de referir, igualmente, que o modelo *Inception-v3* não conseguiu ultrapassar o método estado da arte em nenhuma categoria, no entanto, manteve-se sempre semelhante aos outros modelos, nos resultados obtidos.

Patologia	***ChexNet***	***Ensemble***	***Denset-Ne121***	***Inception-v3***	***Inception-ResNet-v2***
Atelectasis	**72.22%**	64.48%	63.89%	65.81%	60.45%
Cardio-megaly	**82.94%**	77.98%	70.44%	71.10%	80.75%
Effusion	78.57%	**79.17%**	78.77%	78.24%	78.17%
Infiltration	75.40%	76.65%	69.11%	**79.63%**	74.47%
Mass	73.94%	**82.80%**	82.01%	80.36%	78.24%
Nodule	78.31%	76.98%	63.36%	74.60%	**83.33%**
Pneumonia	**78.64%**	76.52%	74.80%	72.22%	75.99%
Pneumo-thorax	**85.12%**	84.99%	84.59%	79.30%	83.73%
Consoli-dation	54.37%	52.25%	48.21%	52.51%	**57.34%**
Edema	**81.35%**	72.95%	73.08%	70.30%	71.36%
Emphysema	**87.76%**	84.39%	85.71%	77.05%	82.94%
Fibrosis	70.83%	80.75%	68.52%	**97.02%**	67.66%
Pleural Thickening	70.83%	69.18%	59.52%	68.72%	**71.03%**
Hernia	**92.72%**	80.22%	74.54%	83.66%	67.26%
Média	**77.36%**	75.67%	71.18	75.04%	73.77%

Tabela 4.5: Tabela com os resultados da avaliação do método estado da arte, da técnica *Ensemble* e modelos *DenseNet121*, *Inception-v3* e *Inception-ResNet-v2* implementados ao longo do projeto. Para efeitos de comparação, a negrito encontram-se os melhores resultados e a sublinhado os segundos melhores resultados, por patologia.

Após a aplicação da técnica *Ensemble*, é possível mostrar como os três modelos treinados conseguem inferir, através do voto por maioria. Desta maneira, é possível combinar a previsão dos vários modelos para visualizar uma previsão ótima, em comparação com cada modelo individual e com o método estado da arte.

Na tabela 4.5 os dados ilustrados foram atingidos durante a fase de avaliação dos diferentes modelos, sobre o conjunto de avaliação. Primeiramente, o *Ensemble* obteve, nas patologias *Effusion* e *Mass*, os resultados 79.17% e 82.80%, respetivamente. O método *CheXNet* superou as outras implemen-

tações com resultados de 72.22%, 82.94%, 78.64%, 85.12%, 81,35%, 87.76% e 92,72% nas patologias *Atelectasis*, *Cardiomegaly*, *Pneumonia*, *Pneumothorax*, *Edema*, *Emphysema* e *Hernia*, respetivamente. Adicionalmente, o modelo *Inception-ResNet-v2* alcançou resultados de 83,33%, 57,34% e 71.03% nas patologias *Nodule*, *Consolidation* e *Pleural Thickening*, por essa ordem. Por fim e com o melhor resultado de avaliação, o modelo *Inception-v3* atingiu um resultado de 97.02% na patologia *Fibrosis*.

Nome	**Taxa de acerto no conjunto de teste**
CheXNet	75.79%
Ensemble	72.30%
Inception-v3	71.56%
Inception-ResNet-v2	71.02%
DenseNet121	68.73%

Tabela 4.6: Taxa de acerto que o método estado da arte *CheXNet*, a técnica *Ensemble* e os modelos *DenseNet121*, *Inception-v3* e *Inception-ResNet-v2* obtiveram, ao serem submetidos no conjunto de teste.

Na tabela 4.6 estão referidos os resultados médios que cada implementação atingiu sobre os dados presentes no conjunto de teste, para simular os resultados de uma execução num cenário realista. Neste caso, o método *CheXNet* superou as outras implementações, no entanto, a técnica *Ensemble* provou obter bons resultados, acima dos outros três modelos treinados.

É de salientar que o *dataset*, ao ser analisado por especialistas, pode ter um impacto significativo sobre os resultados, visto que, na secção 3.3, foi referido que o *dataset* inicial abrangia resultados incertos e nem todos podem ter sido bem analisados, por erro humano.

4.4 Interpretabilidade

Foi referido, na secção 3.5, que o CAM é uma técnica usada para verificar onde um dado modelo se foca mais numa imagem, após a sua análise. Um radiologista demora em média 4 horas para catalogar 420 radiografias do tórax [33]. O algoritmo CAM usado no contexto do projeto necessitou cerca de 7 minutos, por modelo, para catalogar cerca de 980 radiografias do tórax. As figuras 4.10, 4.11 e 4.12 apresentam um exemplo desse algoritmo nos modelos *DenseNet121*, *Inception-v3* e *Inception-ResNet-v2*, respetivamente.

Uma ferramenta com esta capacidade seria muito eficaz no diagnóstico e no acesso à interpretação das radiografias do tórax.

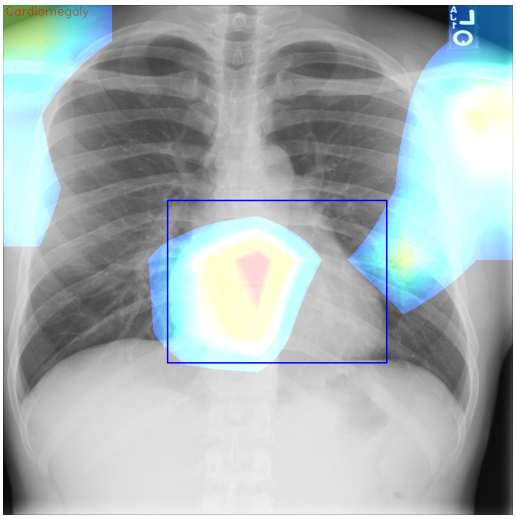


Figura 4.10: Resultado do algoritmo CAM numa imagem radiográfica do tórax, categorizada como *Cardiomegaly*, pelo modelo *DenseNet121*.

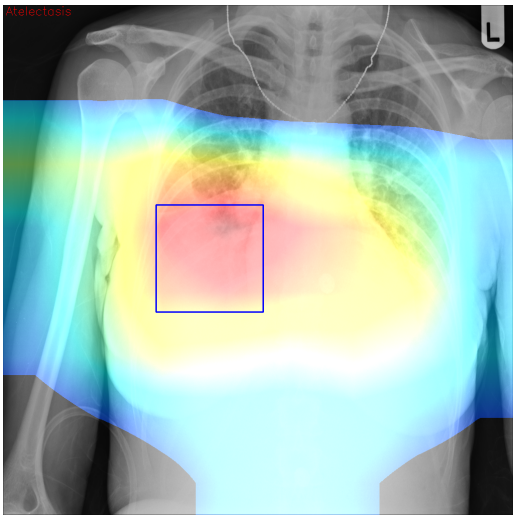


Figura 4.11: Resultado do algoritmo CAM numa imagem radiográfica do tórax, categorizada como *Atelectasis*, pelo modelo *Inception-v3*.

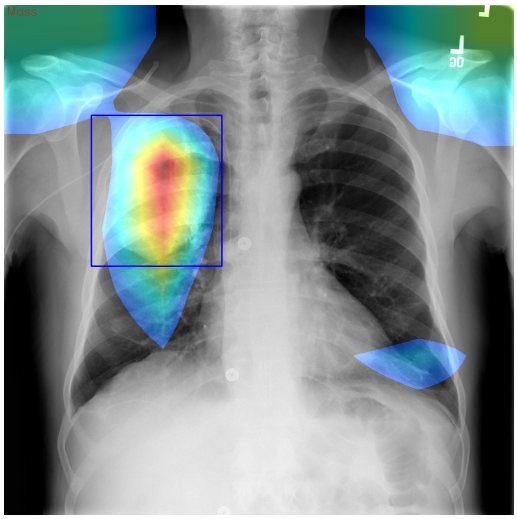


Figura 4.12: Resultado do algoritmo CAM numa imagem radiográfica do tórax, categorizada como *Mass*, pelo modelo *Inception-ResNet-v2*.

No entanto, a própria ferramenta possui limitações e, apesar da análise de imagens relativamente parecidas em diferentes categorias, o algoritmo CAM falha na interpretação das radiografias do tórax, como ilustra a figura 4.13. Na figura 4.12, o algoritmo CAM projeta a sua representação sobre uma figura original onde há a possibilidade da existência de zonas erróneas.

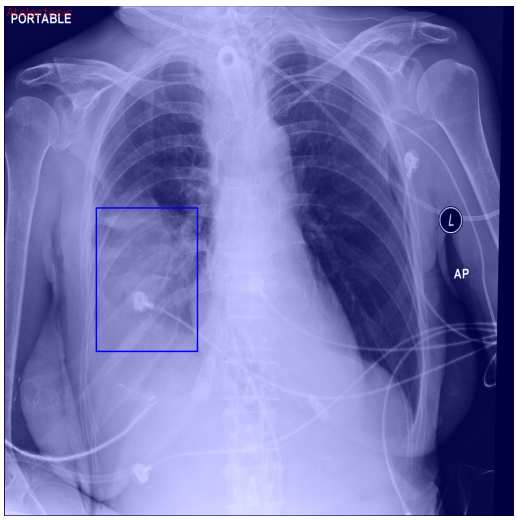


Figura 4.13: Resultado impreciso do algoritmo CAM numa imagem radiográfica do tórax, categorizada como *Atelectasis*, pelo modelo *Inception-ResNet-v2*.

Para contornar as falhas previstas pelo algoritmo CAM, o uso da técnica *Ensemble*, referida na tabela 4.5, permite colmatar essas falhas ao visualizar a interpretação de três modelos concorrentemente.

Nas figuras 4.14, 4.15 e 4.16 estão ilustrados os resultados que cada modelo consegue obter com o uso do algoritmo CAM. Como cada um destes modelos consegue inferir cada patologia de forma distinta, então, a probabilidade de obter uma boa interpretação é maior. As figuras referidas, anteriormente, demonstram a vantagem de usar os três modelos na interpretação de imagens radiográficas do tórax e a desvantagem ao usar apenas um modelo para o efeito.

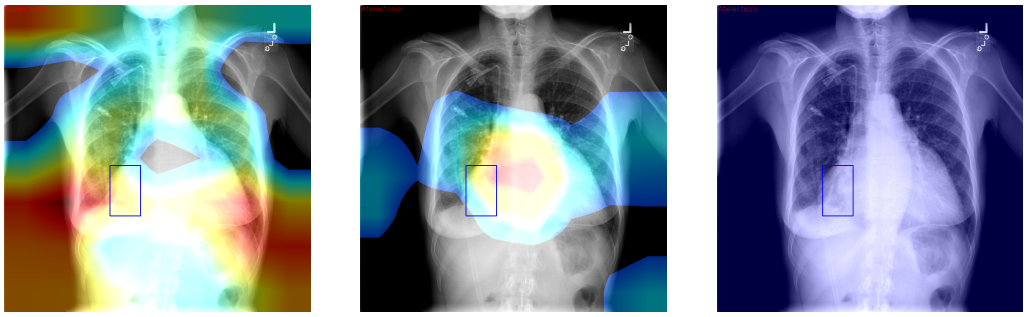


Figura 4.14: Resultado do algoritmo CAM numa imagem radiográfica do tórax, categorizada como *Atelectasis*. Neste exemplo estão ilustrados, da esquerda para a direita, os resultados dos modelos *DenseNet121*, *Inception-v3* e *Inception-ResNet-v2*, respetivamente.

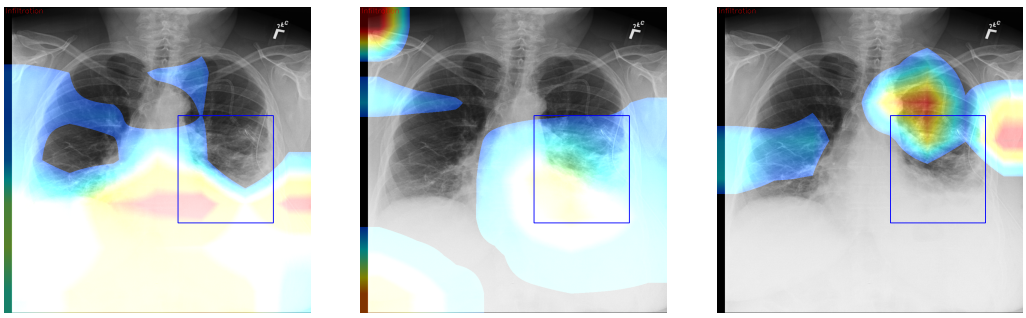


Figura 4.15: Resultado do algoritmo CAM numa imagem radiográfica do tórax, categorizada como *Infiltration*. Neste exemplo estão ilustrados, da esquerda para a direita, os resultados dos modelos *DenseNet121*, *Inception-v3* e *Inception-ResNet-v2*, respetivamente.

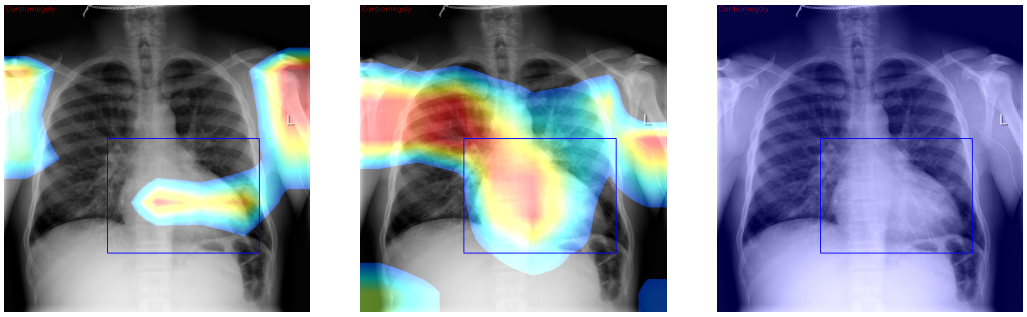


Figura 4.16: Resultado do algoritmo CAM numa imagem radiográfica do tórax, categorizada como *Cardiomegaly*. Neste exemplo estão ilustrados, da esquerda para a direita, os resultados dos modelos *DenseNet121*, *Inception-v3* e *Inception-ResNet-v2*, respetivamente.

4.5 Conclusões

Neste capítulo, demonstrou-se a possibilidade de alcançar resultados comparáveis a um método estado da arte. Também foi possível analisar, visualmente, o comportamento que os modelos treinados têm perante cada imagem através do algoritmo CAM. Apesar deste possuir algumas limitações, a sua implementação pode ajudar profissionais, na análise de radiografias do tórax.

Capítulo

5

Conclusões

5.1 Conclusões Principais

Com este trabalho, foi possível alargar o meu leque de conhecimento, relativamente à área de IA, mais concretamente, na área de DL. Esta encontra-se em constante evolução e consegue superar, cada vez mais, as expectativas dos profissionais nas áreas onde a mesma pode ser implementada.

Pretendi construir o meu modelo baseado em DL para analisar imagens médicas. O modelo em questão, com trabalho suficiente, consegue ser aplicado em outras áreas para além de imagens médicas. O acompanhamento de outros métodos estado da arte facilitou o desenvolvimento do próprio algoritmo e, assim, obter resultados melhorados, como descrito na secção 4.

Dos três modelos implementados apenas dois superaram o método estado da arte, relativamente aos valores da AUC. O *DenseNet121* adquiriu um resultado de 0.773 e 0.731 nas patologias *Pleural Thickening* e *Hernia*, respetivamente. Por sua vez, o *Inception-ResNet-v2* atingiu um resultado de 0.665, 0.845 e 0.914 nas patologias *Pneumonia*, *Edema* e *Emphysema*, por essa ordem. O *Inception-v3* não obteve nenhum resultado superior aos restantes, no entanto, manteve-se próximo dos mesmos.

Em termos de avaliação, o *Inception-v3* foi o modelo que se destacou, em relação às restantes implementações, com um resultado de 97.02%. Ainda é possível referir que o *Ensemble* mostrou ser uma técnica com resultados superiores às restantes implementações, à exceção do método estado da arte, com um resultado de 72.30% num conjunto de teste.

Por fim, pela aplicação do algoritmo CAM nos três modelos referidos, anteriormente, foi possível contornar as limitações que o próprio possui, ao usar os mesmos em conjunto.

Em suma, todos os objetivos propostos, para a evolução do projeto, foram alcançados com sucesso.

Bibliografia

[1] DeepAI. Synapse, 2019. [Online; ultimo acesso 24 de março de 2022].

[2] Towards Data Science. [Online; ultimo acesso 14 de abril de 2022].

[3] MeetUp. [Online; ultimo acesso 14 de abril de 2022].

[4] Research Gate. [Online; ultimo acesso 14 de abril de 2022].

[5] Towards Data Science. [Online; ultimo acesso 14 de abril de 2022].

[6] Research Gate. [Online; ultimo acesso 14 de abril de 2022].

[7] Andrew Ng. Neural networks and deep learning, 2018. [Online; ultimo acesso 9 de abril de 2022].

[8] VitalFlux. [Online; ultimo acesso 14 de abril de 2022].

[9] Research Gate. [Online; ultimo acesso 14 de abril de 2022].

[10] Medium. [Online; ultimo acesso 14 de abril de 2022].

[11] Towards Data Science. [Online; ultimo acesso 14 de abril de 2022].

[12] Towards Data Science. [Online; ultimo acesso 14 de abril de 2022].

[13] Towards Data Science. [Online; ultimo acesso 14 de abril de 2022].

[14] Towards Data Science. [Online; ultimo acesso 14 de abril de 2022].

[15] Research Gate. [Online; ultimo acesso 16 de abril de 2022].

[16] PaperSpaceBlog. [Online; ultimo acesso 25 de abril de 2022].

[17] Medium. [Online; ultimo acesso 10 de junho de 2022].

[18] How to plot ROC curves in multiclass classification? — stats.stackexchange.com. `https://stats.stackexchange.com/questions/2151/how-to-plot-roc-curves-in-multiclass-classification`. [Accessed 14-Jun-2022].

[19] Classification: ROC Curve and AUC | Machine Learning Crash Course | Google Developers — developers.google.com. https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc. [Accessed 16-Jun-2022].

[20] Jeremy Irvin, Pranav Rajpurkar, Michael Ko, Yifan Yu, Silviana Ciurea-Ilcus, Chris Chute, Henrik Marklund, Behzad Haghgoo, Robyn Ball, Katie Shpanskaya, Jayne Seekins, David A. Mong, Safwan S. Halabi, Jesse K. Sandberg, Ricky Jones, David B. Larson, Curtis P. Langlotz, Bhavik N. Patel, Matthew P. Lungren, and Andrew Y. Ng. Chexpert: A large chest radiograph dataset with uncertainty labels and expert comparison, 2019.

[21] ImageNet — image-net.org. https://www.image-net.org/. [Accessed 14-Jun-2022].

[22] Jiechao Ma1, Yang Song2, Xi Tian Yiting Hua Rongguo Zhang, and Jianlin Wu. Survey on deep learning for pulmonary medical imaging. *InferVision, Beijing 100020, China and Dalian Municipal Central Hospital Affiliated to Dalian Medical University, Dalian 116033, China and Affiliated Zhongshan Hospital of Dalian University, Dalian 116001, China*, page 3, 2019.

[23] Qing Guan1, Xiaochun Wan, Hongtao Lu, Bo Ping, Duanshu Li, Li Wang, Yongxue Zhu, Yunjun Wang, and Jun Xiang. Deep convolutional neural network inception-v3 model for differential diagnosing of lymph node in cytological images: a pilot study. *Department of Head and Neck Surgery, Fudan University Shanghai Cancer Center, Shanghai 200032, China and Department of Oncology, Shanghai Medical College, Fudan University, Shanghai 200032, China and Department of Pathology, Fudan University Shanghai Cancer Center, Shanghai 200032, China and Depertment of Computer Science and Engineering, Shanghai Jiaotong University, Shanghai 200240, China*, page 3, 2019.

[24] CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning — arxiv.org. https://arxiv.org/abs/1711.05225. [Accessed 22-Jun-2022].

[25] PyCharm: the Python IDE for Professional Developers by JetBrains — jetbrains.com. https://www.jetbrains.com/pycharm/. [Accessed 14-Jun-2022].

[26] TensorFlow — tensorflow.org. https://www.tensorflow.org/. [Accessed 14-Jun-2022].

[27] Keras Team. Keras: the Python deep learning API — keras.io. https://keras.io/. [Accessed 14-Jun-2022].

[28] Box — nihcc.app.box.com. https://nihcc.app.box.com/v/ChestXray-NIHCC/file/220660789610. [Accessed 16-Jun-2022].

[29] MachineLearningMastery. Gentle Introduction to the Adam Optimization Algorithm for Deep Learning - Machine Learning Mastery — machinelearningmastery.com. https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/. [Accessed 14-Jun-2022].

[30] CAM: Visual Explanations from Deep Networks — glassboxmedicine.com. https://glassboxmedicine.com/2020/05/29/grad-cam-visual-explanations-from-deep-networks/. [Accessed 14-Jun-2022].

[31] GitHub - brucechou1983/CheXNet-Keras: This project is a tool to build CheXNet-like models, written in Keras. — github.com. https://github.com/brucechou1983/CheXNet-Keras. [Accessed 22-Jun-2022].

[32] AUC-ROC Curve in Machine Learning Clearly Explained - Analytics Vidhya — analyticsvidhya.com. https://www.analyticsvidhya.com/blog/2020/06/auc-roc-curve-machine-learning/. [Accessed 14-Jun-2022].

[33] Deep learning for chest radiograph diagnosis: A retrospective comparison of the CheXNeXt algorithm to practicing radiologists — journals.plos.org. https://journals.plos.org/plosmedicine/article?id=10.1371/journal.pmed.1002686#sec015. [Accessed 14-Jun-2022].

[34] Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Mohsen Ghafoorian Francesco Ciompi, Jeroen A.W.M. van der Laak, Bram van Ginneken, and Clara I. Sanchez. A survey on deep learning in medical image analysis. *Diagnostic Image Analysis Group*, 2:1–2, 2017.

[35] Jean-Pierre Briot, Gaetan Hadjeres, and Franc¸ois-David Pachet. Deep learning techniques for music generation – a survey. *Sorbonne Universite and Sony Computer Science Laboratories and Spotify Creator Technology Research Lab*, page 3, 2019.

[36] Pranav Rajpurkar and Bora Uyumazturk and Amirhossein Kiani and Eddy Shyu. Ai for medicine specialization, 2020. [Online; ultimo acesso 14 de abril de 2022].

[37] Najmul Hasan, Yukun Bao, Ashadullah Shawon, and Yanmei Huang. Densenet convolutional neural networks application for predicting covid-19 using ct image. *PubMed Central*, 2021.