Breast Histopathology Dataset

O *Breast Histopathology* é um *dataset* que já incide diretamente no contexto do projeto a realizar, isto é, o processamento e a classificação de imagens biomédicas.

O *Invasive Ductal Carcinoma* (IDC) representa um dos subtipos de cancro da mama, mais comuns nos pacientes[1]. Numa fase inicial, cabe aos médicos efetuar uma análise da amostra, de modo a que seja possível identificar zonas saudáveis e/ou zonas compostas por tumor invasivo e não invasivo[1].

As amostras são representadas em imagens, que resultaram de um processo de *scanning* a tecidos dispostos em *glace slides*[1].

Este *dataset,* é composto por milhares de imagens (277524), inerentes a 280 pacientes. Destas 277524 imagens, 198738 representam tecidos saudáveis, enquanto que as restantes 78786 imagens descrevem tecidos com presença de tumor.

O objetivo do *dataset* é classificar as amostras dos pacientes, de acordo com dois *targets* possíveis (problema binário), amostra saudável ou presença de tumor.

Seguidamente, irá ser efetuada uma breve análise ao *dataset*. De modo, a que seja mais fácil entender o problema.

Análise Exploratória do Problema:

Com o objetivo de aumentar o conhecimento sobre os dados, foi efetuada uma breve análise exploratória. Esta análise tem como objetivo central ajudar o autor a entender os dados, a identificar condições que são necessárias ter em consideração na modelação e processamento dos dados e análise de métricas.

Dessa forma, foi criado um *script* em *Jupyter*, permitindo dessa forma efetuar uma clara separação dos *scripts* inerentes aos modelos, e à análise exploratória. Por outro lado, a utilização do *Jupyter Notebook* na visualização e estudo inicial dos dados torna-se mais simples e intuitiva.

Previamente, à demonstração da análise realizada é necessário esclarecer e demonstrar a estrutura que fora considerada, no armazenamento das imagens inerentes ao *dataset*. Visto que, é necessário conhecer a estrutura considerada, de modo a entender as tarefas subsequentes.

A Figura 1 descreve um excerto da arquitetura do Projeto, e dos seus respetivos ficheiros. Devido ao enorme volume de dados, 277000 ficheiros, foram demonstrados apenas amostras relativas a um paciente, sendo exibidas duas amostras referentes às duas possíveis classes do problema.

```
##Description tree structure of images directory
def rootTree(rootDir):
    list_dirs = os.walk(rootDir)
    for root, dirs, files in list_dirs:
        for i, d in zip(range(2), dirs): #ONLY RUNS TWO TIMES
            print(d)
        for j, f in zip(range(2), files):
            print(f)
```

```
rootTree("../breast_histopathology/input/")
```

```
breast-histopathology-images
IDC_regular_ps50_idx5
10253
10254
0
1
10253_idx5_x1001_y1001_class0.png
10253_idx5_x1001_y1051_class0.png
10253_idx5_x501_y351_class1.png
10253_idx5_x501_y401_class1.png
```

Figura 1 –Exemplo da estrutura que agrega as amostras do dataset

A raíz do projeto é definida através da pasta *breast_histopathology*. O projeto agrega uma outra pasta que tem como principal objetivo efetuar uma separação entre os dados e o código, pasta *input*. As pastas que se seguem na *tree*, correspondem às pastas resultantes da extração das amostras do problema. Essas pastas são: *breast-histopathology-images,* que incorpora uma única pasta no seu interior: *IDC_regular_ps50_idx5*.

Esta pasta fornece acesso às amostras relativas a cada um dos pacientes analisados. Ou seja, reúne uma pasta para cada um dos pacientes analisados. Estas pastas são identificadas através do *id* do paciente, por exemplo e considerando a Figura 1, as pastas 10253 e 10254 correspondem às pastas que contêm as amostras desse paciente específico.

Estas pastas (identificador do paciente), contêm duas pastas no seu interior. Uma pasta que contêm as amostras do paciente que são saudáveis, representadas através da pasta "0", e ainda a pasta: "1", que contêm amostras classificadas com presença de tumor (*Invasive Ductal Carcinoma*).

Por último, ambas as pastas contêm amostras do paciente, sendo estas representadas através de imagens. A pasta "0" contêm apenas imagens da classe 0 – saudável, sendo estas imagens representadas da seguinte forma: *idPacient_*******_class0.png*. Da mesma forma, a pasta "1" contêm apenas amostras – com presença de tumor, onde a única diferença na identificação da imagem é o valor da *classe*: *idPacient_*******_class1.png*.

Posteriormente, foi efetuada uma análise aos dados do problema, mais concretamente ao nº de pacientes, e ainda ao nº de amostras existentes, e à sua distribuição o longo das duas classes do problema.

Através da visualização da Figura 2, contatamos que o nº de pacientes do problema é de 279. Ou seja, o nº de pacientes é reduzido, e mediante o objetivo do estudo do *dataset*, este valor pode revelar-se reduzido. Mas, como o objetivo do estudo do autor prende-se com a classificação binária das amostras do problema, é um fator mais importante o nº de amostras existentes.

```
#HOW MANY PATIENTS ARE IN DATASET'S --> ID FOLDERS LIKE 10253
numberPatients = os.listdir("../breast_histopathology/input/breast-histopathology-images/IDC_regular_ps50_
idx5/")
print(len(numberPatients))
```

```
279
```

Figura 2 - Nº de pacientes do dataset

Sendo assim, procedeu-se à averiguação do nº de amostras existentes no *dataset.* O *BarPlot* ilustrado através da Figura 3 demonstra os resultados obtidos.

O nº total de amostras do problema é muito elevado, existindo 277524 imagens. Contudo, recorrendo à observação da Figura 3 é possível identificar uma limitação do problema, isto é, o nº de amostras das duas classes não são balanceadas. O nº de amostras saudáveis (classe 0) é bem superior ao nº de amostras inerentes à classe 1. Sendo o nº de amostras da classe 0 igual a 198738, enquanto que o nº de amostras da classe 1 fica-se pelas 78786 imagens.

Posteriormente, é necessário identificar uma abordagem que permita efetuar um correto treino, validação e teste dos modelos, tendo em conta a diferença que existe entre o nº de amostras, de ambas as classes. Visto que, é crucial garantir que os modelos aprendem e generalizam os dados da melhor forma, considerando o maior nº de situações possíveis, evitando assim que os modelos apresentem resultados tendenciosos.

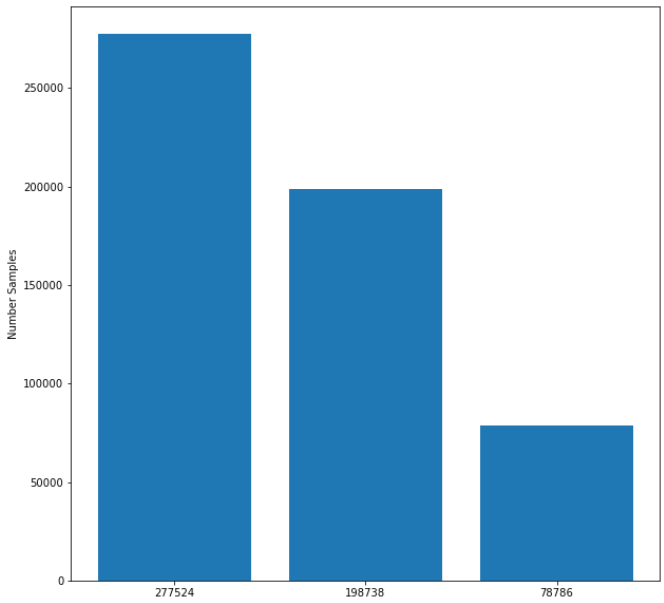

Figura 3 – Bar Plot – distribuição das amostras por classe: 1º Plot: nº total de amostras, 2ºPlot: amostras relativas à classe 0 (saudável), 3ºPlot: amostras classe 1 - Invasive Ductal Carcinoma

De modo a realizar uma análise mais incisiva, foi criado um objeto *DataFrame* com o objetivo de permitir a exploração dos dados do *dataset,* num contexto mais simples e rápido. A sua criação também irá ser importante, na transformação dos dados e respetivas classes em *numpy arrays,* para que depois se possa dar início ao processamento e tratamento dos dados, e posteriormente à criação de modelos.

O *DataFrame* criado contém todas as amostras existentes no *dataset*, ou seja, tal como já fora ilustrado através da Figura 3, 277524 imagens. Na sua identificação (identificação das imagens) foram considerados três atributos: *id* do paciente, caminho da imagem e ainda a classe à qual pertence. A Figura 4 descreve o *shape* do *DataFrame.* Este é constituído por 277524 linhas, isto é, o nº de imagens existentes, e cada linha é composta por três colunas: *id* do paciente, caminho da imagem e ainda o *target*.

```
#CHECK SHAPE OF DATA
data.shape
```

```
(277524, 3)
```

Figura 4 - Shape do DataFrame

Já a Figura 5 ilustra as cinco primeiras linhas do *DataFrame*.

	id	image_path	target
0	10253	../breast_histopathology/input/breast-histopat...	0
1	10253	../breast_histopathology/input/breast-histopat...	0
2	10253	../breast_histopathology/input/breast-histopat...	0
3	10253	../breast_histopathology/input/breast-histopat...	0
4	10253	../breast_histopathology/input/breast-histopat...	0

Figura 5 - Exemplo do DataFrame obtido (5 linhas)

Antes de continuar a análise ao *dataset,* foi efetuada uma verificação ao *DataFrame*, isto é, o autor verificou se existiam dados em falta, que podiam ter ocorrido no ato de criação do *DataFrame*. A Figura 6 ilustra a informação genérica sobre o *DataFrame*. É possível constatar que não existem valores inválidos, para cada uma das suas colunas.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 277524 entries, 0 to 277523
Data columns (total 3 columns):
 #   Column      Non-Null Count   Dtype
---  ------      --------------   -----
 0   id          277524 non-null  object
 1   image_path  277524 non-null  object
 2   target      277524 non-null  object
dtypes: object(3)
memory usage: 8.5+ MB
```

Figura 6 - Informação sobre o DataFrame

Depois de criado o *DataFrame,* torna-se mais simples a análise e manipulação dos dados. E dessa forma, procedeu-se à análise e compreensão dos dados do problema.

Da análise realizada, foi possível identificar que o nº de amostras de cada paciente, são muito dispares entre si. Ou seja, existem pacientes que contêm um nº de amostras na ordem dos milhares, já outros pacientes possuem números bem mais simbólicos, como poucas centenas de imagens. A Figura 7 descreve o nº de *patches* referentes a cada um dos 279 pacientes do problema. A 1ª coluna representa o *id* do paciente, já a 2ª coluna ilustra o seu respetivo nº de amostras.

```
id
8863      979
8864     1133
8865      712
8867     1642
8913      955
          ...
16568     828
16569     337
16570     917
16895     151
16896    1127
```

Figura 7 - Nº de patches por paciente

No âmbito do problema a resolver, esta disparidade de amostras que existe entre os pacientes do problema, não se revela prejudicial.

Outro aspeto complementar que fora evidenciado, prende-se com o diferente nº de amostras, entre ambas as classes, para os vários pacientes do *dataset*. Isto é, como já fora mencionado atrás a classe 0 (“saudável”) encontra-se mais representada que a classe 1 (“presença de tumor”), contudo foi possível identificar que existem realidades muito distintas entre os vários pacientes do problema. Ou seja, existem pacientes em que mais de 70% das suas amostras são amostras pertencentes à classe 1, por outro lado existem pacientes em que apenas 20% das suas amostras representam amostras da classe 1. Sendo que, perto de 40% dos pacientes reúne um conjunto de amostras inferior a 20% da classe 1.

Este facto pode estar diretamente relacionado com o estado de saúde dos pacientes, ou seja, pacientes com tumor mais avançado reúnem um maior conjunto de amostras com *IDC*, já pacientes com menor incidência, reúnem um menor conjunto de amostras com *IDC*. Contudo,

esta reflexão revela-se apenas uma “teoria do autor”, sendo que a justificação para este facto pode não estar relacionado com isso. Este facto também não promove quaisquer implicações para o estudo a realizar.

As Figuras 8 e 9 demonstram respetivamente a percentagem de amostras referentes à classe 1, para cada um dos 279 pacientes do problema, enquanto que a Figura 9 ilustra o nº de pacientes, que contêm um nº de amostras da classe 1 inferior a 20%, ou superior a 80%.

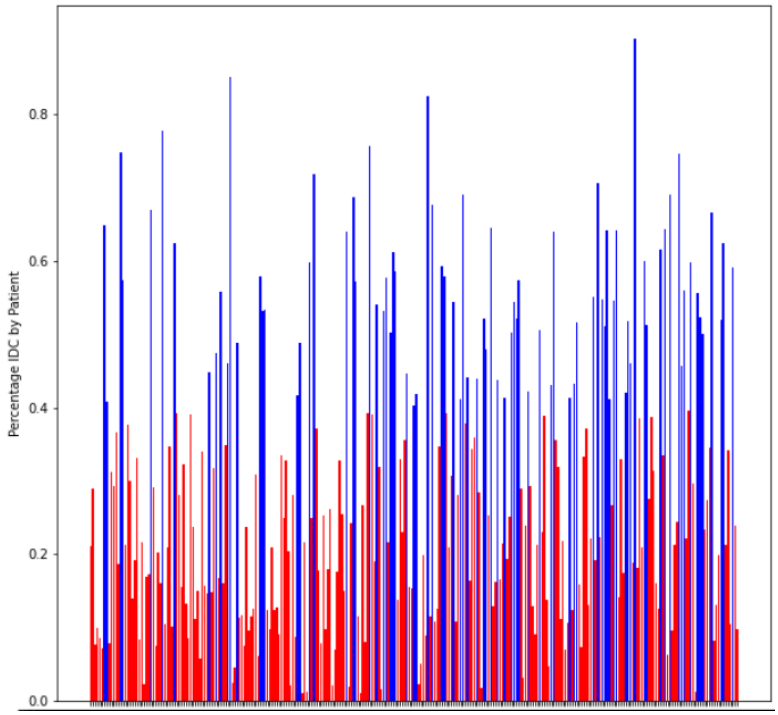

Figura 8 - Percentagem de amostras da classe 1, por paciente - Vermelho inferior a 0,4 , Azul superior a 0.4

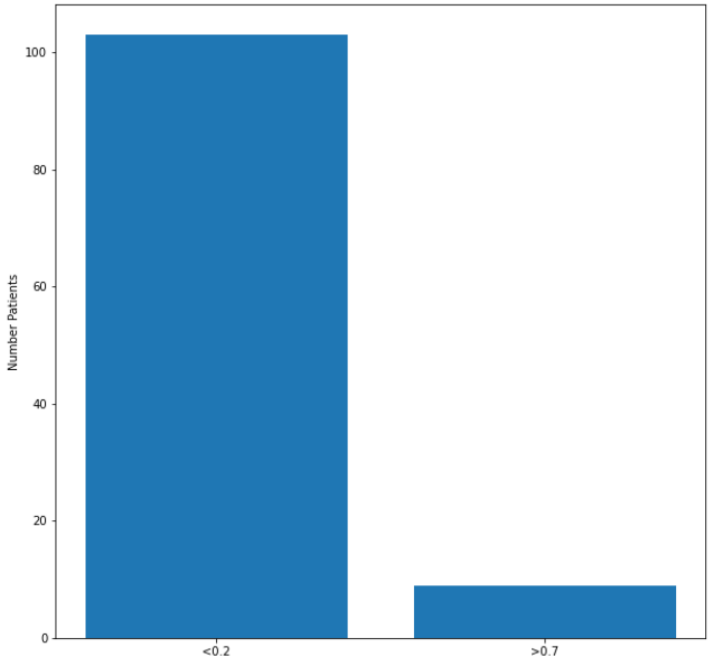

Figura 9 - Nº de pacientes com um nº de amostras da classe 1 inferior a 0.2 e superior a 0.8

Finalmente, foi efetuada uma pequena análise ao "ponto" mais importante do estudo a realizar: as imagens.

As imagens presentes no *dataset* estão representadas por uma altura e largura de 50 pixeis. Para além disso são imagens RGB, ou seja, compostas por três *channels*. A Figura 10 descreve uma das imagens do *dataset*.

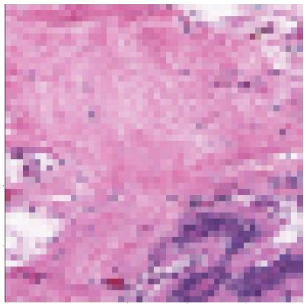

Figura 10 - Exemplo de uma imagem

O *Invasive Ductal Carcinoma (IDC)* é um tipo de cancro que começa a ser desenvolvido através dos *milk ducts*. Estes "caminhos" são responsáveis por transportar o leite que é produzido nos lóbulos, até ao *nipple*[2].

À medida que o cancro se vai desenvolvendo, o mesmo vai se "arrastando" para outras zonas da mama, mais concretamente os seus tecidos e lóbulos[2]. A Figura 11 ilustra uma representação alusiva do interior de uma mama (sendo aqui considerado apenas, as componentes essenciais ao entendimento do problema).

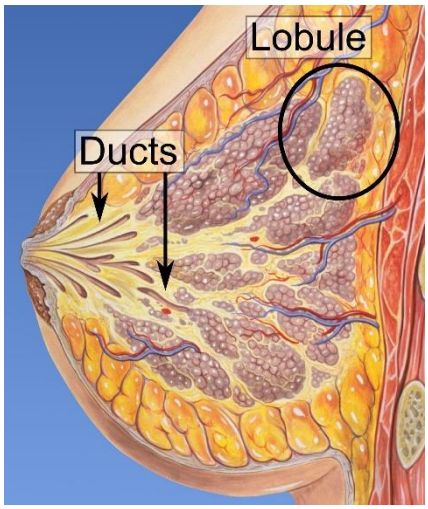

Figura 11 - Milk ducts e Lóbulos – Fonte: https://commons.wikimedia.org/wiki/File:Lobules_and_ducts_of_the_breast.jpg

Na Figura 12 podemos visualizar a diferença entre um *Normal Duct* e um *Invasive Ductal Carcinoma*, bem como a sua progressão. Através da sua visualização é possível constatar que um *Normal Duct* toda a componente de *epithelium* mantem-se inalterada. Já num IDC, a quantidade de *epithelium* vai se alterando, acabando por transpor o *Myopithelium*.

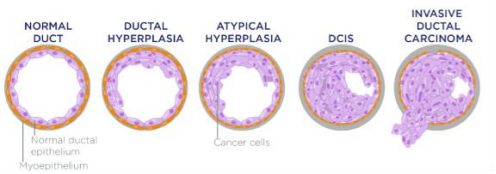

Figura 12 – Diferença entre Normal Duct e IDC

De modo, a verificar a complexidade que existe na classificação das imagens do *dataset,* foi efetuada uma análise de algumas imagens referentes à classe 0 e ainda a outras imagens referentes à classe 1. As Figuras 13 e 14 demonstram respetivamente imagens inerentes à classe 0 e ainda à classe 1.

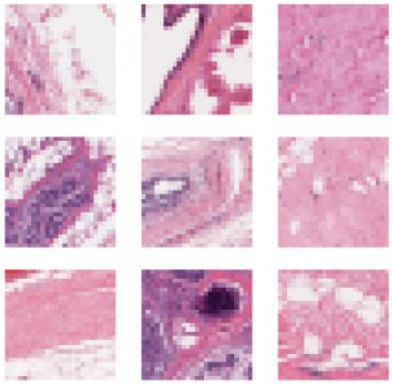

Figura 13 - Exemplo amostras Classe 0 ("saudáveis")

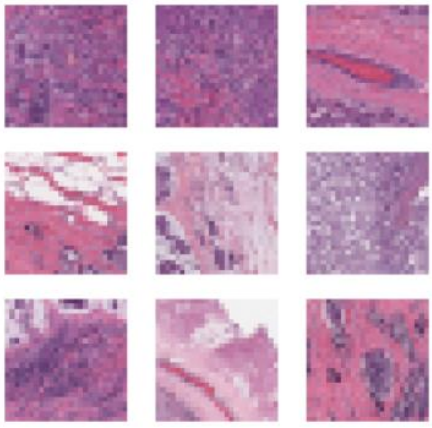

Figura 14 - Exemplo amostras Classe 1 ("IDC")

Observando ambas as Figuras, é notória a dificuldade que existe em identificar um padrão que permita classificar corretamente cada uma das imagens expressas. Essa dificuldade é tão notória que um estudo realizado [4], demonstra a efetividade que os pássaros têm na classificação correta destas imagens, superando mesmo os humanos.

O mesmo estudo [4], revela ainda que os passáros tal como os humanos revelam ainda mais dificuldades, na classificação das imagens, quando existe uma ausência de cor nas imagens, ou quando considerados ângulos distintos[4].

Apesar da dificuldade na classificação das imagens, é possível identificar uma maior presença de cores “vivas” nas imagens relativas à Classe 1, devido sobretudo ao facto do *epithelium* se arrastar ao longo do interior do *Milk Duct*.

Técnicas de pré-processamento dos dados aplicadas:

Posteriormente, e após o estudo e compreensão do *dataset*, seguiu-se a realização de um conjunto de tarefas, que têm como principal objetivo garantir que o *dataset,* reúne as melhores condições possíveis para que um modelo consiga prever corretamente, um determinado conjunto de amostras.

Antes, de dar início à fase de pré-processamento, foi necessário realizar um pequeno conjunto de tarefas, de modo a que fosse possível reunir um conjunto de dados (dados de treino, validação e de teste), que fosse "aceite", durante a execução dos modelos a serem aplicados posteriormente.

Até então, os dados estavam representados em formato *DataFrame* (objeto relativo à biblioteca *Pandas)*, mas de modo a que fosse possível treinar e prever o *ouput* de um modelo, era necessário transformar os dados, num objeto *numpy array* (caso estivesse a utilizar a biblioteca *Tensorflow,* poderia utilizar um objeto *Tensor*, mas está a ser utilizada a API *Keras*). Na realização desta tarefa, o autor contou com a ajuda da biblioteca *opencv*. Esta biblioteca ajuda a transformar uma imagem declarada num formato específico como: ".png" ou .jpeg", num objeto *numpy array.* Para além disso, permite ainda efetuar o *reshape* da imagem, num determinado *output* indicado pelo utilizador, neste caso: (*width, height, channels*).

O próximo passo a realizar prende-se com a separação do conjunto de dados, em três *datasets*: de treino, validação e ainda de teste. Neste momento, foi considerada apenas a utilização de 10000 imagens, ao invés das 277000 imagens presentes no *dataset,* de modo a reduzir o poder computacional exigido nas tarefas a serem desenvolvidas. Contudo o nº de amostras a utilizar é definido pelo utilizador, existindo assim flexibilidade total para definir o nº de amostras pretendidas pelo mesmo. No processo de separação, do nº de amostras pelos três *datasets*, foi considerada a seguinte abordagem: 60%-20%-20%, respetivamente para o *dataset* de treino, validação e teste.

Finalmente, foi ainda aplicado o conceito de *one-hot-encoding,* aos *targets,* relativos aos três conjuntos de dados criados atrás.

O conjunto de dados inicial já se encontra devidamente separado, pelos três conjuntos a utilizar nas fases de treino/validação e ainda na fase de *predict*. Os *targets* inerentes a cada *dataset* criado, também já se encontram em formato binário. Com naturalidade, segue-se a fase de pré-processamento dos dados.

A fase de pré-processamento, é normalmente dividida em "três tarefas principais": Limpeza de dados, Transformação e ainda Seleção. Seguidamente, irá ser abordado o que fora efetuado, e não efetuado tem em conta o projeto em análise.

A tarefa de Limpeza de Dados é normalmente dividida em duas tarefas, sendo estas: tratamento de *missing values* e ainda identificação de possíveis *outliers*. Considerando, o problema em análise e ainda a Figura 6 é possível constatar que o problema, não reúne dados em falta. Não existindo assim a necessidade de realizar qualquer tipo de operação, para responder a este problema.

Já, ao nível da identificação de *outliers,* não foi considerada a aplicação de qualquer técnica para dar resposta a este problema, visto que o autor não reúne conhecimentos na área, que lhe permita identificar com certeza, se eventualmente existem *patches*, que possam ser ruidosos. Para além disso, e perante o tipo de problema em causa (problema inserido no âmbito

hospitalar), é expectável que o *dataset* represente um nível de qualidade máximo. Ainda assim, o autor recorreu à utilização de *box plots*.

Seguidamente, procedeu-se à análise e implementação da técnica de transformação dos dados do problema. A aplicação de técnicas de transformação dos dados revela-se vital na maioria dos problemas.

Revela-se fulcral a utilização deste tipo de técnicas, de modo a que seja possível reduzir as diferenças na escala, que existem entre os valores presentes no *dataset*. Este tipo de abordagem ajuda a minimizar a importância que uma determinada *feature,* teria em relação à outra, dado que é considerada a mesma escala, para cada *feature*.

Considerando, este tipo de problema em específico (que apesar de apresentar escalas iguais para cada um dos *channels,* entre 0 e 255, pode eventualmente existir uma variação distinta entre si), e ainda o tipo de modelo a utilizar: rede CNN (neste tipo de redes existe uma constante partilha de parâmetros, sendo importante que exista uma escala uniforme para os dados), é notória a importância que a transformação dos dados apresenta.

Existem duas técnicas, que são consideradas *standard,* no que toca à transformação dos dados: a standardização e ainda a normalização. Ambas as técnicas visam tornar o processo de convergência mais rápido, no treino de um determinado modelo. Contudo, existem algumas diferenças entre si, nomeadamente no seu processo de cálculo. A normalização – técnica *min-max*, revela-se mais simples, estabelecendo uma escala de [0-1], para todos os pontos inerentes ao conjunto de dados. Sendo considerada a diferença entre um ponto e o valor mínimo no *dataset,* sendo depois este valor dividido pela diferença entre o valor máximo e mínimo no conjunto de dados. A Fórmula 1 exibe esta técnica.

$$X_{new} = \frac{X - X_{min}}{X_{max} - X_{min}} \quad (1)$$

Por outro lado, a standardização, técnica *z-score*, considera a transformação dos dados, para que estes sigam uma distribuição normal. Ou seja, o objetivo assenta na transformação dos dados, de modo a que estes sigam uma distribuição normal, ou seja, tenham uma $\mu = 0$ e um $\sigma = 1$. A Fórmula 2 ilustra o cálculo considerado para o efeito.

$$X_{new} = \frac{X - \mu}{\sigma} \quad (2)$$

Para este problema, foi considerada a aplicação da técnica *z-score*. A computação da média e do desvio padrão foi aplicada ao longo dos três *channels* separadamente, no conjunto de treino. Sendo que, a média e desvio padrão obtidos no conjunto de treino, foram consideradas na computação do *dataset* de validação e ainda de treino, de modo a evitar o problema de *data leakage*.

A última técnica de pré-processamento a executar é a técnica de Redução da dimensionalidade do *dataset,* ou também conhecida por *Feature Selection*. Quando considerando um típico problema de classificação de imagens, uma das abordagens mais consensuais na redução das dimensões do problema, assenta na técnica de "redimensionamento" das imagens. O utilizador diminui ou aumenta a resolução de um determinado conjunto de imagens. Geralmente, recorre-se a técnicas de *interpolation* como por exemplo: *Nearest-Neighbor Interpolation, Linear Interpolation, Fourier Interpolation,* ou ainda *Cubic Interpolation*. Estas técnicas permitem calcular valores entre pontos das amostras, por exemplo, quando se pretende aumentar o

tamanho de uma imagem, estas técnicas calculam a "nova informação" a inserir nos novos pixéis adicionados à imagem.

Antes de proceder ao redireccionamento das imagens, é necessário ter em consideração que a sua aplicação, reduz a qualidade das imagens, o que por sua vez pode ter impacto no processo de previsão dos modelos.

Dessa forma, e considerando o tamanho das imagens do *dataset* (50*50), não foi aplicado qualquer tipo de redireccionamento às imagens, isto é, aumento do seu tamanho ou decréscimo, visto que, este valor ainda é comportável. Contudo, e caso os resultados obtidos sejam satisfatórios considerando o tamanho atual das imagens, posteriormente pode ser aplicada uma técnica de redução do tamanho das imagens, com o intuito de reduzir os custos computacionais exigidos, e obter da mesma forma resultados satisfatórios.

Outra técnica usual para reduzir a dimensionalidade, passa pelo encapsulamento dos três canais RGB, num único canal *grayscale*. Tal, como as restantes técnicas, ajuda a reduzir o poder computacional, mas reduz a qualidade das imagens, e dificulta ainda mais o processo de aprendizagem.

Arquitetura:

Depois de aplicadas as várias técnicas de pré-processamento, segue-se a criação de modelos, que visam a classificação das imagens do problema.

De modo a aumentar a flexibilidade, escalabilidade e usabilidade do código, foi criado um diagrama de classes que fornece uma visa geral do comportamento do "sistema" criado, para dar resposta à resolução do problema em estudo. Para além disso, o diagrama de classes ajuda a definir e a perceber, com maior clareza, quais os padrões "de arquitetura", que melhor se adequam, e que devem ser considerados na definição da arquitetura do sistema.

Foi despendido algum tempo na criação e implementação desta arquitetura, com o objetivo de criar uma arquitetura que permitisse, a criação, treino, previsão e otimização de qualquer *tipo* de *image dataset*. Para tal, foi considerada a utilização de vários padrões arquiteturais, tais como: Fábricas de Objectos, Estratégia, Template ou Abstração.

O desenho e a modelação das interações estabelecidas entre os vários objetos criados, estabelece uma visão mais clara do comportamento do sistema, e ajuda ainda a otimizar gradualmente a arquitetura criada.

A Figura 15 ilustra o Diagrama de Classes criado. **De realçar, que a ilustração dos métodos e atributos das classes, por vezes contêm pequenas abreviações e/ou tipos de dados ou retornos "corretos", dado que a ferramenta utilizada contêm algumas limitações nesse âmbito. Ou seja, a análise do diagrama, deve ser complementada com a visualização do código**.

Foi considerada a criação de um objeto, que têm como objetivo a agregação dos vários conjuntos de dados, que irão ser utilizados, ao longo da execução do *lifecycle* de um modelo, como: treino e previsão.

Desta forma, e observando a classe *Dataset* é possível evidenciar isso mesmo, isto é, este objeto irá agregar os vários conjuntos de dados a utilizar, durante o processo de treino e previsão de um modelo. Isto é, os *datasets* obtidos na fase de pré-processamento: dados de treino, validação e ainda de teste. Esta abordagem aumenta essencialmente a escalabilidade e a

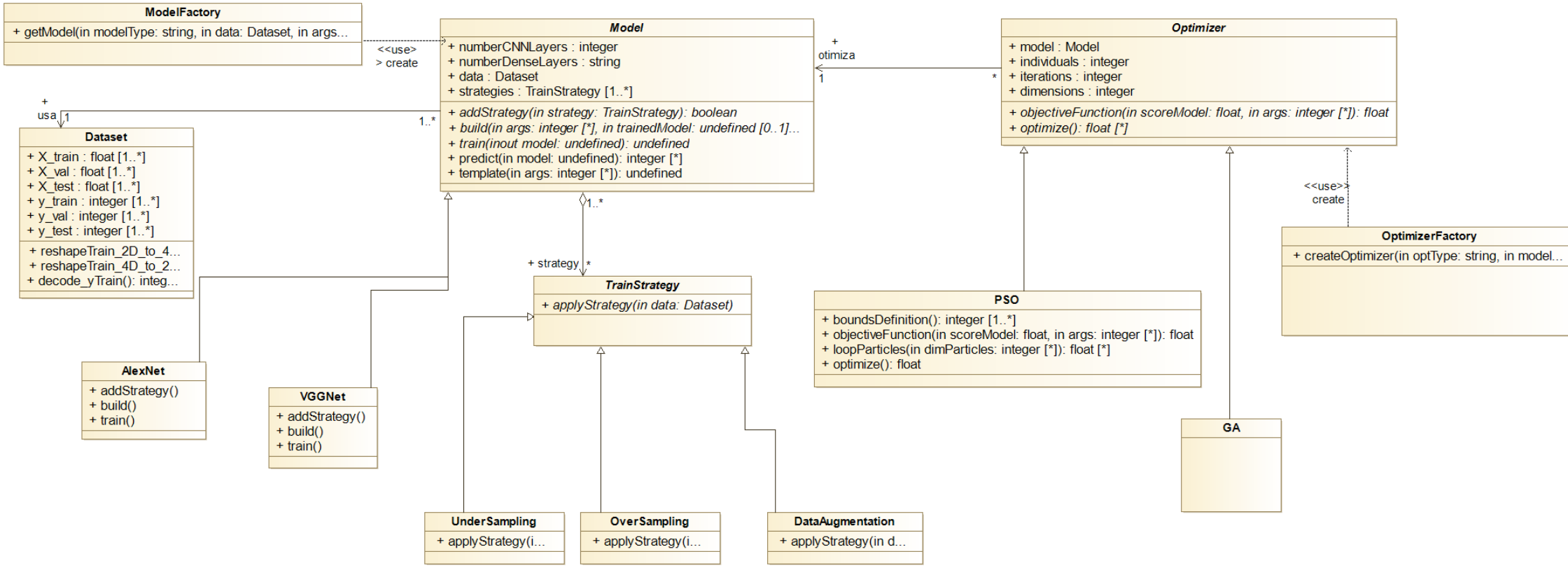

Figura 15 – Diagrama de Classes do "Sistema"

reutilização do código, ajudando o utilizador a não repetir código e a aumentar a legibilidade do código (por exemplo: evitar a criação de *templates* de uma função extensos). Para além disso, esta classe armazena funções, que aplicam funcionalidades que estão diretamente relacionadas com os dados armazenados, como por exemplo: *reshape* de um determinado conjunto de dados.

As vantagens obtidas através da criação deste objeto são facilmente reconhecíveis. Um exemplo disso, remete à *classe Model*, isto é, quando necessita de aceder a um determinado conjunto de dados (ao ter associado um objeto *Dataset*), permite-lhe melhorar a flexibilidade do código e ainda a robustez.

Posteriormente, foi criada a classe abstrata *Model*. Esta classe, tem como principal objetivo descrever uma generalização, de vários modelos que podem ser aplicados neste *dataset*, tais como: *AlexNet, VGGNet, ResNet, Inception*, entre outros. Ou seja, recorrendo ao conceito de abstração desta classe existe assim uma maior flexibilidade, para o utilizador definir extratos de código específicos, a cada "variante" da classe *Model*, como por exemplo: a construção dos modelos, onde cada tipo de rede apresenta a sua própria arquitetura, ou ainda o modo de treino a aplicar em cada tipo de rede, que pode ser distinto.

Relativamente, aos atributos desta classe são essencialmente: o nº de camadas CNN, que o modelo têm, e ainda o nº de *Dense Layers*. Para além disso, têm associado a si um objeto *Data* reunindo os vários conjuntos de dados (treino, validação e teste), que serão utilizados para treinar os modelos e prever resultados. Agrega ainda um conjunto de estratégias, que são utilizadas, no processo de treino de um modelo.

Já ao nível dos seus métodos, esta classe agrega métodos abstratos e ainda métodos comuns a todas as suas generalizações. Ou seja, os métodos abstratos constituem operações, que representam variações de implementação, tendo em consideração o tipo de objeto criado. Isto é, os métodos: *addStrategy, build* e *train*, são métodos em que a sua implementação varia mediante o tipo de modelo considerado, por exemplo a construção de um modelo *AlexNet* é distinta da criação de um modelo *VGGNet*. Por sua vez, o processo de treino de diferentes tipos de modelos, também pode apresentar variações.

Já os restantes métodos constituem implementações, que são comuns a todas as generalizações. Exemplo disso, é o método de avaliação dos modelos, que é comum a todos eles. Não existindo assim necessidade de implementar diferentes abordagens, para os vários tipos de modelos considerados. Ainda assim, e caso haja necessidade de distinguir a abordagem utilizada na avaliação dos vários tipos de modelos, este método é facilmente "convertido" para abstrato, e de seguida pode-se implementar para cada classe a devida abordagem considerada (ou efetuar *pass,* da implementação abstrata).

Recorre-se ainda à implementação de um *template method,* que têm como objetivo estabelecer uma espécie de *pipeline*, que permita ao utilizador executar todo o *lifecycle* de um modelo, de uma forma contínua. Ademais, permite ainda que todas as generalizações da classe *Model*, sigam a mesma diretriz definida, para a execução correta do modelo, pois o "esqueleto" do algoritmo está definido na sua classe "pai".

Neste método, *template_method*, foi considerada a *pipeline* típica de criação, treino e previsão de um modelo. Ou seja, inicialmente é construída a arquitetura do modelo, depois segue-se o treino do modelo (considerando as várias estratégias definidas para o mesmo), e por último recorre-se à sua avaliação (*predict*).

De modo a desacoplar o código do cliente, sobre as classes concretas do objeto *Model*, foi criada uma Fábrica de Objetos, não expondo dessa forma a lógica de implementação de uma classe. Para além disso, aumenta a reutilização de código, e facilita a escalabilidade do sistema.

Como já fora indicado anteriormente, um *Model* agrega um conjunto de estratégias de Treino no seu interior. Estas estratégias representam diferentes abordagens, que podem ser utilizadas por um modelo, durante o seu processo de treino.

Esta abordagem foi criada recorrendo ao conceito do padrão Estratégia. Para tal, foi considerada a criação de uma *interface* que estabelece as operações (funcionalidades), que as suas implementações realizam. Este padrão revela diversas vantagens, como: o facto do comportamento de uma classe poder ser alterado em *runtime*, ou a facilidade de alterar o código de uma *concrete strategy,* sem necessidade de alterar mais nenhuma classe.

A classe *TrainStrategy* descreve a interface que é implementada, pelas várias estratégias concretas: *Undersampling, Oversampling* e *Data Augmentation*. Esta classe define várias ações, que são depois implementadas pelas classes concretas, que implementam a *interface*. Neste caso, existe uma única ação: *applyStrategy*.

Esta ação é depois implementada, em cada uma das estratégias consideradas (vários tipos de estratégias). Tal como já fora indicado previamente, estas estratégias representam abordagens, que podem ou não ser consideradas durante a fase de treino de um modelo. As estratégias consideradas: *Undersampling, Oversampling* e *Data Augmentation*, descrevem ações que reúnem abordagens de implementação distintas.

Um modelo agrega um conjunto de estratégias, sendo que o seu comportamento é variável, tendo em consideração o tipo de estratégias que utiliza. Ou seja, caso o modelo não utilize estratégia nenhuma, então o processo de treino ocorre normalmente, caso utilize por exemplo: a estratégia *UnderSampling*, então o treino do modelo vai variar, ou por exemplo caso utilize duas estratégias ainda varia mais.

Este tipo de abordagem flexibiliza o código e a interpretabilidade do mesmo. Contudo, apresenta algumas limitações: como o aumento do nº de classes, ou o nº de objetos que é necessário manter em *runtime*.

Por último, segue-se a explicação da abordagem considerada, na implementação dos algoritmos de otimização, que são utilizados no processo de otimização dos modelos criados.

Para tal, foi considerada a utilização do conceito de abstração. Sendo criada a classe abstrata *Optimizer*, que reúne um conjunto de atributos que são "partilhados", por qualquer tipo de algoritmo de otimização: nº de partículas/população, nº total de iterações e ainda o nº de dimensões do problema. Para além disso, é lhe ainda associado o modelo que irá otimizar.

Neste caso, em concreto foi considerada a utilização de apenas dois algoritmos de otimização: *Particle Swarm Optimization* (*PSO*) e *Genetic Algorithm* (*GA*). De realçar que, o algoritmo *GA* ainda não foi implementado, e dessa forma, ainda não foi especificado o "seu conteúdo" no diagrama de classes ilustrado (Figura 16).

A classe abstrata *Optimizer,* contêm dois métodos abstratos, que também são variáveis mediante o tipo de algoritmo a utilizar: a definição da função objetivo (*objectiveFunction*) e ainda a implementação da funcionalidade de otimização do modelo (*optimize*). Ou seja, o PSO descreve uma abordagem de otimização distinta do GA, e de outros algoritmos de otimização,

e para além disso cada algoritmo de otimização, pode ou não, descrever funções objetivo, distintas entre si. Dessa forma, podemos constatar observando a classe PSO, que a mesma implementa ambos os métodos abstratos descritos na classe *Opimizer*, e para além disso contém ainda outros métodos concretos, que são apenas alusivos à sua classe, isto é, são métodos necessários para complementar determinadas funcionalidades específicas do *PSO*.

Finalmente, e tal como aplicado no caso da classe *Model,* foi criada uma fábrica de objetos, que ajuda na criação dos optimizadores a utilizar. Melhorando assim, a flexibilidade e a reutilização do código.

Análise de Resultados:

Depois de efetuado um estudo breve ao *dataset*, que permitiu melhorar a compreensão do problema, passando pela definição/aplicação das várias técnicas de pré-processamento e terminando na definição de todo o código que sustenta a criação, treino, previsão e otimização dos modelos, segue-se a fase de análise de resultados.

Nesta fase, o objetivo assenta no teste de várias condicionantes, e na análise dos resultados obtidos, quando aplicados cenários distintos. O objetivo é sempre o mesmo, a tentativa de garantir modelos capazes, eficientes, com menor custo possível, e que garantam bons resultados finais.

Dessa forma, irá ser considerada a aplicação de mais do que um tipo de rede, considerando diferentes volumes de dados (quantidades de amostras), análises dos resultados considerando a utilização de redes simples e complexas, utilização de diferentes estratégias de treino dos modelos, e o impacto que proporciona aos dados, entre outros.

Seguidamente, irá ser efetuada uma análise dos resultados obtidos, considerando os vários cenários aplicados. Sendo utilizadas diversas figuras, de modo a facilitar a compreensão dos resultados, e também com o intuito de ajudar na melhoria contínua dos resultados obtidos. A análise segue uma abordagem progressiva, ou seja, existe uma análise faseada dos resultados obtidos, existindo a alteração e adição de conceitos, de uma forma incremental. Esta abordagem, ajuda a analisar o impacto dos resultados obtidos, e a melhorar os modelos e resultados obtidos.

AlexNet:

Primeiramente, irá ser efetuada uma análise de resultados, considerando a aplicação do tipo de rede *AlexNet*.

Revela-se um tipo de rede bastante interessante, devido sobretudo à sua facilidade de compreensão e sua simples arquitetura. Para além disso, representa um modelo, que não implica a utilização de muitos parâmetros (ainda assim, existem arquiteturas bem mais leves).

Adiante encontra-se sobre a forma de tópicos, os principais resultados obtidos, quando aplicados cenários de "teste" distintos.

Análise à complexidade da rede:

O primeiro teste aplicado considerou a utilização de um modelo simples, composto por um baixo nº de parâmetros e ainda por uma arquitetura baseada em poucas camadas, quer convolucionais, de *pooling* ou *Dense*. Para além disso, considerou ainda a utilização de um baixo nº de imagens, ou seja, das 277000 imagens foram apenas consideradas 5000 imagens (divididas pelos vários conjuntos de dados).

Não foi considerada a utilização de qualquer tipo de estratégia de treino: *Undersampling, Oversampling* ou *DataAugmentation*. Tal como já referido anteriormente, as classes do problema não são balanceadas. O "não tratamento deste problema" pode levar a que o modelo não consiga aprender e generalizar a classe menos representativa, demonstrando assim uma aprendizagem "uni-classe".

É expectável que os resultados obtidos apresentem uma elevada *accuracy*, visto que como não é considerada a utilização de nenhuma técnica que "atenue" o não balanceamento das classes,

existe uma enorme probabilidade do modelo, apenas generalizar a aprendizagem à classe mais representativa, neste caso a classe 0 (“saudável”). Já, as restantes métricas, espera-se que as mesmas sejam baixas, nomeadamente a *precision* e/ou *recall.* Resumindo, a expetativa é que os resultados não sejam satisfatórios, contudo este exemplo é aplicado, de modo a aferir o impacto que a utilização de uma arquitetura mais simples e/ou utilização de um menor conjunto de dados têm no processo de aprendizagem do modelo. Pois, para além da obtenção de “bons resultados”, o objetivo passa pela tentativa de minimização dos custos computacionais e de tempo exigidos na execução, treino e previsão dos modelos.

Relativamente aos resultados obtidos, é possível salientar que os mesmos foram melhores do que os expectáveis, considerando um nº baixo de *epochs*, 10 *epochs*. Sendo que, as métricas referentes a ambas as classes apresentaram “muito bons” indicadores. As Figuras 16 e 17 ilustram um resumo dos resultados obtidos. Já a variação do custo e da *accuracy* ao longo das *epochs,* também revelou bons indicadores, onde a validação conseguiu acompanhar o treino, revelando assim uma generalização adequada (no parágrafo seguinte, constatamos que este cenário, é enganador). As Figuras 18 e 19 ilustram a variação do custo e da *accuracy*, ao longo das *epochs*.

	precision	recall	f1-score	support
Healthy	0.91	0.86	0.89	532
With IDC	0.76	0.84	0.80	268
accuracy			0.85	800
macro avg	0.83	0.85	0.84	800
weighted avg	0.86	0.85	0.86	800

Figura 16 - Resultados das métricas

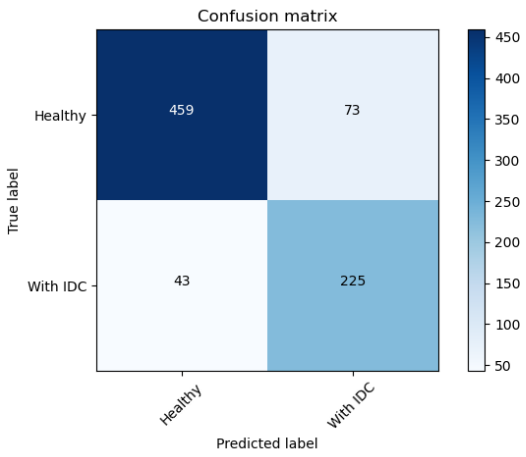

Figura 17 - Matriz Confusão

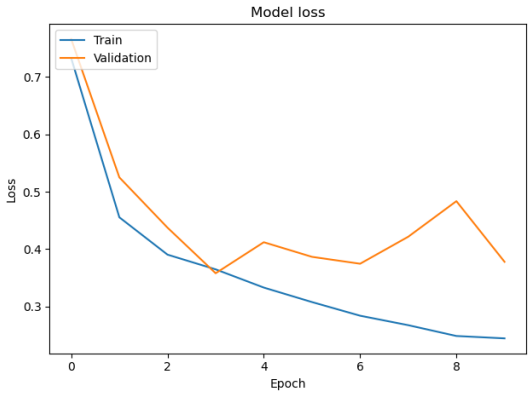

Figura 18 – Variação do custo ao longo das epochs

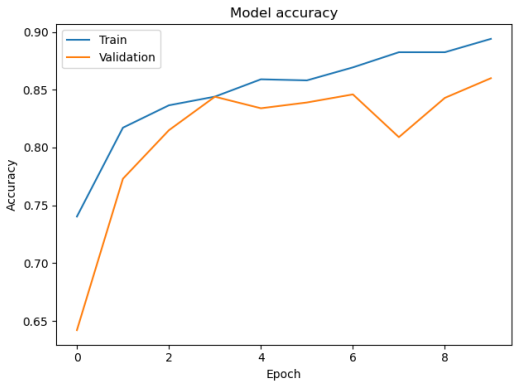

Figura 19 - Variação da accuracy ao longo das epochs

Contudo, quando considerado um maior número de *epochs*, 25 *epochs*, é possível concluir que o modelo tende a sofrer de *overfitting* muito cedo, por volta das 5 *epochs*, quando o custo e a *accuracy* da validação deixam de acompanhar o treino. Este facto leva a que o modelo não consiga generalizar corretamente os dados, mais concretamente as imagens alusivas à Classe 1. Ou seja, ao fim de um determinado nº de *epochs,* o modelo tende a generalizar todas as imagens como Classe 0. As Figuras 20 e 21 demonstram o efeito de *overfitting* ao longo das *epochs*, já as Figuras 22 e 23 demonstram a queda das métricas da Classe 1.

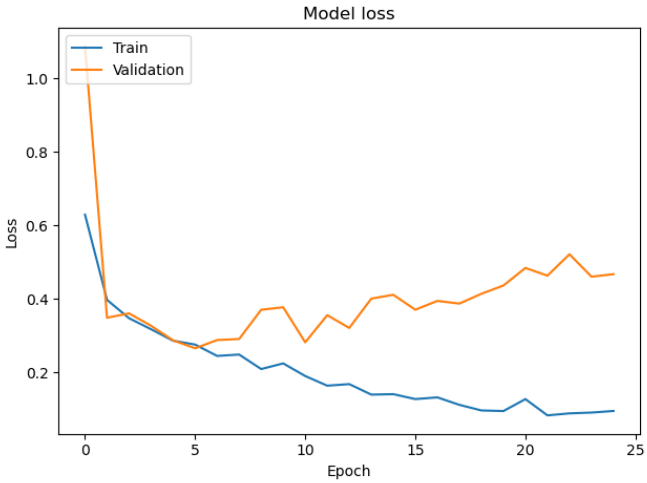

Figura 20 – Variação do custo ao longo das epochs

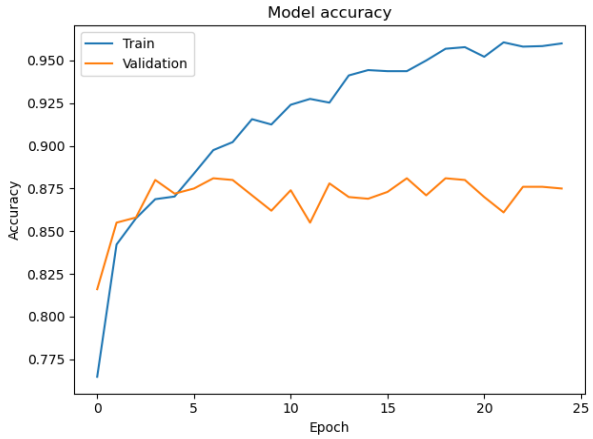

Figura 21 - Variação da accuracy ao longo das epochs

	precision	recall	f1-score	support
Healthy	0.94	0.91	0.92	682
With IDC	0.56	0.69	0.62	118
accuracy			0.87	800
macro avg	0.75	0.80	0.77	800
weighted avg	0.89	0.87	0.88	800

Figura 22 - Resultados das métricas

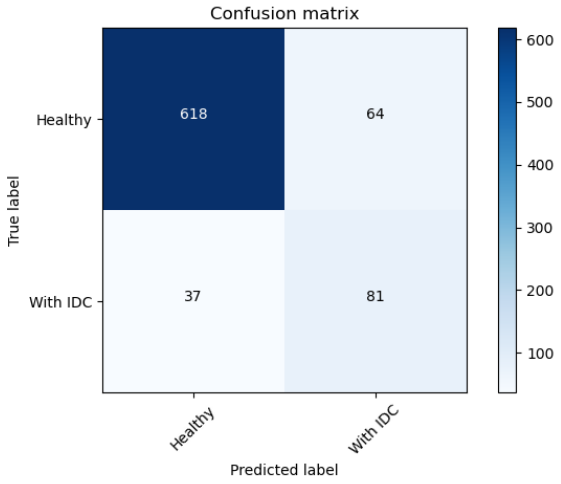

Figura 23 - Matriz confusão

São várias as consequências que podem estar relacionadas com esta queda acentuada dos valores das métricas (*overfitting),* nomeadamente o baixo nº de imagens consideradas quer de treino, quer de validação, ainda o facto do modelo ser muito pouco complexo para responder a este problema, e ainda o facto de não ter sido considerada qualquer "estratégia" para atenuar o problema das classes não serem balanceadas.

De modo, a comprovar esta tendência, foi considerado um nº de *epochs* ainda mais elevado. Sendo expectável que os resultados, sejam ainda mais baixos, do que os já apresentados, nas Figuras 22 e 23. Dessa forma, as Figuras 24 e 25 ilustram os resultados obtidos, e as Figuras 26 e 27 a variação do custo e *accuracy*, ao longo das *epochs*.

	precision	recall	f1-score	support
Healthy	0.96	0.93	0.95	709
With IDC	0.57	0.74	0.64	91
accuracy			0.91	800
macro avg	0.77	0.83	0.80	800
weighted avg	0.92	0.91	0.91	800

Figura 24 - Resultados obtidos

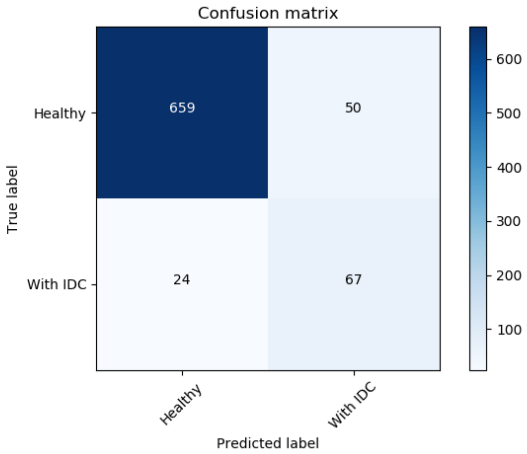

Figura 25 - Matriz confusão

Figura 26 - Variação do custo ao longo das epochs

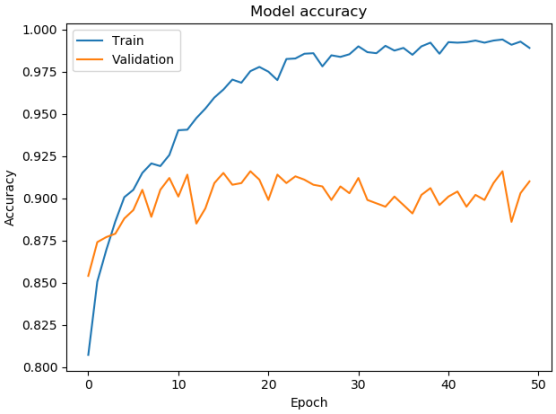

Figura 27 – Variação da accuracy ao longo das epochs

Tal, como já esperado, os resultados foram semelhantes aos obtidos atrás. Permitindo assim constatar que é necessário efetuar alterações de modo a melhorar os resultados. Essas alterações, devem passar pela melhoria do modelo (arquitetura/profundidade/hiperparâmetros), no nº de dados a considerar para treino/validação e teste, e/ou a introdução de técnicas que permitam reduzir o efeito de *overfitting/*não balanceamento das classes. Porque, o modelo atual sofre claramente de *overfitting*, visível nas através das Figuras 26 e 27, sendo necessário erradicar este problema, de modo a que seja possível a obtenção de um modelo mais capaz, permitindo assim resultados a obtenção de melhores resultados, menos tendenciosos e reunindo uma generalização adequada do modelo.

Aumento do nº de dados:

Com o objetivo de melhorar os resultados obtidos, foi considerado o aumento do nº de dados a utilizar durante o treino e previsão do *dataset*. Dessa forma, a única alteração efetuada em relação, aos exemplos anteriores foi a utilização de 50000 imagens, ao contrário das 5000.

Todas as restantes configurações do modelo, e seus hiperparâmetros mantiveram-se inalteradas. Esta abordagem foi apenas considerada, de modo a comprovar que o aumento do nº de atributos, neste problema, não resolve os problemas já identificados, visto que existem outros problemas que não foram resolvidos, como: o não tratamento das classes (inexistência de balanceamento, entre estas), e/ou verificação do impacto, que a utilização de uma arquitetura mais complexa promove.

Desta forma é expectável que os resultados obtidos sejam semelhantes aos obtidos anteriormente. Porque, o aumento do nº de imagens, mantendo a arquitetura pouco complexa do modelo, leva a que o modelo tende a manter a dificuldade de generalização dos dados. Tudo isto leva a que, o modelo mantenha a dificuldade em aprender corretamente, imagens relativas à classe 1. As Figuras 28 e 29, demonstram os resultados associados às métricas analisadas: *accuracy, precision* e *recall*. Já as Figuras 30 e 31 ilustram respetivamente a variação do custo e da *accuracy*, ao longo das *epochs*.

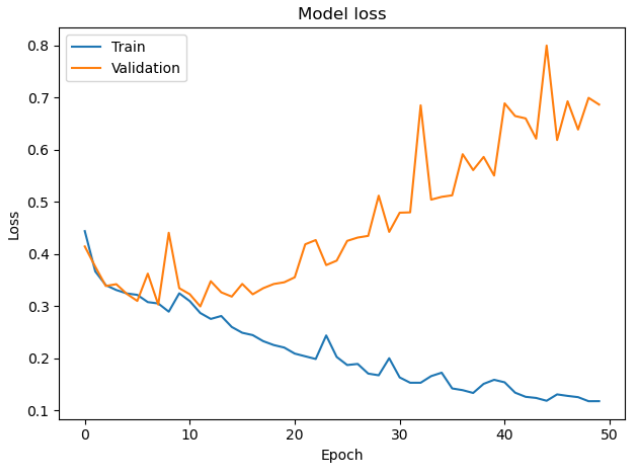

Figura 28 - Variação do custo ao longo das epochs

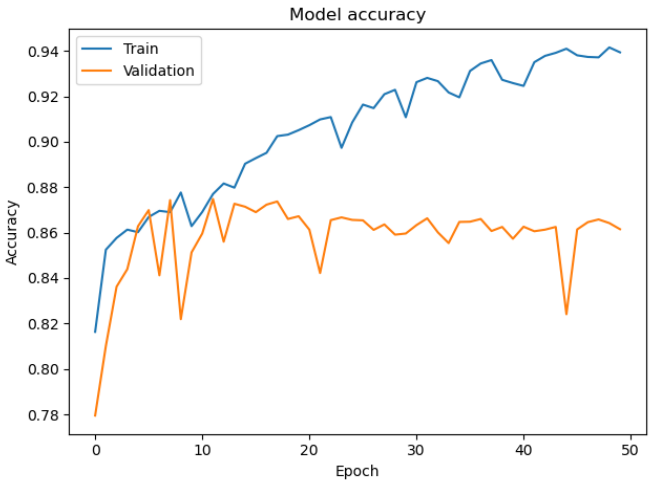

Figura 29 - Variação da accuracy ao longo das epochs

	precision	recall	f1-score	support
Healthy	0.91	0.90	0.90	5888
With IDC	0.72	0.75	0.74	2112
accuracy			0.86	8000
macro avg	0.82	0.82	0.82	8000
weighted avg	0.86	0.86	0.86	8000

Figura 30 - Resultados obtidos (métricas)

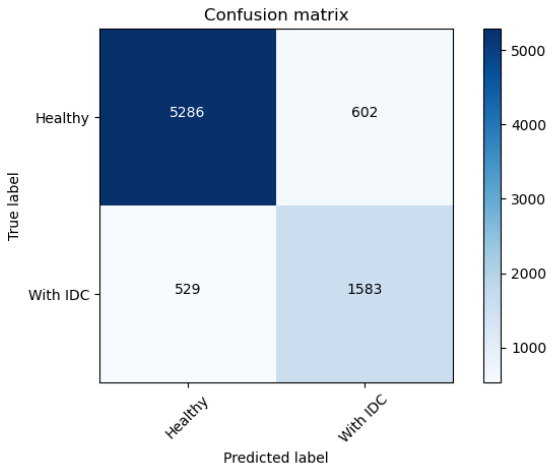

Figura 31 - Matriz confusão

Recorrendo à visualização das Figuras 28 e 29 é possível comprovar que o modelo sofre claramente de *overfitting*. Ou seja, associando os resultados obtidos atrás, com os resultados obtidos neste exemplo: é possível comprovar, que existe a necessidade de aumentar a complexidade da rede, de modo a que seja possível apresentar uma melhoria na aprendizagem e na generalização dos modelos.

Relativamente às métricas obtidas, Figuras 30 e 31, é possível constatar que o modelo apresenta uma maior dificuldade na classificação das imagens associadas à Classe 1. Ainda assim, este valor foi superior ao expectável e ao obtido anteriormente. Com isto, é possível concluir que a utilização de um maior nº de dados, ajuda o modelo a generalizar mais corretamente os dados.

Mas, estes valores necessitam de ser otimizados. Ambas as classes devem possuir resultados semelhantes entre si.

Sendo assim, é crucial a definição de um modelo, que contenha um nº de parâmetros, ajustado face ao volume de dados utilizado. Pois, só assim é possível garantir um modelo, que não revele *overfitting, underfitting* ou resultados tendenciosos.

Isto é, é necessário identificar um equilíbrio entre a complexidade do modelo, e ainda o nº de dados a utilizar (**é necessário ter em consideração outros pontos, como: redução do efeito das classes não balanceadas, mas esta análise está a seguir uma abordagem progressiva**).

Utilização de Arquiteturas mais complexas/profundas:

Tendo em conta, os resultados obtidos anteriormente, e de acordo com a análise efetuada e as conclusões retidas, o autor decidiu recorrer à definição de uma rede mais complexa, com o objetivo de perceber se existem melhorias, com a definição de uma arquitetura mais complexa na resolução do problema.

Este problema, revela-se um problema complexo, dadas as várias condicionantes em causa, levando a considerar que a utilização de modelos muitos simplistas, como os modelos utilizados anteriormente, não permitem a obtenção de modelos capazes de criar uma generalização adequada dos dados.

Neste exemplo, foi apenas considerada a utilização de uma arquitetura, baseada em *AlexNet,* mais complexa que as apresentadas atrás. Ainda assim não foi aplicada qualquer estratégia de "combate" ao problema relativo ao não balanceamento das classes. Ou seja, apesar de ser expectável a obtenção de ligeiros benefícios com a adoção de uma arquitetura mais complexa, o facto de existirem muito mais imagens alusivas à classe 0 ("saudável"), pode continuar a dificultar a generalização adequada dos dados.

Desta forma, foi criado um modelo apresentando um maior nº de camadas convolucionais. Já o nº de parâmetros de treino, manteve-se semelhante ao definido atrás, visto que a utilização de um maior nº de filtros e/ou de neurónios, implicaria um aumento deste número, e considerando o problema de *overfitting* identificado atrás, constatamos que o modelo já está a sofrer de *overfitting*, e ao considerar um maior nº de parâmetros, acabaria por agravar ainda mais este problema.

Resumindo, foi considerado um maior nº de camadas convolucionais, passando de duas simples camadas convolucionais para 6 camadas convolucionais, sendo adicionadas duas *stacked* CNN Layers. Tal como referido anteriormente, o nº de parâmetros de treino manteve-se semelhante aos exemplos demonstrados anteriormente.

Ou seja, a rede tornou-se bem mais profunda, do que as representadas atrás. A utilização de um maior nº de camadas convolucionais (incluindo *stacks*), permite a extração de *features* mais complexas. Ou seja, à medida que a rede se torna mais profunda, existe uma maior probabilidade de identificar particularidades nos dados. Visto que, cada camada convolucional "filtra" o *output* das camadas anteriores. Num problema complexo, como é o caso, a utilização deste tipo de arquiteturas tende a ser mais benéfico.

As Figuras 32 e 33 descrevem os resultados obtidos, enquanto que as Figuras 33 e 34 ilustram respetivamente, a variação do custo e da *accuracy*, ao longo das *epochs*.

	precision	recall	f1-score	support
Healthy	0.96	0.94	0.95	662
With IDC	0.74	0.82	0.78	138
accuracy			0.92	800
macro avg	0.85	0.88	0.87	800
weighted avg	0.92	0.92	0.92	800

Figura 32 - Resultados obtidos

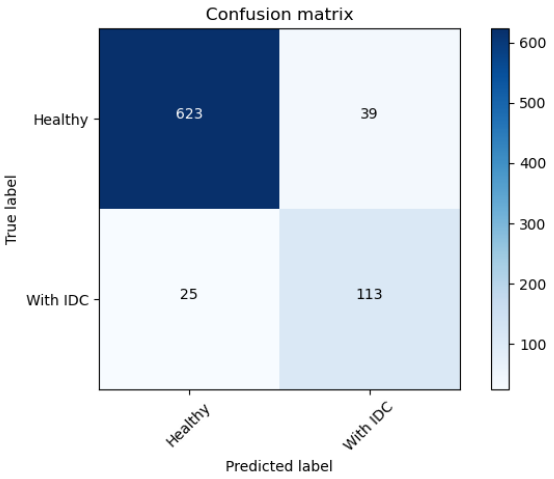

Figura 33 - Matriz confusão

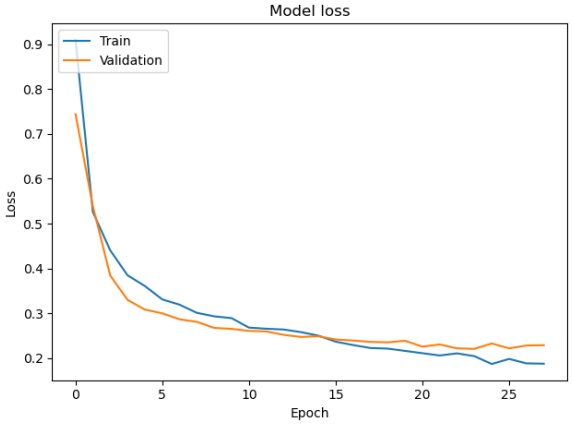

Figura 34 - Variação do custo ao longo das epochs

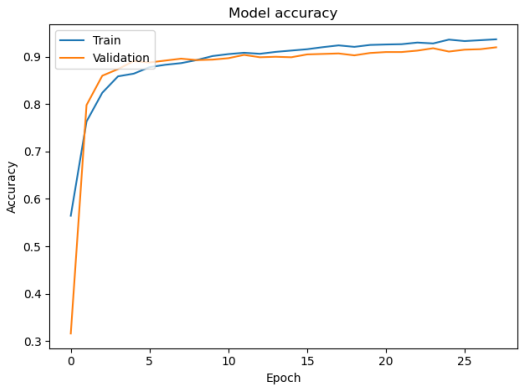

Figura 35 - Variação da accuracy ao longo das epochs

Observando, as Figuras 32 e 33 e comparando com as Figuras 22, 23, 24 e 25 é possível constatar que a utilização de uma rede mais profunda, trouxe bastantes significativas. Apesar, da classe 1 ("IDC"), ainda não apresentar os resultados pretendidos, é possível identificar uma melhoria significativa das métricas em análise.

Analisando, as Figuras 34 e 31 é possível destacar que ao contrário dos modelos anteriores, este não revelou problemas de *overfitting*. Nos modelos analisados atrás, ao fim de 10 *epochs* já existiam indícios de *overfitting*. Esta rede, ao fim de 28 *epochs* não apresentou indícios de *overfitting*, apesar de começar a dar indícios de estagnação (na fase de treino foi definido um *callback,* que verifica se ao fim de 4 *epochs*, o modelo não reduzir o custo de validação, dá-se por terminado o treino).

Posteriormente, deve-se recorrer à utilização de um maior volume de dados para treino, bem como à utilização de técnicas de "redução de *unbalaced classes",* como: *Undersampling*, na tentativa de melhorar ainda mais os resultados obtidos.

```
valuesLayers = (
    8,
    12,
    16,
    16,
    12,
    8
)
```

Da mesma forma, foi aplicada uma rede CNN mais profunda, mas desta vez considerando um maior volume de dados. O objetivo deste exemplo passa essencialmente, por tentar conciliar as vantagens, que as redes mais profundas oferecem (e evidenciadas nos parágrafos anteriores), com a utilização de um maior conjunto de dados.

As redes CNN, tal como as restantes redes de *Deep Learning*, requerem a utilização de elevadas quantidades de dados, de modo a que seja possível treinar o elevado nº de parâmetros, que normalmente as mesmas apresenta.

Ou seja, deve-se tentar um equilíbrio entre a complexidade do modelo (ou profundidade), com o nº de dados utilizados em treino. Visto que, só dessa forma é possível garantir a obenção de um modelo não tendencioso, e sem problemas de *underfitting* ou *overfitting*.

Contudo, nem sempre é fácil estabelecer esta estabilidade, podendo existir a necessidade de alteração do modelo, adicionando mais camadas convolucionais, ou adicionar ainda mais dados ao problema. Isto é, é um processo que necessita de diversos ajustes "incrementais", até que seja possível a obtenção de um modelo capaz.

Sendo assim, foi criado um exemplo para responder a este problema. A solução final resulta de um processo incremental, onde o autor teve a necessidade de alterar/ajustar diversas componentes, de modo a que fosse possível obter uma solução final, que permitisse a obtenção de um modelo, que não revelasse problemas de *overfitting*, e retornando da mesma forma os melhores resultados possíveis.

Foi considerada a utilização de 100000 imagens para treino, sendo as mesmas dividas, numa proporção de 60%-20%-20%, respetivamente para o conjunto de treino, validação e ainda de teste. A arquitetura da rede manteve-se inalterada, isto é, composta por duas camadas convolucionais, ainda por duas *stacked cnn layers* e finalmente uma *dense layer*. A única

alteração face ao modelo anterior, remete ao uso de um maior nº de filtros, nas camadas convolucionais, visto que ao ter existido um aumento considerável do nº de imagens (de 5000 para 100000), é necessário aumentar o nº de parâmetros de treino, de modo a que a rede consiga aprender corretamente os dados.

O valor relativo ao *batch size*, isto é, o nº de atualizações efetuadas aos parâmetros da rede durante a fase de treino, também teve de ser devidamente ajustado, de modo a que fosse possível estabelecer um nº correto de atualizações, pois só assim é possível evitar *overfitting*.

Foi ainda considerada a utilização de dois *callbacks*, tendo como principais objetivos: evitar a estagnação em óptimos locais do modelo, ou seja, caso o custo de validação não decresça ao fim de duas *epochs*, o *learning rate* é atualizado, sendo considerado um fator de redução de 0.7. Já o 2º *callback*, é utilizado de modo a garantir que se ao fim de 6 *epochs*, o custo de validação não diminuir, então dá-se por concluído, o processo de treino. Evitando assim, o "desperdício de tempo", na obtenção dos resultados, visto que o modelo nesse momento já se encontra num estado de estagnação.

O treino do modelo, considera diferentes "pesos" para ambas as classes do problema. Ou seja, a classe 0, como é mais representada que a classe 1 então terá um peso bem menor que a classe 1. Este peso, define o valor que é dado a cada instância, de cada classe, no cálculo da função de custo. Desta forma, torna-se possível aplicar uma penalização "igualitária", a ambas as classes (mais representada e menos representada).

As Figuras 36 e 37 ilustram os resultados obtidos. Por sua vez, as Figuras 38 e 39 descrevem respetivamente a variação do custo e da *accuracy*, ao longo das *epochs*.

	precision	recall	f1-score	support
Healthy	0.90	0.94	0.92	10748
With IDC	0.87	0.78	0.83	5252
accuracy			0.89	16000
macro avg	0.89	0.86	0.87	16000
weighted avg	0.89	0.89	0.89	16000

Figura 36 - Resultados obtidos

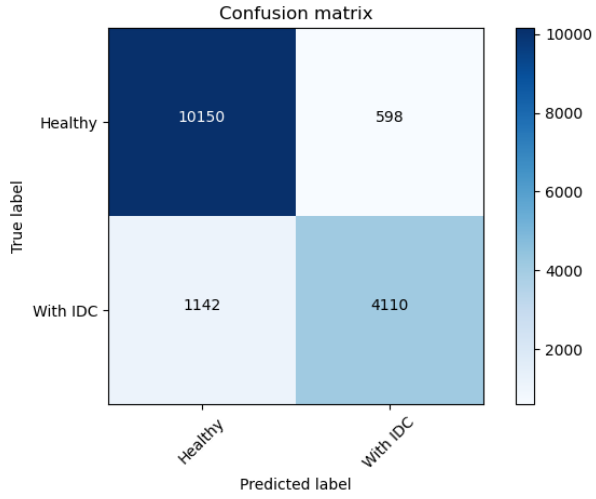

Figura 37 - Matriz confusão

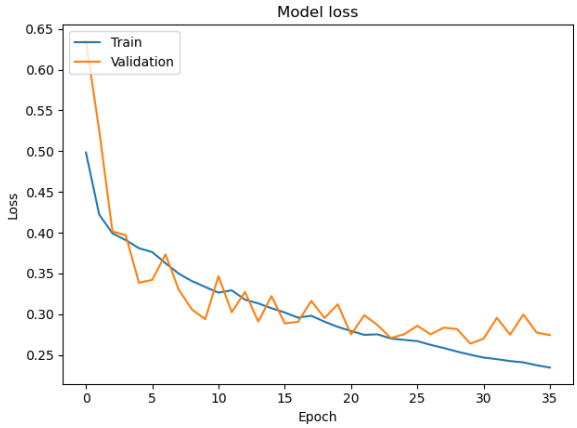

Figura 38 - Variação do custo ao longo das epochs

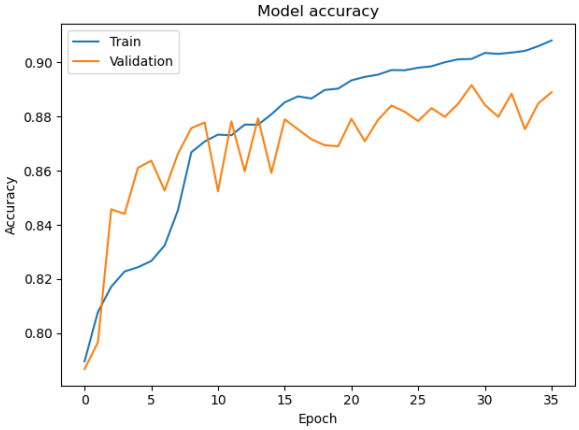

Figura 39 - Variação da accuracy ao longo das epochs

Começando pela análise das métricas obtidas, e ilustradas através das Figuras 36 e 37, é possível constatar que os resultados obtidos, na classificação das imagens alusivas à classe 1, foram os que mais se destacaram (positivamente), até então.

Ou seja, a *precision* centrou-se na gama dos 87%, um valor bem acima do melhor obtido até então 72%. O *recall*, ficou-se nos 75%, um valor ligeiramente superior ao obtido e demonstrado através da Figura 30.

Já os valores relativos à Classe 0 foram um pouco inferiores, aos já demonstrados ao longo deste estudo, “muito por culpa”, da curva de treino ainda estar numa fase “embrinonária”, isto é a *accuracy* de treino, quando a fase de treino parou encontrava-se apenas nos 91%, existindo ainda muita margem de progressão, que iria certamente melhorar as métricas referentes à Classe 0.

Relativamente, à observação das Figuras 38 e 39, é possível constatar que o modelo começou a dar indicadores de estagnação, entre as 20-25 *epochs*. É necessária a realização de mais testes, de modo a conseguir suprimir este problema, de modo a que seja possível a obtenção de melhores resultados.

Após a realização deste exemplo, é possível salientar que a adoção de algumas estratégias consideradas tiveram um impacto positivo nos resultados obtidos, nomeadamente, a definição de uma rede mais complexa e profunda, a definição de pesos distintos na definição da função

de custo para ambas as classes, a utilização de *callback's,* com o objetivo de evitar que o modelo "estagne" em ótimos locais, ou ainda a introdução de camadas *Dropout*.

Seguidamente, irão ser aplicadas estratégias de treino, de modo a reduzir o problema do "balanceamento" das classes, e ainda de modo a melhorar os resultados obtidos, até então. De salientar que os exemplos descritos posteriormente, irão recorrer à utilização de modelos, que apresentem melhorias face ao demonstrado atrás, isto é, modelos que evitem o problema de *overfitting* demonstrado através das Figuras 38 e 39.

Utilização de estratégias de Treino:

Tal como já fora referido anteriormente, as amostras do *dataset* não são balanceadas, perante as duas classes do problema. O não tratamento deste tipo de *dataset's,* geralmente resulta na obtenção de resultados tendenciosos. Isto é, o modelo apresenta uma *acuraccy* elevada, mas muito à conta da classe 0, apresentando mais dificuldades na aprendizagem das imagens, relativas à classe 1.

Como fora indicado, durante a contextualização da arquitetura do Sistema, foram criadas várias estratégias de modo a reduzir o efeito deste problema. Para tal, foram consideradas duas das técnicas mais utilizadas para a resolução deste problema: *Undersampling* e *Oversampling*.

Ambas, referem-se a técnicas de *sampling*, mas implementando abordagens contraditórias, uma em relação à outra. As técnicas de *sampling*, representam abordagens que tentam reduzir o impacto que uma classe apresenta em relação à outra (maior nº de amostras), efetuando alterações, no *dataset*.

Undersampling, procura "balancear" as classes do problema recorrendo à eliminação de amostras referentes à(s) classe(s) mais representativa(s) do problema. Já, a técnica *Oversampling*, aumenta o nº de amostras da(s) classe(s) menos representativa(s), através da duplicação de amostras da classe, ou através da introdução de amostras sintéticas (não irá ser considerada a utilização de amostras sintéticas, visto que estaria a introduzir erros no *dataset*, e tendo em conta o âmbito do mesmo, não deve ser ponderada essa opção).

Undersampling:

A primeira técnica aplicada é: *Undersampling*. Tendo em conta, o elevado nº de amostras existentes, faz mais sentido a utilização desta técnica, em detrimento do *Oversampling*, que é mais utilizado em problemas, que reúnem poucos dados.

Ou seja, a aplicação desta técnica num enorme volume de dados como é o caso, apesar de reduzir este número, continua a existir um elevado nº de amostras para treino. E, passa a existir um *dataset* balanceado, reduzindo dessa forma, a dificuldade em classificar corretamente as imagens associadas à classe menos representativa.

Ainda assim, é expectável que haja um decréscimo nas métricas associadas à classe 0, que eram anteriormente muito elevadas, devido ao facto do modelo estar claramente a demonstrar elevada tendenciosidade, na previsão única da sua classe (*overfitting* da classe mais representativa). Este decréscimo, deve ser erradicado recorrendo à aplicação de um processo de treino mais rigoroso (possibilidade de utilização de *Data Augmentation)*, tendo também especial atenção ao modelo desenvolvido.

Foi considerada a utilização de 120000 imagens, divididas de igual forma, 60%-20%-20%, pelos conjuntos de treino, validação e teste. Já o modelo manteve-se semelhante ao utilizado no exemplo anterior, sendo apenas definidas ligeiras alterações.

Importa ainda salientar, e tendo em consideração o que fora referido no exemplo anterior, neste exemplo foi considerada a utilização do modelo anterior, mas considerando uma melhoria ado nível da sua arquitetura. Isto é, o modelo anterior sofria de *overfitting*, e de modo a reduzir este efeito e a melhorar os resultados e a aprendizagem do modelo, foram efetuados mais testes, de modo a garantir um modelo mais eficiente.

As principais alterações efetuadas foram essencialmente: a alteração do nº de filtros das camadas convolucionais, e ainda ajustes pontuais na arquitetura da rede.

Da aplicação do exemplo resultaram as métricas sintetizadas através das Figuras 40 e 41. Já, as Figuras 42 e 43 ilustram a variação do custo e da *accuracy*, ao longo das *epochs*.

	precision	recall	f1-score	support
Healthy	0.88	0.94	0.91	13014
With IDC	0.86	0.73	0.79	6186
accuracy			0.88	19200
macro avg	0.87	0.84	0.85	19200
weighted avg	0.87	0.88	0.87	19200

Figura 40 - Resultados Obtidos

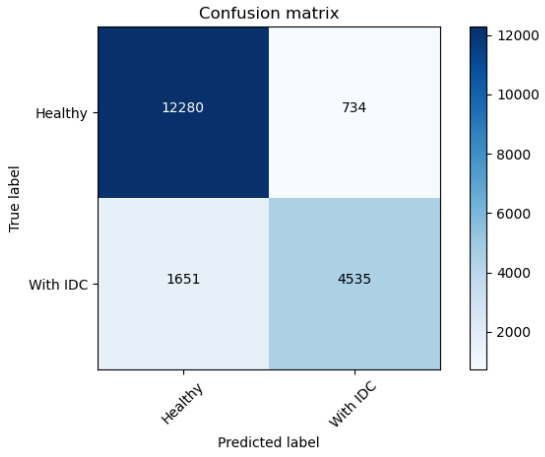

Figura 41 - Matriz confusão

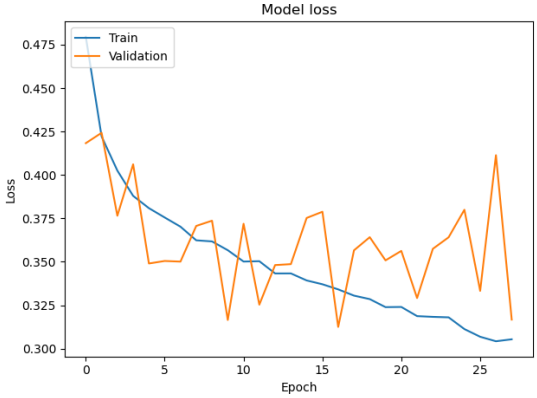

Figura 42 - Variação do custo ao longo das epochs

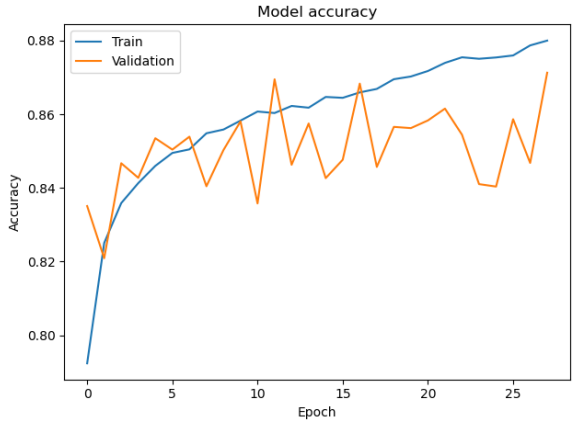

Figura 43 - Variação da accuracy ao longo das epochs

Observando os resultados obtidos, através das Figuras 40 e 41, é possível constatar que os mesmos não estão próximos do expectável. Isto é, o autor esperava a obtenção de resultados superiores.

Contudo, e efetuando uma análise mais ponderada sobre o modelo e os dados considerados, na exemplificação desta abordagem, é possível concluir que o autor foi um "pouco ambicioso", mais concretamente no nº de dados, que considerou, para a resolução deste exemplo.

Ou seja, o autor ao aplicar o conceito de *Undersampling,* está a reduzir, de uma forma significativa, o nº total de amostras utilizadas no processo de treino. Visto que, a proporção entre o nº de imagens da classe 0 e da classe 1, é cerca de 70% - 30%.

Considerando este exemplo, o autor inicialmente definiu um nº total de amostras de 120000 (a distribuir pelos vários conjuntos – treino, validação e ainda teste). Ou seja, aplicando a distribuição pelos vários conjuntos, foram definidas 76800 amostras para treino. Mas, ao aplicar o conceito de *Undersampling*, este valor foi reduzido para 42154. Dado que, o conjunto inicial de amostras de treino da classe 0 era igual a 55723, já da classe 1 de 21077. O nº de amostras da classe 0 foi reduzida de 55723 para 21077. Perfazendo assim, o valor final de 42154.

Desta forma é possível identificar uma redução do nº de imagens de treino, de cerca de 45%. Um valor muito elevado, reduzindo de forma considerável o nº de amostras de treino.

Neste caso, o utilizador tinha ainda acesso a mais dados, ou poderia ter recorrido a técnicas, que aumentam o nº total de amostras, como por exemplo: *Data Augmentation*. Porque, ao ser considerado um nº tão baixo de amostras, dificulta o processo de aprendizagem do modelo, bem como a sua generalização, algo que é bem visível através das Figuras 42 e 43. Isto é, o modelo, precocemente deixa de conseguir generalizar corretamente, dificultando assim a obtenção de bons resultados.

Ou seja, quando se recorre à aplicação deste tipo de técnica, deve existir o cuidado, de "voltar" a aumentar o nº total de amostras a utilizar, mas neste caso num contexto já balanceado.

Outro aspeto que foi possível observar, é afeto à dificuldade contínua de classificar corretamente a classe 1. Ou seja, apesar do *dataset* já se encontrar devidamente balanceado, contínua a existir uma maior dificuldade na aprendizagem desta classe. Por exemplo, o *recall* alusivo à classe 1 (a métrica, que exige maior preocupação), permanece em valores baixos, na ordem dos 73%.

De forma a colmatar este problema, é necessário garantir que o modelo generalize o melhor possível, de modo a evitar maiores discrepâncias entre *precision* e *recall*.

Undersampling* com *Data Augmentation:

Outra das estratégias, a utilizar em contextos semelhantes, ao problema em estudo, prende-se com a técnica *Data Augmentation*. Esta técnica é normalmente utilizada, quando o problema a resolver enfrenta algumas limitações, tais como: (1) reduzido nº de amostras, (2) problemas, onde as classes não são balanceadas e (3) na tentativa de melhoria dos resultados obtidos.

O problema a resolver enquadra-se nos últimos dois pontos referidos, no parágrafo anterior. Isto é, para além da utilização de abordagens, como: *Undersampling* ou *Oversampling*, recorre-se à utilização desta técnica, como um complemento a utilização destas duas técnicas.

É importante a utilização desta abordagem em simultâneo, com as restantes técnicas aplicadas, com o objetivo de melhorar a aprendizagem e a generalização do modelo. Porque, considerando a utilização única de *Undersampling*, é possível constatar de antemão, que apesar de se erradicar o problema do não balanceamento das classes, o nº de amostras utilizadas para treino decresce. Então, uma das abordagens utilizadas para aumentar o nº de amostras de treino, passa pela utilização de *Data Augmentation*, que aplica um conjunto de transformações aos *inputs*, de modo a que seja possível ter acesso a um conjunto de dados elevado.

Neste problema, não existem condicionantes na aplicação de algumas técnicas de *augmentation,* como por exemplo: *flips*. Porque, ao rodar uma imagem ao longo de um determinado eixo, o "valor" da imagem continua a ser exatamente o mesmo.

Contudo, é necessário ter em consideração que a utilização de *Data Augmentation,* requer um maior custo computacional, para treino dos modelos. Para além disso, nem sempre é assegurada a melhoria dos resultados.

Sendo assim, e de modo a avaliar a aplicação desta técnica, no problema em estudo, foi considerada a sua aplicação no modelo exemplificado atrás (isto é, em simultâneo com *Undersampling*).

Para tal foi considerada a sua aplicação considerando o mesmo nº de amostras, as 120000 imagens iniciais. Recorrendo à aplicação simultânea de ambos os conceitos, *Undersampling* e *Data Augmentation*, é expectável que haja uma redução de alguns dos problemas destacados atrás, tais como: *overfitting* do modelo, ou ainda discrepância muito acentuada entre *precision* e *recall*.

Os resultados obtidos encontram-se ilustrados através das Figuras 44 e 45. Já a variação do custo e da *accuracy*, ao longo das *epochs*, nas Figuras 46 e 47.

	precision	recall	f1-score	support
Healthy	0.93	0.91	0.92	14386
With IDC	0.75	0.80	0.77	4814
accuracy			0.88	19200
macro avg	0.84	0.85	0.85	19200
weighted avg	0.89	0.88	0.88	19200

Figura 44 - Resultados obtidos

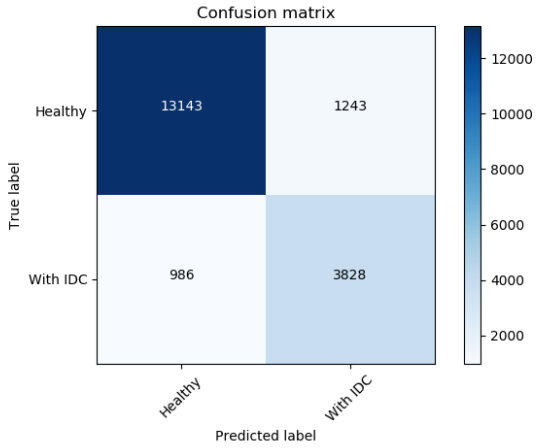

Figura 45 - Matriz confusão

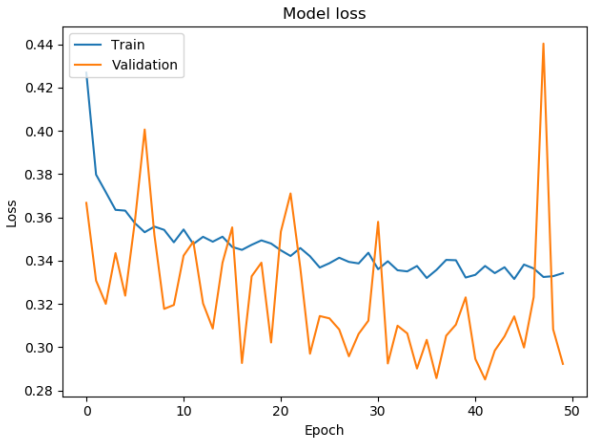

Figura 46 - Variação do custo ao longo das epochs

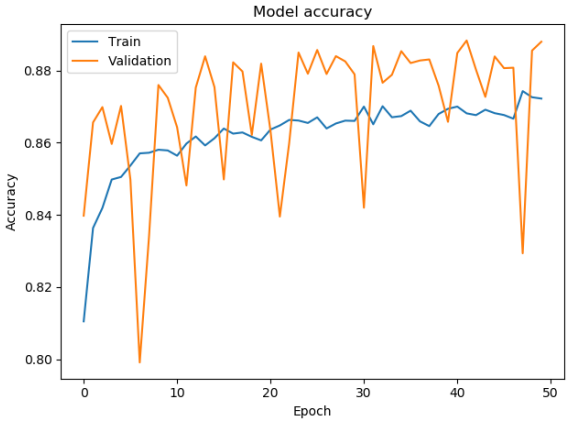

Figura 47 - Variação da accuracy ao longo das epochs

Tal como previsto, os resultados obtidos já se encontram mais "nivelados", face aos anteriormente obtidos. Isto é, a *precision* e o *recall,* apresentam valores semelhantes, não existindo assim uma elevada dispersão entre ambas as métricas. Para além disso, o *recall* é superior à *precision*, revelando-se assim um bom indicador, **porque "o maior erro", é classificar uma amostra de um paciente como saudável, tendo a mesma "vestígios" de *IDC***.

Contudo, e recorrendo à observação das Figuras 46 e 47 é possível constatar a presença de *underfitting*, ou seja, o modelo revela-se pouco complexo, para conseguir aprender e generalizar corretamente os dados.

De modo a melhorar os resultados obtidos, é necessário retificar/alterar a arquitetura do modelo previamente definida. Existindo assim diversas abordagens, para reduzir este problema, como por exemplo: o aumento do nº de filtros e/ou neurónios das camadas do modelo, ou ainda a introdução de mais camadas no modelo, isto é, a criação de uma rede mais profunda.

Tendo em consideração, o nº atual de camadas convolucionais, e o *input shape* das imagens, não é possível recorrer ao aumento do nº de camadas convolucionais, dado que já não é possível estabelecer um mecanismo de *pooling*. Sendo assim, a abordagem a utilizar passa pelo aumento do nº de filtros inerentes às camadas convolucionais já utilizadas.

De forma, a melhorar os resultados obtidos foi então considerada a utilização de uma rede mais complexa, aliando da mesma forma um maior nº de dados, de modo a encontrar um ponto de equilíbrio entre ambas as vertentes. O nº total de dados utilizado foi de 150000 (a dividir pelos vários conjuntos de dados).

É expectável, a obtenção de um modelo mais robusto, isto é, que não demonstre problemas de *underfitting*. Já ao nível dos resultados obtidos, é difícil estabelecer uma previsão que seja fidedigna, ainda assim espera-se que haja uma melhoria dos mesmos, ainda que ligeira (dada a dificuldade, na previsão das imagens alusivas à classe 1).

A aplicação do modelo, descrito atrás, gerou os resultados enumerados através das Figuras 48 e 49. Já, as Figuras 50 e 51 descrevem a variação do custo e da *accuracy*, ao longo das epochs.

	precision	recall	f1-score	support
Healthy	0.93	0.92	0.92	17178
With IDC	0.80	0.83	0.81	6822
accuracy			0.89	24000
macro avg	0.86	0.87	0.87	24000
weighted avg	0.89	0.89	0.89	24000

Figura 48 - Resultados obtidos

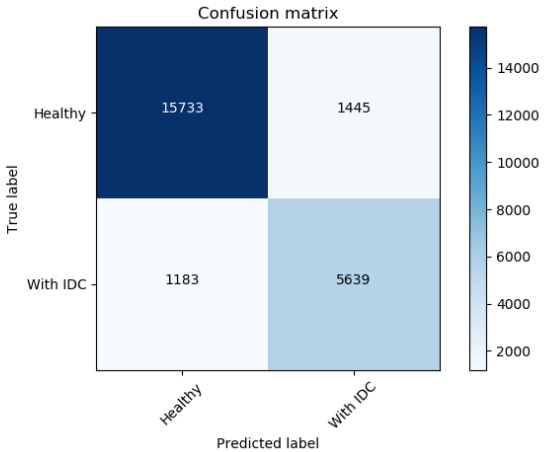

Figura 49 - Matriz confusão

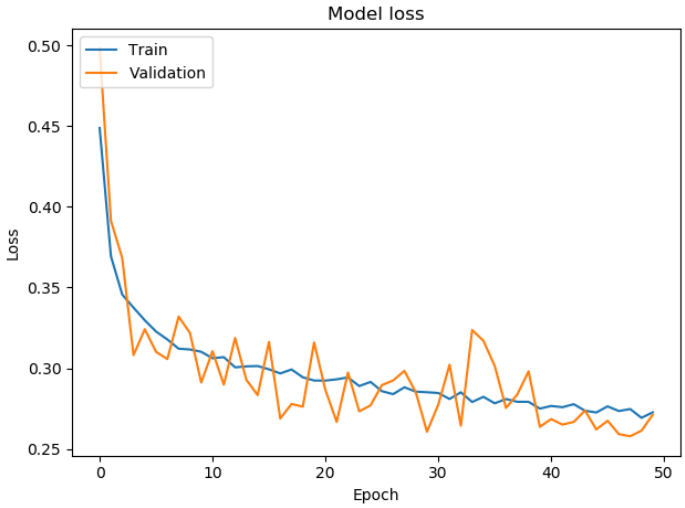

Figura 50 - Variação do custo ao longo das epochs

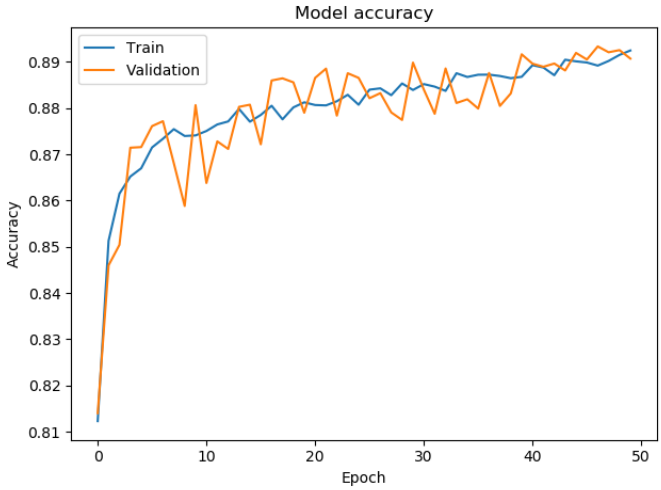

Figura 51 - Variação da accuracy ao longo das epochs

Recorrendo à visualização das Figuras 48 e 49, é possível constatar os bons resultados obtidos. Isto é, foi possível manter os bons indicadores relativos à classe 0, e do mesmo modo existiu um aumento das métricas relativas à classe 1, nomeadamente e num contexto mais expressivo do *recall*.

Ao fim de 50 *epochs* foi possível a obtenção de uma *accuracy* final de 89%, destacando que existiu uma melhoria bem significativa dos resultados alusivos à classe 1, ao longo de toda a análise efetuada. Conseguindo, dessa forma erradicar a tendenciosidade constante do modelo, em reter a sua aprendizagem unicamente para a classe 0.

Muito provavelmente, os resultados obtidos podem sofrer um aumento, ainda que ligeiro, visto que e recorrendo à análise das Figuras 50 e 51 é possível identificar que o modelo ainda possui margem de progressão, quer de aprendizagem, quer de generalização.

Ou seja, algumas das abordagens possíveis para a sua melhoria, passam essencialmente: (1) aumento do nº de *epochs*, visto que estar a ser utilizado um *Schedule callback,* que reduz o *learning rate* de acordo com a aprendizagem do modelo, não existindo assim problemas, neste aumento, pois quando o modelo estagna, o modelo para (*stop callback)* ou (2) aumento do nº

de dados utilizados, porque foram apenas utilizadas 150000 das 277524 imagens existentes, existindo ainda informação disponível para o modelo aprender corretamente os dados.

Otimização do modelo – PSO:

Com o objetivo de reduzir o poder computacional exigido na execução do modelo, recorreu-se à utilização do *Particle Swarm Optimization (PSO)*, na otimização de alguns dos hiperparâmetros do modelo, nomeadamente o nº de filtros das camadas convolucionais e ainda o nº de neurónios, da *Dense layer* intermédia.

Ou seja, o objetivo incide, na procura dos valores mínimos de cada hiperparâmetro, de modo a reduzir o poder computacional e tempo na execução do modelo, mas ao mesmo tempo garantir "bons resultados", mais concretamente um elevado *recall* e *precision*, com maior interesse na Classe 1 – estes valores necessitam de ser otimizados "ao máximo", dado que a segurança dos pacientes está em risco.

Sendo assim, foi criada uma função, que descreve essencialmente o objetivo descrito atrás (função objetivo). Esta função representa um "balanceamento", entre os principais interesses a atingir. Isto é, são fornecidas importâncias distintas às várias condicionantes impostas na função, como por exemplo: o *recall* é mais importante que a *precision*.

Esta função, é representada tal como qualquer outra função utilizada num algoritmo de otimização, isto é, sobre uma função de custo. Ou seja, as partículas descrevem o seu comportamento de acordo com o seu "custo", e ainda das restantes formigas.

A Fórmula 3 descreve a função objetivo criada.

$$loss = \left(2.0 * \left(1.0 - \frac{1.0}{totalFilters} + 1.0 - \frac{1.0}{totalNeurons}\right)\right) + 1.5 * \left(1.0 - \frac{1.0}{acc}\right) + 3.0 * \left(1.0 - \frac{1.0}{prec}\right) + 5.0 * \left(1.0 - \frac{1.0}{rec}\right) \quad (3)$$

Relativamente à Fórmula 3, *totalFilters* representa o somatório do nº total de filtros utilizados, ao longo das várias camadas convolucionais. Sendo que, é atribuído um peso a cada camada, visto que existem camadas convolucionais, onde existe um maior interesse na sua redução, como por exemplo, as *stacked* CNN, que agregam um maior nº de parâmetros face às restantes camadas. O *totalNeurons* segue a abordagem referida anteriormente. A *acc* representa a percentagem de acertos final do modelo. Já a *prec* e o *rec,* representam respetivamente a *precision* e o *recall* obtidos, na classificação das imagens.

Ou seja, as partículas ao longo das várias iterações, têm como principal objetivo minimizar o *loss*.

Para tal, foi considerado um nº total de partículas igual a 20 e ainda um nº de iterações também igual a 20. Já, a topologia considerada foi a *Global Best*. Não foi considerado um valor mais elevado quer de partículas, quer de iterações dado o tempo exigido, para a obtenção final dos resultados. Ainda assim, recorrendo à abordagem atual é esperada a obtenção de um modelo quer se revele bem menos complexo que o obtido previamente, nomeadamente a redução do nº de parâmetros de treino, e ainda a obtenção de resultados semelhantes e/ou melhores.

Otimização do modelo – GA:

Para além da aplicação do PSO, foi ainda considerada a aplicação de outro algoritmo de otimização, o *Genetic Algorithm (GA).* O objetivo, é exatamente o mesmo garantir a obtenção

de um modelo menos complexo, e garantindo melhores resultados e/ou semelhantes. Para tal, a função objetivo considerada é exatamente a mesma, isto é, a Fórmula 3.

A representação da solução foi estabelecida, recorrendo ao formato binário, isto é, cada hiperparâmetro a otimizar, é representado em formato binário, onde o seu valor representa a dimensão máxima, do hiperparâmetro. Ou seja, ao otimizar o filtro de uma rede convolucional, o tamanho do *bitarray* é estabelecido, de acordo com o valor máximo que se pretende otimizar, por exemplo: tamanho do *bitarray* igual a 7, estabelece um valor máximo de $2^7 = 128$. Perante todos os hiperparâmetros a otimizar, é constituído um único *bitarray,* que é composto pela "concatenação" dos vários *bitarray's* criados, para cada um dos hiperparâmetros.

A "configuração" estabelecida para o *PSO*, manteve-se na aplicação do *GA*. Na sua execução foi considerada a utilização de 20 indivíduos e ainda 20 gerações.

Análise VGGNet:

Para além da aplicação do modelo *AlexNet,* foi promovida também a utilização de outras redes, como: a rede *VGGNet.* Esta rede apresenta semelhanças e diferenças, quando comparada com a *AlexNet*, mas a principal diferença entre ambas as abordagens prende-se com a arquitetura "muito típica" da rede *VGGNet*. Isto é, a rede *VGGNet* considera a utilização única de *stacked convolution layers*, ao invés da *AlexNet*, onde as suas primeiras camadas convolucionais são compostas, por uma única Conv *Layer*.

Ou seja, este tipo de arquitetura revela um maior nº de parâmetros de treino, face à *AlexNet*, dada a utilização única de *stacked layers*. Dessa forma, é necessário ter em atenção o nº de filtros e de neurónios, de cada camada, visto que o nº de parâmetros pode ser muito expressivo, aumentando assim o tempo e poder computacional exigidos na execução do modelo.

Todavia, esta rede têm-se revelado mais eficiente que a *AlexNet,* aliás a mesma tem apresentado muito bons resultados, em diversas *benchmarks* aplicadas. O único senão desta arquitetura, é mesmo o elevado poder computacional, que normalmente lhe é exigido, ao contrário de outras redes, "mais leves". Este facto, por vezes leva os investigadores a fornecerem uma maior importância a outros tipos de redes.

Mas, dada a sua simplicidade e eficácia, a mesma foi colocada à prova neste problema. De modo a verificar a sua performance, e comparando-a com a rede *AlexNet*.

A análise efetuada à aplicação desta rede, irá ser mais breve e concisa comparativamente, à análise desenvolvida para a rede *AlexNet*. Visto que, toda a análise desenvolvida atrás, para além da "tentativa" de obter os melhores resultados possíveis, existiu ainda a preocupação de identificar e exemplificar, o comportamento do modelo, quando exposto a diferentes situações. Mas, tendo esse ponto já explorado, segue-se uma análise mais breve da rede *VGGNet*, contemplando assim os "pontos" cruciais da sua análise. Tentando da mesma forma, a obtenção dos melhores resultados possíveis.

VGGNet – sem estratégias de Treino:

O primeiro teste realizado está diretamente relacionado com a utilização desta arquitetura não considerando a aplicação de qualquer tipo de estratégia, isto é, *Undersampling*, ou *Data Augmentation*.

Mas, ao contrário da rede *AlexNet,* não irá ser efetuada uma documentação tão descritiva, das várias abordagens consideradas, ao longo do "teste" desta arquitetura. Dessa forma, irá ser

efetuada a descrição única do “melhor exemplo” identificado, exemplo esse que foi possível obter recorrendo ao teste e análise de inúmeros exemplos, que foram realizados previamente.

Ou seja, antes da identificação do exemplo, que irá ser descrito posteriormente, foram definidos vários exemplos, que promoveram a análise de: (1) complexidade da rede, (2) variação do nº de dados e (3) utilização/variação de camadas de normalização/redução *overfitting,* como: *Dropout*, ou ainda *Batch Normalization*.

Relativamente, à complexidade da rede foi possível comprovar que a utilização de um maior nº de camadas convolucionais permite a obtenção de melhores resultados. Contudo, é necessário ter em atenção o nº de filtros a utilizar, sendo que o mesmo foi ajustado face ao nº de dados a utilizar e à complexidade do problema, caso contrário o nº de parâmetros da rede sobe significativamente.

A variação do nº de dados utilizados no processo de treino, tem impacto na aprendizagem e generalização do modelo, o que por sua vez têm consequências nos resultados finais obtidos. Isto é, quanto maior for o volume de dados considerado, maior é a probabilidade do modelo conseguir generalizar corretamente os dados, mas ainda assim é necessário identificar um equilíbrio, entre o nº de dados e a complexidade do modelo, de modo a que se evite problemas, como *underfitting* ou *overfitting*.

A introdução de camadas, como *Dropout*, ou *Batch Normalization,* têm um enorme impacto na aprendizagem do modelo. Visto que ajudam a reduzir a probabilidade do modelo vir a sofrer de *overfitting*, mais concretamente a camada *Dropout*, que de uma forma aleatória “desabilita” algumas das dimensões de entrada do modelo, isto é, estabelece as unidades de entrada a 0. Por sua vez, o *Batch Normalization* é uma camada utilizada, com o intuito de ajudar a coordenar a atualização dos parâmetros, das várias camadas, de um modelo[5]. A utilização destas camadas, ajudou a reduzir os efeitos de *overfitting*, e a tornar o processo de treino mais rápido.

Tendo em conta, todos os exemplos efetuados, segue-se a demonstração do modelo que demonstrou os melhores resultados.

Importa apenas salientar que foi considerado o uso de apenas 80000 imagens, ao longo dos vários exemplos considerados. Sendo que, o exemplo ilustrado seguidamente, e que representa o melhor exemplo identificado, considerou da mesma forma a utilização das 80000 imagens (a dividir pelos vários conjuntos de dados).

As Figuras 52 e 53 ilustram os resultados obtidos. Enquanto que, as Figuras 54 e 55 descrevem a variação do custo e da *accuracy* ao longo das *epochs*.

	precision	recall	f1-score	support
Healthy	0.92	0.92	0.92	9004
With IDC	0.82	0.82	0.82	3796
accuracy			0.89	12800
macro avg	0.87	0.87	0.87	12800
weighted avg	0.89	0.89	0.89	12800

Figura 52 - Resultados obtidos

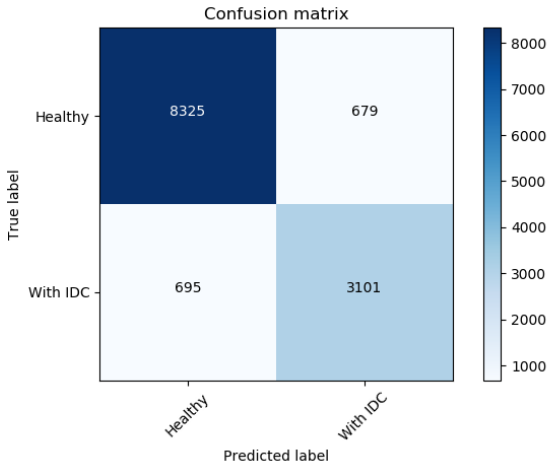

Figura 53 - Matriz confusão

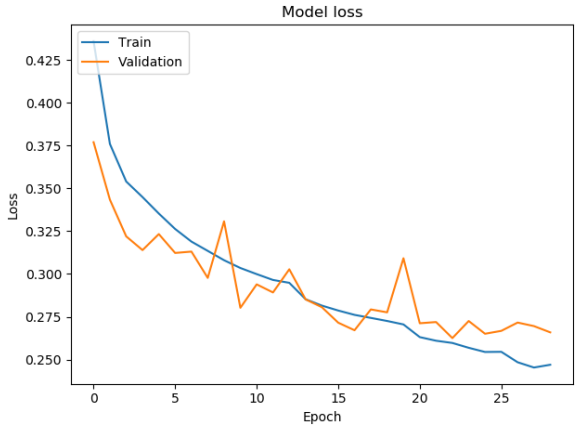

Figura 54 - Variação do custo, ao longo das epochs

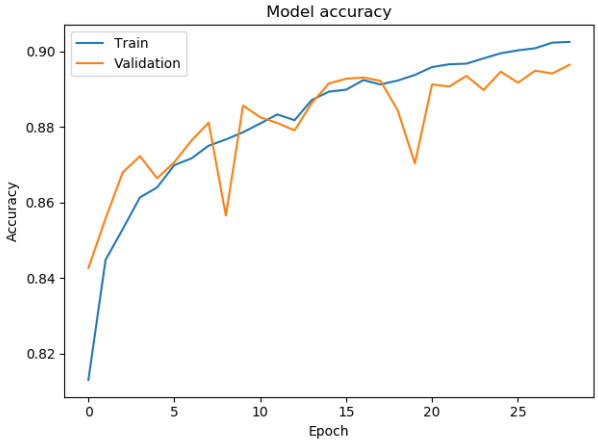

Figura 55 - Variação da accuracy, ao longo das epochs

Observando, as Figuras 52 e 53 é possível comprovar a eficácia que o modelo demonstrou, na classificação de ambas as classes. Apesar, de continuar a existir uma maior preponderância, de acerto para a classe 0, as métricas alusivas à classe 1 revelaram-se adequadas, tendo em consideração os exemplos já aplicados previamente.

Outro aspeto relevante identificado, prende-se com estabilidade existente, entre *recall* e *precision* (mais precisamente na classe 1). Isto é, a obtenção de um *recall* de 82%, tendo em consideração o não balanceamento das classes revela-se um bom indicador, e por sua vez a *precision* também se mantêm elevada, o que é muito importante, pois não existe um "aproveitamento" do *recall* sobre a *precision*.

Analisando da mesma forma, as Figuras 54 e 55, importa destacar a "boa convergência", entre treino e validação até às 15 *epochs*. Ou seja, até esse valor é possível constatar que o modelo apresentou curvas bem definidas, quer na aprendizagem, quer ainda na generalização.

Contudo, após as 15 primeiras *epochs* é notória a presença de *overfitting*, isto é, o modelo deixa de conseguir generalizar corretamente os dados. Algo, que que pode ser atenuado recorrendo a um maior nº de amostras, e/ou reduzindo o nº de parâmetros de treino do modelo.

Ou seja, este modelo revela-se "poderoso", visto que com os devidos ajustes ao mesmo, pode se tornar possível a obtenção de ainda melhores resultados.

***VGGNet - Undersampling*:**

Da mesma forma, foi ainda considerada a aplicação do conceito de *Undersampling,* neste tipo de rede, de modo a tentar verificar o comportamento do modelo, quando considerado um "novo *dataset"*, isto é, um conjunto de dados composto por um nº de amostras coerente entre ambas as classes do problema (classes balanceadas).

A aplicação desta técnica, na rede *AlexNet,* permitiu comprovar a necessidade que existe, no aumento do nº de dados utilizados, quer pela via comum – "adição manual de dados", ou ainda pela adição de mais dados, recorrendo a estratégias, como: *Data Augmentation* ou utilização de amostras sintéticas (medidas utilizadas quando o nº de dados é limitado). Sendo assim, esta técnica foi aplicada considerando um nº de amostras ajustado, face ao modelo, e tendo em conta as amostras que foram "descartadas".

É expectável que os resultados obtidos sejam "balanceados", entre ambas as classes do problema, tal como aconteceu na utilização da rede *AlexNet*. Ou seja, as métricas alusivas à classe 1, devem ser otimizadas, aproximando-se assim, dos resultados da classe 0.

Para tal, foi considerada a utilização de 1456 0000 imagens, a dividir pelos vários *datasets* (treino, validação e teste). Já o modelo manteve-se praticamente inalterado face ao modelo utilizado no exemplo anterior, sendo apenas consideradas alterações ligeiras ao mesmo.

As Figuras 56 e 57 ilustram os resultados obtidos, enquanto que as Figuras 58 e 59 descrevem a variação do custo e da *accuracy*, ao longo das *epochs*.

	precision	recall	f1-score	support
Healthy	0.81	0.97	0.88	13287
With IDC	0.94	0.66	0.77	9113
accuracy			0.84	22400
macro avg	0.87	0.81	0.83	22400
weighted avg	0.86	0.84	0.84	22400

Figura 56 - Resultados obtidos

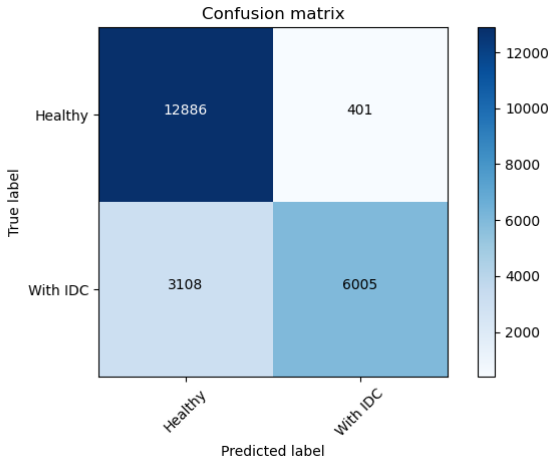

Figura 57 - Matriz confusão

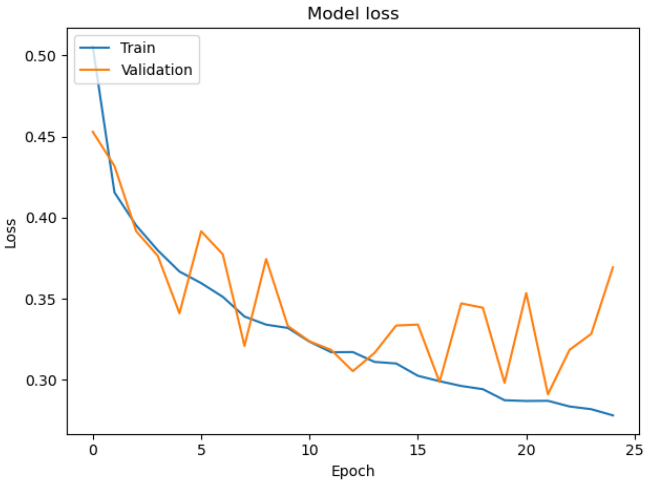

Figura 58 - Variação do custo, ao longo das epochs

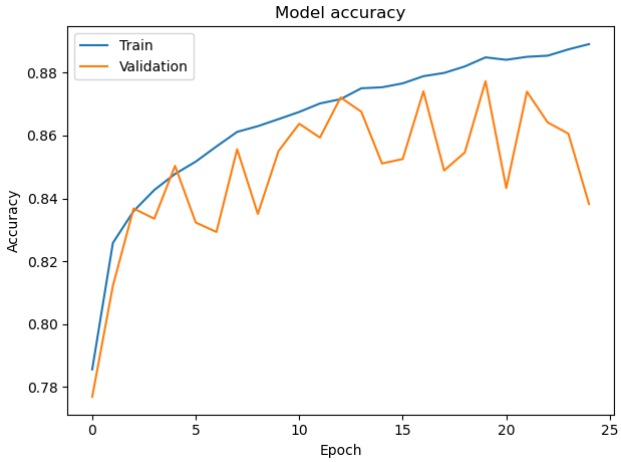

Figura 59 - Variação da accuracy, ao longo das epochs

Observando, as Figuras 56 e 57, é possível constatar que os resultados obtidos estão longe, do pretendido. Ou seja, a aplicação de *Data Augmentation*, não se revelou benéfica, comparando com o exemplo anterior, onde não fora considerada esta técnica.

Ou seja, o balanceamento das classes não produziu as melhorias esperadas. Sendo que, os resultados estão "longe" do pretendido. Recorrendo à análise da Matriz Confusão, Figura 57, conseguimos perceber que a classe 0 também foi afetada por este balanceamento das amostras. Visto que, a *precision* da classe 1 foi elevada muito "à custa", da baixa *precision* da classe 0. Podendo assim concluir, que o modelo está a ter dificuldades na aprendizagem correta dos dados.

Este facto é visível, quando se observa as Figuras 58 e 59, onde é possível constatar a estagnação precoce do modelo, por volta das 12 *epochs*. A introdução de um *dataset* balanceado promoveu ao modelo, a dificuldade em identificar padrões, que pertencessem unicamente a cada classe, algo facilmente reconhecível através da enorme disparidade entre *precision* e *recall*, em ambas as classes. Contrariamente, ao que acontece com os exemplos anteriores (não consideração de *undersampling*), que ao existir um maior nº de amostras da classe 0 facilita a identificação de padrões alusivos a esta classe, mas dificulta a classificação da classe 1.

Uma das abordagens, que é normalmente utilizada para erradicar este problema, passa pelo aumento do nº de dados. Neste caso em concreto, é possível aumentar esse valor de duas formas: (1) ainda existem muitos dados disponíveis, pois apenas foram utilizadas 140000 imagens e (2) utilização de estratégias como *Data Augmentation*.

Seguidamente, e como tentativa de resolução dos problemas identificados, irão ser consideradas ambas as abordagens, na tentativa de criar um modelo mais robusto, e que apresente melhores resultados.

***VGGNet – Undersampling* e *Data Augmentation*:**

De modo a reforçar a *performance* do exemplo anterior, foi considerada a sua aplicação, recorrendo à técnica *Data Augmentation*. Ou seja, para além da utilização de um largo volume de dados, foi considerada ainda a utilização desta abordagem, de modo a atenuar o "desperdício", de amostras que se fez sentir após a aplicação da técnica *Undersampling*.

Foram consideradas as mesmas "*augmentation options",* já consideradas nos exemplos descritos anteriormente, visto que as condicionantes em causa, são exatamente as mesmas. De salientar apenas, que o nº de opções manteve-se reduzido, de modo a evitar que o poder computacional e tempo exigidos na execução do modelo, aumentasse ainda mais.

Da aplicação conjunta, de ambas as técnicas, é expectável uma melhoria dos resultados obtidos, dado que o modelo tem acesso a um maior nº de dados, aumentando assim a probabilidade do modelo conseguir aprender e generalizar corretamente as imagens.

Recorreu-se à utilização de 220000 imagens. Foram ainda efetuadas ligeiras alterações no modelo anterior, de modo a dar suporte a um maior nº de dados (*learning rate* e *batch size*). Já, a utilização dos *callbacks* manteve-se inalterada, de modo a evitar a estagnação em ótimos locais.

Da aplicação do modelo resultaram as métricas ilustradas através das Figuras 56 e 57. Já, as Figuras 58 e 59 descrevem a variação do custo e da *accuracy*, ao longo das *epochs*.

	precision	recall	f1-score	support
Healthy	0.91	0.92	0.91	24548
With IDC	0.81	0.78	0.80	10652
accuracy			0.88	35200
macro avg	0.86	0.85	0.85	35200
weighted avg	0.88	0.88	0.88	35200

Figura 60 - Resultados obtidos

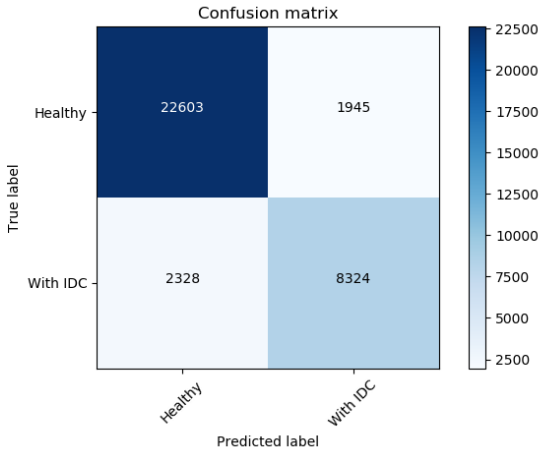

Figura 61 - Matriz Confusão

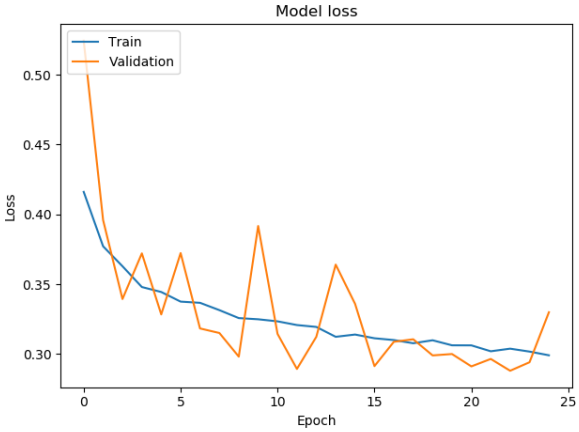

Figura 62 - Variação do custo, ao longo das epochs

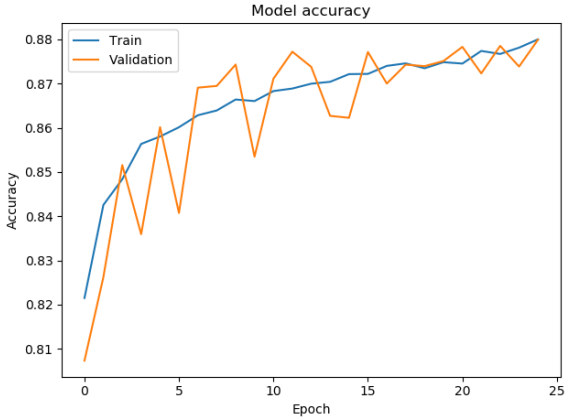

Figura 63 - Variação da accuracy, ao longo das epochs

Recorrendo à observação das Figuras 60 e 61, é possível constatar a melhoria nos resultados, face ao exemplo anterior.

Ou seja, as indicações referenciadas atrás, revelaram-se benéficas, na melhoria dos resultados. Dado que, as métricas relativas a cada uma das classes encontram-se muito mais balanceadas, para além a *f-score*, em ambas as classes revelou-se bem mais ajustada, bem como a *accuracy* global.

Ainda assim, estes resultados poderiam ser ainda melhores, dado que o modelo ainda não se encontrava em "estganação". Contudo, e de modo a garantir uma generalização mais adequada do mesmo, era necessário proceder a um maior nº de tentativas, de modo a que fosse possível "afinar" corretamente o modelo. Sendo que, a solução mais plausível passa pela otimização da rede *VGGNet*, recorrendo a algoritmos de otimização como o *PSO*.

Da análise, às Figuras 62 e 63 é possível concluir que o modelo apresentou indicadores de *noise sensivity*, ou seja, o modelo pode ser demasiado profundo para a resposta a este problema, ou a necessidade de ajuste do *learning rate*. Esta última hipótese é mais plausível, dado que neste exemplo, foi considerada a utilização de vários *callback's*, que reduziam iterativamente o seu valor, dada a dificuldade de convergência, quer em treino, quer em validação. E a partir do meio do treino (12 *epochs*, para diante) é possível visualizar a redução da flutuação do custo e da *accuracy*.

Para além disso, demonstrou indicadores de estagnação precoce (por volta das 12 *epochs*), mas é difícil retirar conclusões muito salientes daqui, visto que a variação do custo ao longo do treino, foi muito ligeira, dificultando da mesma forma a diminuição do custo de validação. Contudo, é possível constatar que apesar de existirem alguns picos, o custo permanece quase sempre mais baixo, que o custo de treino, o que é um bom indicador. Isto pode, querer dizer que o modelo ainda tinha capacidade de generalizar mais, contudo está dependente da "boa aprendizagem" em treino. Sendo assim, é necessário identificar uma arquitetura adequadas, e ainda valores para os hiperparâmetros, que permitam a criação de um modelo mais capaz.

Seguidamente, irá ser dado destaque à otimização do modelo, recorrendo ao algoritmo de otimização *Particle Swarm Optimization*.

VGGNet* – Otimização com *PSO:

O último passo a ser definido, é a tentativa de otimização da arquitetura desenvolvida. Ou seja, tal como fora demonstrado através da rede *AlexNet*, o objetivo da aplicação de algoritmos de otimização, mais concretamente o *Particle Swarm Optimization*, passa pela tentativa de identificar um modelo robusto, que apresente os melhores resultados possíveis, e da mesma forma utilize os menores recursos computacionais.

Apesar de ser exigido algum custo temporal, na otimização dos modelos, é possível identificar uma série de benefícios obtidos, sendo que os mesmos encontram-se enumerados, na demonstração da aplicação do *PSO*, na rede *AlexNet*.

Todas as configurações inerentes à aplicação do algoritmo, mantiveram-se inalteradas, considerando a abordagem aplicada na rede *AlexNet*. Isto é, os hiperparâmetros a otimizar foram os mesmos, a função objetivo considerada permaneceu igual, bem como os hiperparâmetros relativos ao *PSO* (partículas, iterações, entre outros).

Do mesmo modo, é expectável a obtenção de resultados ligeiramente superiores/semelhantes aos obtidos atrás, mais concretamente aos resultados obtidos, não considerando estratégias de treino, isto é, na ordem dos 80%, *precision* e *recall* – classe 1. Ou seja, apesar de ser previsível uma ligeira melhoria dos resultados, o ponto de maio interesse, está relacionado com a expectável redução da complexidade da rede, mais concretamente, a redução do nº dos seus parâmetros.

A obtenção de redes "leves e poderosas" revela-se um fator crucial nos dias de hoje, devido à necessidade constante de treinar e prever os modelos, no menor espaço de tempo e recorrendo ao menor custo possível. Esta foi a principal razão, que levou à utilização do *PSO*, para além da melhoria dos resultados, e da redução de trabalho manual ou evitar a utilização de técnicas muito exaustivas, como *GridSearch*.

Sendo assim, seguidamente encontram-se enumerados os resultados obtidos, bem como uma análise breve aos mesmos. As Figuras 60 e 61 ilustram os resultados obtidos, enquanto que as Figuras 62 e 63 descrevem a variação do custo e da *accuracy*, ao longo das *epochs*.

Referências:

[1] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.725.4294&rep=rep1&type=pdf → Automatic detection of invasive ductal carcinoma in whole slide images with Convolutional Neural Networks

[2] https://www.breastcancer.org/symptoms/types/idc

[3] https://medium.com/minimalist-pharmacist/breast-cancer-disease-state-review-patient-case-presentation-4ac505174c4e

[4]https://www.universityofcalifornia.edu/news/pigeons-can-distinguish-cancerous-breast-tissue-normal

[5] https://arxiv.org/pdf/1502.03167.pdf --> Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift

[6] https://www.ncbi.nlm.nih.gov/pubmed/27563488 --> Deep learning for digital pathology image analysis: A comprehensive tutorial with selected use cases. Resultados → F-Score: 76%

[7] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4977982/ → Artigo Original, por acaso também usou *AlexNet*