Universidade de Brasília
Departamento de Estatística

REDES NEURAIS CONVOLUCIONAIS APLICADA NO APOIO AO DIAGNÓSTICO COM BASE EM RADIOGRAFIAS.

Gabriel José dos Reis Carvalho

Relatório final apresentado para o Departamento de Estatística da Universidade de Brasília como parte dos requisitos necessários para obtenção do grau de Bacharel em Estatística.

Brasília
2022

Gabriel José dos Reis Carvalho

REDES NEURAIS CONVOLUCIONAIS APLICADA NO APOIO AO DIAGNÓSTICO COM BASE EM RADIOGRAFIAS.

Orientador(a): Eduardo Monteiro de Castro Gomes

Relatório final apresentado para o Departamento de Estatística da Universidade de Brasília como parte dos requisitos necessários para obtenção do grau de Bacharel em Estatística.

Brasília
2022

Resumo

Redes Neurais Artificiais são um conjunto de funções aninhadas, que tem como finalidade simular tarefas similares ao que se ocorre no cérebro humano. Atualmente, devido a sua ampla capacidade em solucionar problemas não lineares, diversas áreas do conhecimento têm buscado explorar de seus resultados. Sendo assim, o presente trabalho tem como intuito, esboçar um estudo teórico e prático sobre a implementação de modelos de Redes Neurais Convolucionais aplicados em um problema de classificação na área da saúde.

Palavras-chaves: Redes Neurais Artificiais; Redes Neurais Convolucionais; Nih chest x-ray; VGG19.

Abstract

Artificial Neural Networks are a gathering of aligned functions that has the purpose of simulating similar tasks to the human brain. Nowadays, due to its wide capacity to solve non-linear problems, several areas of knowledge have tried to explore its results. Thus, this work aims to project a theoretical and practical study of the implementation of Artificial Neural Networks models applied to a classification problem in the health area.

keywords: Redes Neurais Artificiais; Redes Neurais Convolucionais; Nih chest x-ray; VGG19.

Lista de Tabelas

Lista de Figuras

Sumário

1 Introdução

Deep Learning, ou aprendizado profundo, é um subcampo do aprendizado de máquina em que por uma série de instruções previamente definidas em um programa computacional, ou seja, um algoritmo, este torna-se capaz de observar padrões nos dados inseridos juntamente com as respostas esperadas, para que assim, o algoritmo utilize destes padrões na criação de regras, as quais poderão ser aplicadas a um outro conjunto de dados originando respostas para esse. Portanto, faz-se necessário citar o primeiro algoritmo da história que utilizava-se de cartões perfurados para programar uma máquina analítica a computar números de Bernoulli, este feito deve-se a Augusta Ada Byron King, mais conhecida como Ada Lovelace a primeira programadora.

É de interesse apontar que em uma de suas anotações ela prevê o uso das invenções de Charles Babbage para criar imagens, este é o inventor da primeira máquina analítica de uso geral, conhecida como computador mecânico. Em Ignotofsky (2017), é dito que Ada Lovelace imaginava um mundo em que computadores fizessem mais que meros cálculos e atualmente com a inteligência artificial (IA), o que era apenas imaginação começa a fazer parte do cotidiano da sociedade trazendo benefícios em diversos campos como na área médica, reconhecimento facial, controle de estoques e assistentes virtuais, resumidamente, atividades que eram realizadas apenas por pessoas tendem a ser reinventadas e automatizadas.

Este estudo versará sobre a implementação de algoritmos que deem suporte à decisão clínica com base na análise de radiografias, auxiliando o médico num prognostico mais efetivo tendo um caráter unicamente de segunda opinião. Face ao exposto, é relevante citar o responsável pela primeira radiografia, a qual revolucionou a medicina, possibilitando a partir daquele momento, visualizar o interior do corpo humano de forma não invasiva, este foi Wilhelm Roentgen.

Os fundamentos que baseiam uma aplicação do deep learning na análise de imagens de raio-x, deve-se ao fato de que diferentes doenças tendem a ter alterações características, neste caso em particular no tórax. Dessa maneira, sua viabilidade é atendida, como também sua relevância, pois, atualmente, o erro médio entre os diagnósticos oriundos de interpretação de imagens médicas está na faixa de 20% a 30% (SAMEI, 2006).

Ademais, doenças em que o paciente apresente algum sintoma indicativo de um quadro cardiorrespiratório, é desejável e usual, a realização de uma radiografia de tórax. Por exemplo, suspeitando-se de pneumonia a radiografia ainda é o melhor método de diagnostico, segundo a Organização mundial da saúde (OMS,2001,p.4).

É sabido que um dos pontos basilares para o diagnostico é efetuar um bom exame clínico, que consiste na realização da anamnese. Assim dizendo, um dialogo entre médico

e paciente com a intenção de identificar todos os sintomas, seguido de exames físicos, a fim de coletar sinais compatíveis, em princípio, com o relato do paciente. À medida que é coletado estas informações o médico tende a ter suposições sobre possíveis causas, sendo umas mais prováveis que outras e que minunciosamente serão descartadas ou confirmada por meio de testes (MOYER; KENNEDY, 2003).

Diante das incertezas até a confirmação do diagnostico, é imprescindível exaurir todas as possibilidades, logo, o médico deve amparar-se de todos os recursos disponíveis. Desta maneira, o auxilio de um modelo de classificação poderia agregar na interpretação de radiografias, indicando algum ponto que tenha passado despercebido ou até mesmo corroborando suas ideias inicias. Posto que, médicos são passíveis de erros, seja devido a dificuldade natural em distinguir certas doenças ou até mesmo por vieses cognitivos. Além disso, radiologistas tiveram melhores resultados na avaliação de radiografias quando sugestionados em suas análises (POTCHEN, 2006).

2 Objetivos

Esta Seção lista o objetivo geral deste relatório, bem como seus objetivos específicos. Uma vez que, sejam cumpridos os objetivos específicos elencados, o objetivo geral é satisfeito.

2.1 Objetivo Geral

O objetivo deste relatório de pesquisa é o desenvolvimento e avaliação de um algoritmo de Deep Learning que possibilite detectar automaticamente quatorze (14) tipos de patologias torácicas em rótulos únicos.

2.2 Objetivos Específicos

Os objetivos específicos necessários para que o objetivo geral seja alcançado são:

1. Compreender o processo de classificação de imagens.
2. Desenvolver modelos de classificação capaz de distinguir as patologias.
3. Avaliar o desempenho dos modelos desenvolvidos.

3 Revisão de Literatura

3.1 Aprendizagem de máquina

3.1.1 Métricas de desempenho

Será abordado métricas de avaliação comumente usadas na literatura para avaliar o desempenho de modelos de classificação.

3.1.1.1 Matriz de confusão

A matriz de confusão é uma forma tabular de se visualizar os valores que foram preditos de forma correta ou errada em cada classe. Abaixo é mostrado um exemplo de análise para uma classificação com duas classes, podendo ser generalizada para um número superior de classes.

Tabela 1: Matriz de confusão para classificação de 2 classes.

	Valor predito	
Valor real	Classe a	Classe b
Classe a	$x_{1,1}$	$x_{1,2}$
Classe b	$x_{2,1}$	$x_{2,2}$

Depreende-se da tabela 1, quatro possíveis cenários quando uma classe é e escolhida como centro de análise:

- Verdadeiro positivo (VP): são as observações classificadas pelo modelo como pertencentes a uma das classes, dado que realmente pertencem a esta classe, ou seja, $x_{1,1}$.
- Falso positivo (FP): são as observações classificadas erroneamente pelo modelo dentre a coluna da classe em análise.
- Falso negativo (FN): são as observações classificadas erroneamente pelo modelo dentre a coluna da outra classe.
- Verdadeiro negativo (VN): são as observações classificadas corretamente em outra classe.

3.1.1.2 Precisão

A precisão tem como finalidade mensurar o número de acertos dentre os valores classificados para a classe em análise.

$$\text{Precisão} = \frac{\text{VP}}{(\text{VP+FP})}. \tag{3.1.1}$$

3.1.1.3 Acurácia

A acurácia do modelo tem como finalidade mensurar o número de acertos totais dentre todas as observações.

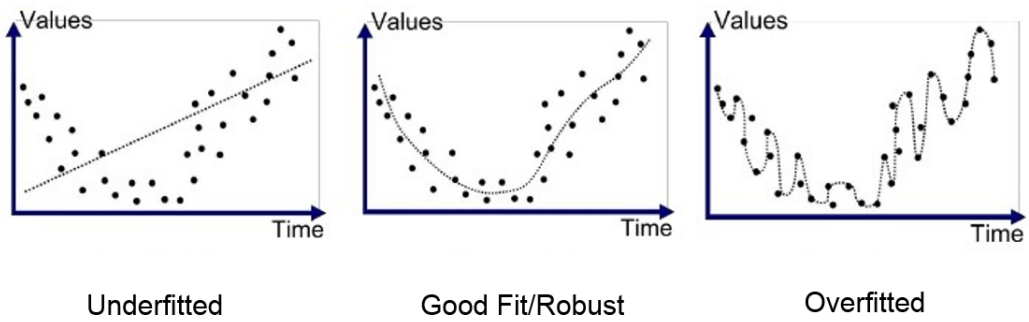
$$\text{Acurácia} = \frac{(\text{VP+VN})}{\text{TOTAL}}. \tag{3.1.2}$$

3.1.1.4 Sobreajuste e Sub-ajuste

O objetivo primordial de um modelo é ser capaz de generalizar padrões aprendidos para dados que ainda não foram observados, e para avaliar se este objetivo esta sendo atendido, faz-se necessário avaliar seu desempenho experimentalmente. No caso em que o modelo não consegue generalizar as regras aprendidas para dados novos, porém tem alto desempenho nos dados de treino, tem-se o sobreajuste também chamado de overfitting.

Entretanto, caso não possua uma boa performance em ambos conjuntos de dados, isto significa que o modelo não aprendeu ou não possuí capacidade suficiente para relacionar o conjunto de dados a uma variável resposta, acometendo em um sub-ajuste também conhecido como underfitting. Em suma, são duas situações em que o modelo não satisfaz seu propósito, segue abaixo as situações supracitadas intercaladas por um modelo desejável.

Figura 1: Representação do sobreajuste e sub-ajuste.



Fonte: Branco (2020).

3.1.1.5 Hiperparâmetros e Conjuntos de Validação

Os modelos de Redes Neurais que serão apresentados possuem várias configurações que devem ser utilizadas para controlar o comportamento do algoritmo de aprendizagem como: taxa de aprendizado, tamanho do mini-lote, épocas, funções de ativação, inicialização de pesos, otimizadores, e arquitetura da rede, para estas configurações atribui-se o nome de hiperparâmetros.

Estes hiperparâmetros estão diretamente relacionados ao desempenho do modelo, e de tão importante serão melhores detalhados ao longo das seções, entretanto, tamanho do mini-lote e épocas serão descritos previamente. O tamanho do mini-lote refere-se ao número de imagens que deverá ser avaliado durante uma iteração, quando todas iterações são efetuadas abrangendo todo o conjunto de imagens, tem-se uma época.

Em Goodfellow, Bengio e Courville (2016), é descrito que se estes hiperparâmetros fossem aprendidos no conjunto de treinamento, sempre escolheriam a capacidade máxima possível do modelo, resultando em overfitting. Para resolver esse problema, utiliza-se de um conjunto de exemplos não observados para avaliar se o modelo possuí um bom desempenho preditivo, e efetuando ajustes nos hiperparâmetros para que ambos conjuntos tenham um bom desempenho. Porém, devido aos ajustes que serão executados, até se escolher um bom modelo, as vezes, isto pode incorrer no overfitting de ambos os conjuntos, fazendo necessário a existência de um terceiro conjunto que será utilizado para ratificar o poder preditivo do modelo.

3.2 Redes Neurais Artificiais

As Redes neurais são um conjunto de ordenamentos, interconectados por uma série de etapas, que tem como finalidade realizar tarefas similares ao que se ocorre no cérebro humano. Exemplificando, suponha que uma criança está aprendendo a executar uma tarefa simples, como distinguir um gato de um cachorro. Para isso, primeiramente faz-se necessário que esta criança observe os dois tipos de animais, como também que alguém a indique, qual dos dois animais seria um gato e qual um cachorro.

Semelhantemente, nos algoritmos de Redes Neurais, depreende-se o conceito de aprendizado supervisionado, que seria o processo de informar as respostas esperadas para cada observação. Além disso, também deduz-se o entendimento de que gato e cachorro seriam duas classes ou rótulos em um problema de classificação.

Prosseguindo com esta contextualização, é factível não esperar que a criança simplesmente visualizando dois animais torne-se capaz de compreender e distinguir com exatidão para demais casos. Visto que, ela poderia ter entendido, por exemplo que gatos

são os animais de menor porte e cachorros os de maior. Entretanto, isto não condiz com a realidade, já que um conjunto de características que define o que é cada um, sendo assim, esta criança deverá ser ensinada com um conjunto maior de observações, até que torne-se capaz de criar uma regra para distinção destes animais.

3.2.1 Componentes

3.2.1.1 Arquitetura

Uma parte substancial na construção de uma rede neural é definir sua arquitetura, que esta diretamente relacionada ao seu desempenho. Para isso, é preciso conhecer as partes que a compõem. Basicamente, uma rede neural é composta por três camadas: uma de entrada, oculta e saída. Doravante, diversas mudanças podem ser efetuadas, como o acréscimo da quantidade de neurônios ou de camadas. Além disto, as camadas não necessariamente precisam ter um encadeamento linear entre suas estruturas, por exemplo, alguns neurônios poderiam retroalimentar camadas antecedentes.

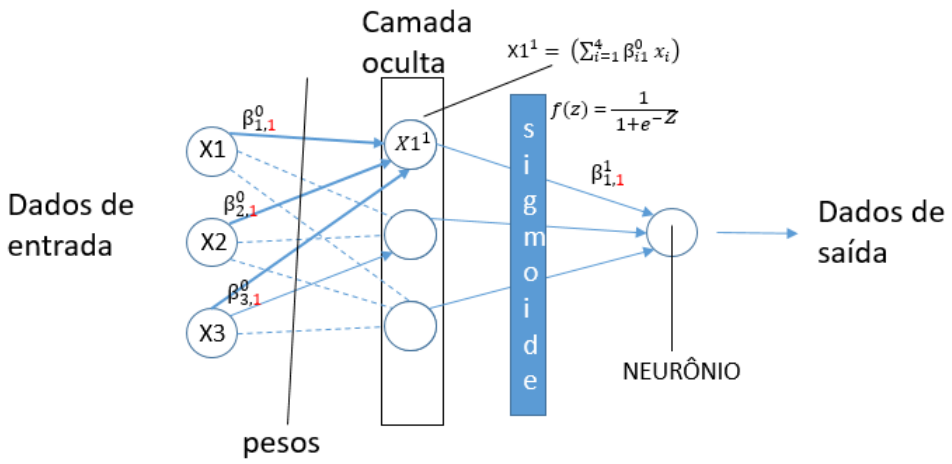
Na figura 2, tem-se uma ilustração de uma arquitetura como supracitado, assim dizendo, essa rede possui uma camada de entrada, uma camada oculta contendo três neurônios, e com função de ativação sigmoide. Seu funcionamento consiste na entrada de três valores sendo estes $X1$, $X2$ e $X3$, os quais são as informações que fluirão ao longo da rede.

Em seguida, são apresentados os pesos $\beta^k_{i,j}$, onde: i refere-se ao neurônio que a informação veio, j para qual neurônio da camada a seguir ele vai e k a camada de referência. Adiante, na camada oculta, tem-se o primeiro neurônio $X1^1$ o qual realizará a soma dos Xi que o conectam ponderado pelos seus respectivos pesos. Desta maneira, por meio da formula abaixo é possível calcular o valor resultante o qual será aplicado em uma função de ativação, esta será descrita na próxima subseção.

$$Xi^{(k+1)} = \sum_{i=1}^{\#i} \beta^k_{i,j} Xi \tag{3.2.1}$$

Por ultimo, tem-se uma camada de saída, a qual neste exemplo possui apenas um neurônio e será responsável pelo calculo da informação final. Vale ressaltar, que este nerônio também possui uma função de ativação, além disto, no contexto de classificação, é usual que o número de neurônios desta ultima camada seja igual ao número de classes.

Figura 2: Arquitetura de uma rede.



Fonte: Elaborada pelo autor.

3.2.1.2 Função de ativação

A função de ativação é caracterizada pelo processamento que ocorre na saída de cada neurônio e são usadas para produzir efeitos de não linearidade proporcionando uma melhor generalização dos dados. Além disso, sua escolha impacta diretamente no ajuste dos pesos, que será melhor detalhado na subseção 3.2.2.2.

Uma das funções que podem ser utilizadas nos modelos de Redes Neurais é a função Identidade, a qual apenas aplicar transformações lineares nos dados, representada pela seguinte equação:

$$f(z) = z. \tag{3.2.2}$$

Uma das funções amplamente utilizadas na literatura é a Sigmoide, que limita seus valores de saída entre zero e um, e é representada pela seguinte equação:

$$f(z) = \frac{1}{1 + e^{-z}}. \tag{3.2.3}$$

Outra função de ativação comumente utilizada na literatura é a Softmax, já que, esta resulta em saídas situadas entre zero e um, além disto, os valores resultantes dos neurônios envolvidos nesta função devem somar um. Por conseguinte, acometendo em uma interpretação mais significativa sobre o quão provável é da imagem pertencer a cada uma das classe, sendo representada pela seguinte equação:

$$f(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{|C|} e^{z_j}}, \tag{3.2.4}$$

Onde: z_i representa os valores para cada neurônio da camada, e $|C|$ é a quantidade

de neurônios na camada em questão.

3.2.1.3 Função de perda

Para realizar ajustes na Rede Neural, primeiramente, faz-se necessário ter uma função que mensure o quão distante está as previsões realizadas do valor real, e esta função é comumente denominada de função de perda ou custo. Quanto maior a perda mais erros o modelo estará cometendo, logo, para se dispor de um modelo melhor, basicamente, deve-se minimizar esta função. Na literatura, uma das funções amplamente utilizadas para modelo de classificação multiclasse é a cross-entropy antecedida por uma função de ativação Softmax, podendo ser expressar, por:

$$L_{CE} = -\sum_{i=i}^{|C|} t_i ln(p_i), \tag{3.2.5}$$

Onde: $|C|$ corresponde ao número de classes a serem classificadas, visto que, para cada classe deve existir um neurônio correspondente; t_i é o valor da probabilidade esperada e p_i é a probabilidade aferida pela função softmax.

3.2.2 Otimização

3.2.2.1 Método da Descida do Gradiente

A Descida do gradiente tem como principio a otimização, que se refere à tarefa de minimizar alguma função diferenciável, como a L_{CE}. É de conhecimento, que a derivada de primeira ordem de uma função é capaz de calcular sua inclinação ou curvatura, em um determinado ponto. Por exemplo, supondo uma função de custo $f(x) = y$ e analisando sua derivada, é possível saber como alterar x para realizar pequenas mudanças em y.

$$f(x + e) \approx f(x) + ef'(x). \tag{3.2.6}$$

O método da Descida do Gradiente consiste em alterar x com base em um processo repetitivo e de sinal oposto da derivada, minimizando $f(x)$. Para funções com múltiplas entradas, deve-se avaliar seu gradiente, que é o conjunto de todas as suas derivadas parciais num vetor unidimensional.

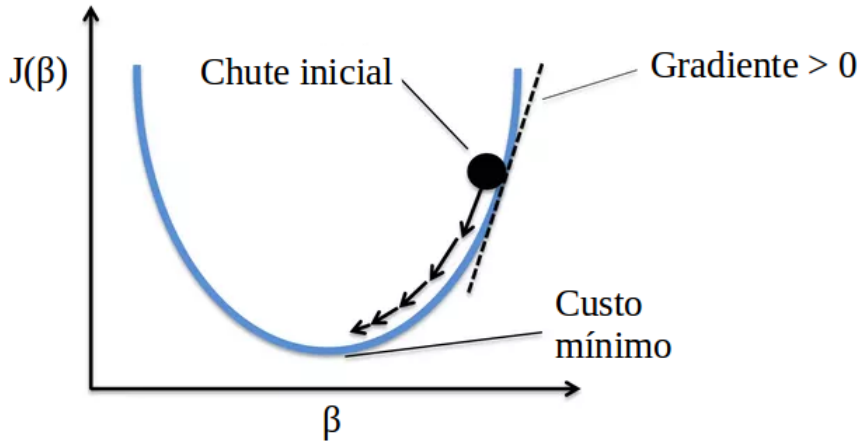
$$\nabla f = (\frac{\partial f}{x_1}, \ldots, \frac{\partial f}{x_i}). \tag{3.2.7}$$

Portanto, pode-se ajustar os parâmetros x, pela seguinte equação:

$$x' = x - n\nabla_x f(x), \tag{3.2.8}$$

onde: n corresponde à taxa de aprendizado, sendo um escalar positivo que determina o ajuste em cada iteração realizado pelo algoritmo.

Figura 3: Exemplificação da descida do gradiente.



Fonte: LEG/UFPR.

No caso de uma Rede Neural os pesos em cada camada interferem no valor da função de perda e estes são parâmetros que devem ser ajustados, por meio do processo descrito anteriormente. Entretanto, em um conjunto de dados massivo, isto levaria muito tempo para obter a solução. Uma alternativa com menor requisição de poder computacional é a Descida do Gradiente Estocástico, que deve atualizar os parâmetros com base em amostras do conjunto de treinamento.

As calibrações em cada iteração possuirá distorções da descida real, porém seu calculo é mais rápido e os resultados assemelham-se em muito com a inclinação real. Portanto, o desafio desta otimização, consiste na escolha apropriada do tamanho deste hiperparâmetro, como também da taxa de aprendizagem. Para mais detalhes sobre a Descida do Gradiente veja, (GOODFELLOW; BENGIO; COURVILLE, 2016).

3.2.2.2 Backpropagation

As Redes Neurais aprendem devido a um processo iterativo de ajuste de pesos oriundo do método da descida do gradiente. É sabido que as Redes Neurais são um conjunto de funções aninhadas, e para se efetuar os ajustes descritos anteriormente, faz-se

necessário o uso da regra da cadeia. Por exemplo, supondo que $y = g(x)$ e $z = f(g(x)) = f(y)$, pela regra da cadeia, tem-se

$$\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}. \tag{3.2.9}$$

Por meio da aplicação da regra da cadeia do cálculo, ajustes são efetuados a partir do final da rede e fluindo para o inicio, este processo é conhecido como o Backpropagation.

3.2.2.3 Momentum

Uma técnica para otimização do gradiente descendente é adicionar conceitualmente uma velocidade no gradiente através de uma média móvel exponencial. Esta média móvel é calculada, com base nos resultados dos gradientes anteriores e seu método é descrito nas tabela 2, retirada de Goodfellow, Bengio e Courville (2016, p.298).

Algorithm 8.2 Stochastic gradient descent (SGD) with momentum

Require: Learning rate ϵ, momentum parameter α.
Require: Initial parameter $\boldsymbol{\theta}$, initial velocity $\boldsymbol{v}$.
  while stopping criterion not met **do**
    Sample a minibatch of m examples from the training set $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ with corresponding targets $\boldsymbol{y}^{(i)}$.
    Compute gradient estimate: $\boldsymbol{g} \leftarrow \frac{1}{m}\nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)})$
    Compute velocity update: $\boldsymbol{v} \leftarrow \alpha\boldsymbol{v} - \epsilon\boldsymbol{g}$
    Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \boldsymbol{v}$
  end while

Fonte: Goodfellow, Bengio e Courville (2016).

Tabela 2: Representação da técnica de Momentum.

3.2.2.4 Transferência de aprendizado

Nas seções anteriores, descreveu-se como ocorre o aprendizado das Redes Neurais, porém quando estas redes possuem parâmetros de forma demasiada, os ajustes demandam muito tempo. Sendo assim, ao invés de inicializar os pesos da rede aleatoriamente, percebeu-se que em muitos casos obtinham-se uma convergência mais rápida utilizando de pesos pré-treinados, ou seja, advindos de outras redes.

Esta técnica é conhecida como transfer learning e existem diversas adaptações como treinar camadas especificas, adaptando a rede para seu uso.

3.2.3 Regularização

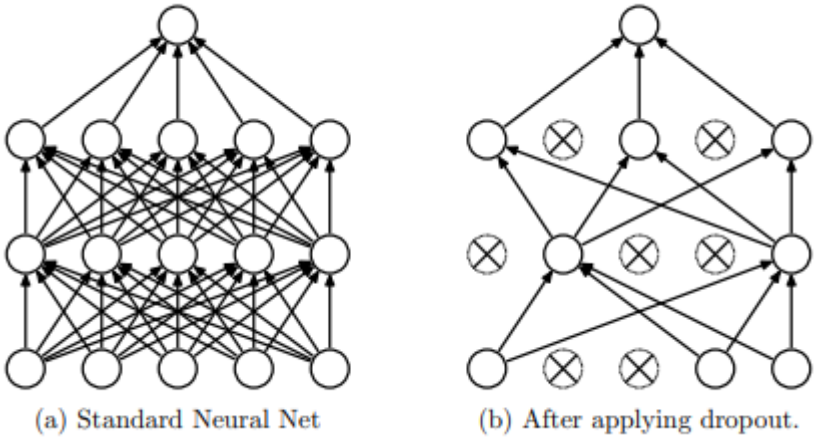
Regularização é qualquer modificação efetuada em um algoritmo de aprendizado, visando reduzir seu erro de generalização Goodfellow, Bengio e Courville (2016).

3.2.3.1 Dropout

Dropout é uma técnica comumente utilizada para regularização e tem como intuito evitar o overfitting, situação descrita na subseção 3.1.1.4. É descrita na literatura como uma das técnicas mais eficaz.

Consiste em inutilizar aleatoriamente alguns neurônios da rede, com intuito de reduzir-se o surgimento de padrões de causalidade (GOODFELLOW; BENGIO; COURVILLE, 2016).

Figura 4: Representação da técnica de dropout.

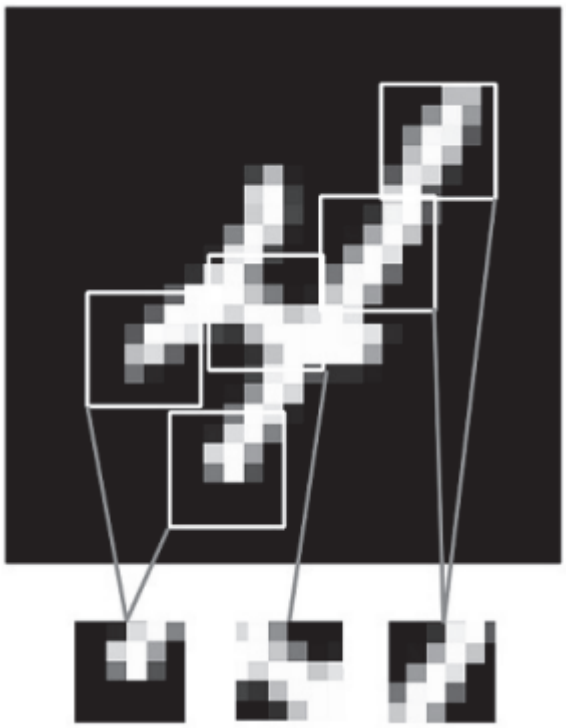


Fonte: Srivastava et al. (2014).

3.2.4 Redes Convolucionais

Uma Rede Neural Convolucional (CNN), possui uma estrutura mais apropriada para se analisar imagens, pois diferentemente de outras redes esta é suscetível a perceber alterações visuais de translação, rotação ou distorção, algo muito comum em imagens. Essas diferenças, devem-se as camadas de convolução, pois, diferentemente das camadas densamente conectadas, que analisam a imagem em seu conjunto total de pixeis, estas tem como característica aprender padrões locais e saber reconhece-los em qualquer região (CHOLLET, 2018). Na figura 5, é exposto o que seriam padrões locais utilizados na classificação.

Figura 5: Entendimento de padrões locais.

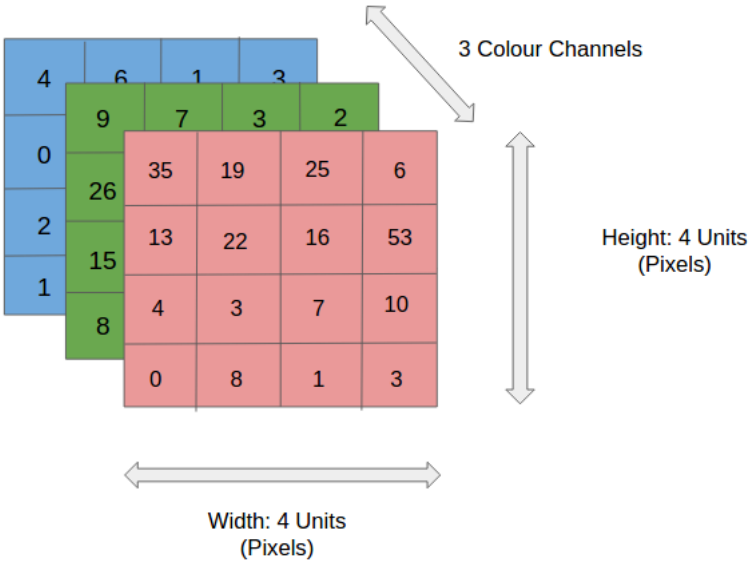


Fonte: Chollet (2018, p. 115).

3.2.4.1 Representação de imagens

Primeiramente, faz-se necessário compreender como um algoritmo é capaz de captar a informação de uma imagem. Em relação a estrutura de uma imagem, esta pode ser representada como uma coleção de matrizes, sendo uma para cada canal de cor. As matrizes são compostas por dois eixos: um para a altura e outro para a largura. Além disto, cada valor contido na matriz se refere a intensidade de sua coloração.

Figura 6: Representação de uma matriz RGB.

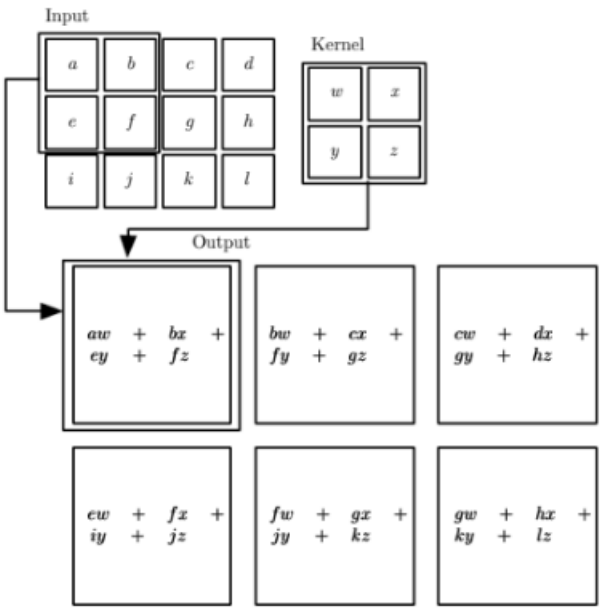


Fonte: Esteves (2020).

3.2.4.2 Camadas de Convolução

A camada convolucional é responsável por extrair ou realçar partes de uma imagem, por exemplo suas bordas, com o intuito de orientar a rede a focar nos padrões que sejam realmente de interesse. Isto ocorre, devido a uma operação linear que adiciona cada elemento da imagem aos seus vizinhos locais, ponderados por um kernel. Na figura 7 é exposto esta operação.

Figura 7: Convolução aplicada a uma imagem.



Fonte: Goodfellow, Bengio e Courville (2016).

Matematicamente a convolução supracitada pode ser descrita por:

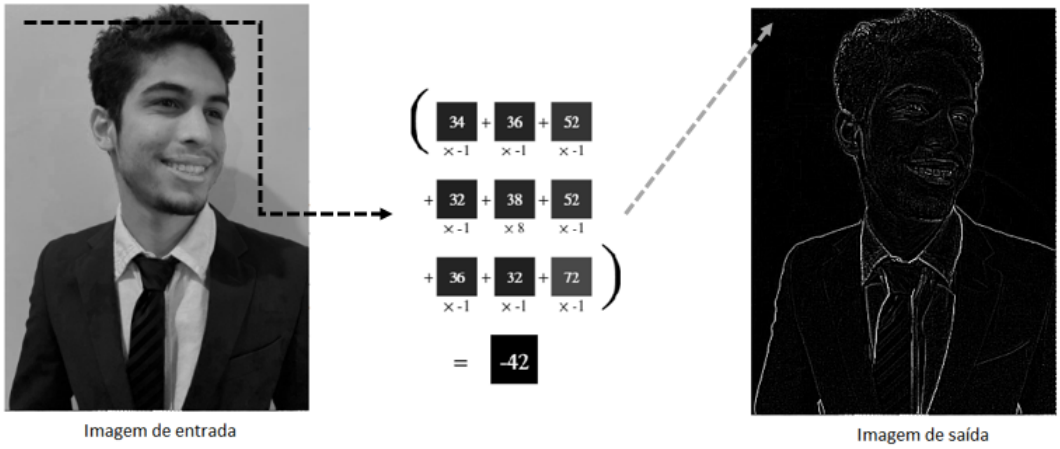
$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n), \tag{3.2.10}$$

Onde:

- I é a matriz da imagem de entrada;
- K é uma matriz de duas dimensões, também chamada de Kernel com dimensões mxn;
- S é a imagem resultante do deslocamento deste Kernel sobre a imagem de entrada.

Na subseção 3.2.4.1, foi explicado que uma imagem pode ser representada por matrizes, porém, a operações descrita não é a multiplicação usual da álgebra linear. Exemplificando, na figura 8 observa-se uma imagem com apenas um canal de cor, isto é, escala de cinza. Nesta, será realizado uma convolução a partir do Kernel exposto de tamanho $3x3$, ou seja, os noves pixeis no inicio da primeira seta serão multiplicados pelo valor correspondente no Kernel e somados. Adiante, destes pixeis que encontram-se na primeira imagem, a soma resultante de -42 será atribuído ao pixel da imagem de saída. Esta operação percorreu por toda a dimensão da imagem de entrada, realçando as linhas de borda.

Figura 8: Convolução aplicada a uma imagem.



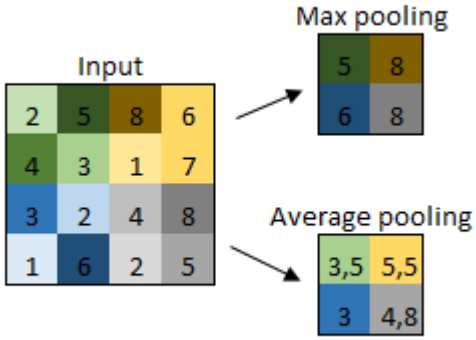
Fonte: Elaborada pelo autor.

3.2.4.3 Camada de Pooling

A técnica de pooling é comumente utilizada após a camada de convolução, tendo por finalidade a redução da dimensionalidade da imagem, preservando suas características.

Na literatura dois tipos de pooling são amplamente utilizadões, sendo estes o max pooling e o average pooling. A redução de dimensionalidade, esta diretamente relacionada a escolha do tamanho do filtro, e a distancia deste deslocamento, também conhecido como stride. Na figura 9, é mostrado as duas operações de pooling citadas, com base em um filtro de dimensões $2x2$ e um stride = 2.

Figura 9: Representação das técnicas de pooling mencionadas.



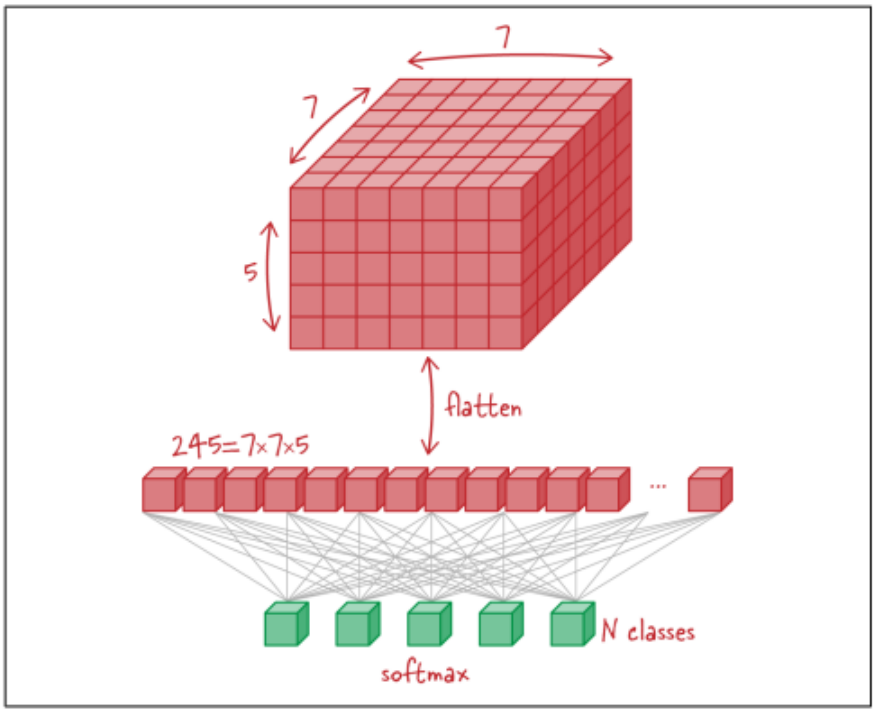
Fonte: Elaborada pelo autor.

Ainda analisando a figura 9, observa-se que dos 4 pixeis superiores do input, selecionou-se apenas o pixel de maior valor na operação de max poling, seguido de um deslocamento de dois quadrados para a direita, onde seleciono-se novamente o pixel de maior valor. Este processo originou uma imagem $4x$ menor do que a original e preservou as informações expostas. O mesmo processo se aplica no caso do average pooling, porém o pixel resultante se baseia na média dos valores.

3.2.4.4 Flattening

Flattening é uma operação que consiste, basicamente, em rearranjar os dados das matrizes, advindas das operações de convolução e pooling, em um vetor unidimensional, em seguida conecta cada informação a todos os neurônios da camada de saída, como mostrado na figura 10.

Figura 10: Operação de flattening.

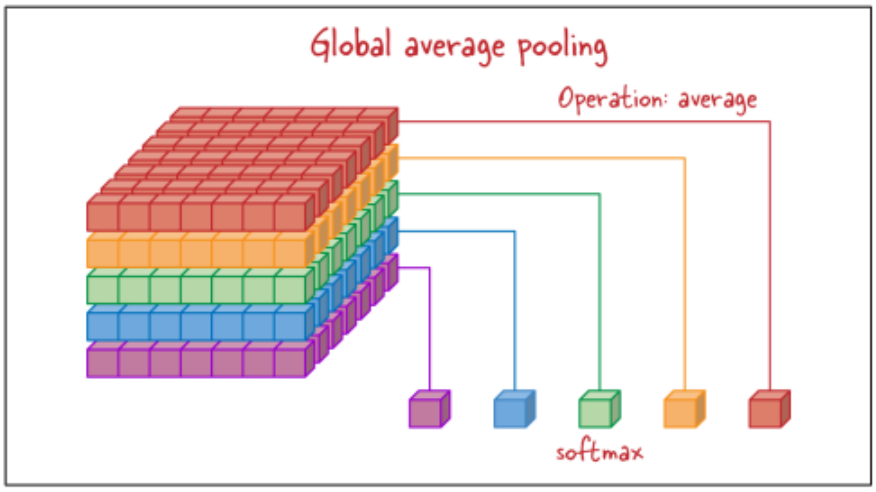


Fonte:Lakshmanan, Görner e Gillard (2021, p. 86) .

3.2.4.5 Global Average Pooling

Global Average Pooling é uma operação projetada para substituir a operação de flattening, demandando menor requisição computacional, pois este condessa a informação em vetores e os classifica diretamente. Além disto, possui vantagens em problemas que não necessitem de informações posicionais (LAKSHMANAN; GÖRNER; GILLARD, 2021).

Figura 11: Representação da operação de Global Average Pooling.



Fonte: Lakshmanan, Görner e Gillard (2021, p. 87).

4 Metodologia

4.1 Conjunto de dados

O conjunto de dados utilizado para treinamento, validação e teste do modelo será o ChestX-ray, de propriedade do National Institutes of Health. Este conjunto armazena 112.120 imagens de raios-X de visão frontal pertencentes a 30.805 pacientes únicos. Para cada uma destas imagens, foram atribuídos rótulos de quatorze tipos de patologias torácicas, sendo estas: Atelectasia, Consolidação, Infiltração, Pneumotórax, Edema, Enfisema, Fibrose, Efusão, Pneumonia, Espessamento Pleural, Cardiomegalia, Nódulo, Massa e Hérnia. Mais detalhes sobre a coleta e rotulação das imagens são relatado em (WANG et al., 2017).

4.2 Preparação dos dados

A etapa de preparação dos dados, que envolve coleta, limpeza e particionamento das imagens, foi dividida em duas fases. Sendo a primeira composta por coletar os dados e efetuar a identificação dos respectivos rótulos pertencentes a cada imagem. Após isto, iniciou-se a segunda fase que consistiu na exclusão de imagens com mais de uma classificação patológica, por ultimo, foi efetuado o particionamento dos dados em bases de treinamento, teste e validação.

4.2.1 Primeira fase

Primeiramente, efetuou-se o download do repositório dos dados, que são cerca de 45,7 gigabytes, compreendendo tanto imagens como arquivos informacionais relativo ao conjunto de dados. Abaixo é exposto uma tabela com algumas das informações disponíveis, além do conjunto de imagens.

- Nome do arquivo de imagem;
- Patologias classificadas;
- Número da consulta;
- Identificação do paciente;
- Idade do paciente;
- Sexo do paciente.

4.2.2 Segunda fase

Após efetuado a identificação dos caminhos de cada imagem e patologia acometida, realizou-se uma exclusão das imagens que possuía mais de um rótulo, pois foge aos objetivos elencados neste relatório. Por conseguinte, das 112.120 imagens iniciais restou-se 91.324, sendo 60.361 não pertencentes a nenhuma das patologias, ou seja, imagens sem alterações. Dispondo deste novo conjunto de dados, particionou-se de forma aleatória em bases de treinamento, teste e validação, dividindo-os em 70%, 15% e 15% das imagens, respectivamente.

Tabela 3: Quantitativo de imagens por base de dados.

Base	Quantidade de imagens
Treinamento	63.926
Teste	13.699
Validação	13.699

4.3 Análise exploratória

O banco de dados é composto por 39.981 imagens referentes ao sexo feminino e 51.343 do sexo masculino. Em relação a idade dos pacientes, percebe-se uma distribuição de uma população adulta, já que, os dados apresentam uma curva em formato de sino e média de aproximadamente 46 anos como podemos ver no histograma abaixo e corroborado pelas métricas descritivas que mostram que 50% do dados são superiores aos 34 anos e menores de 58 anos.

Figura 12: Histograma da idade.

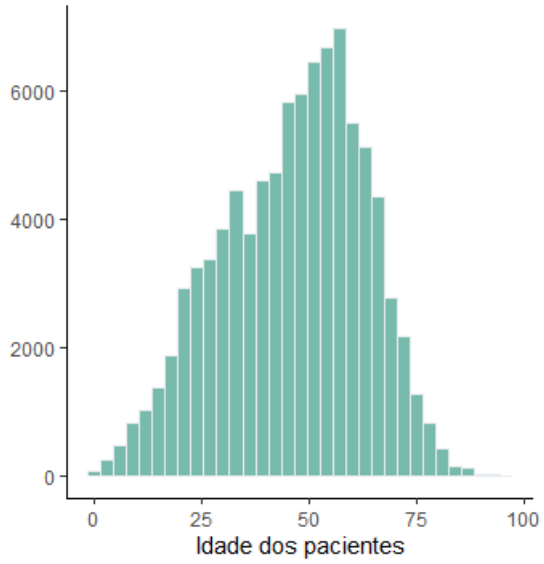
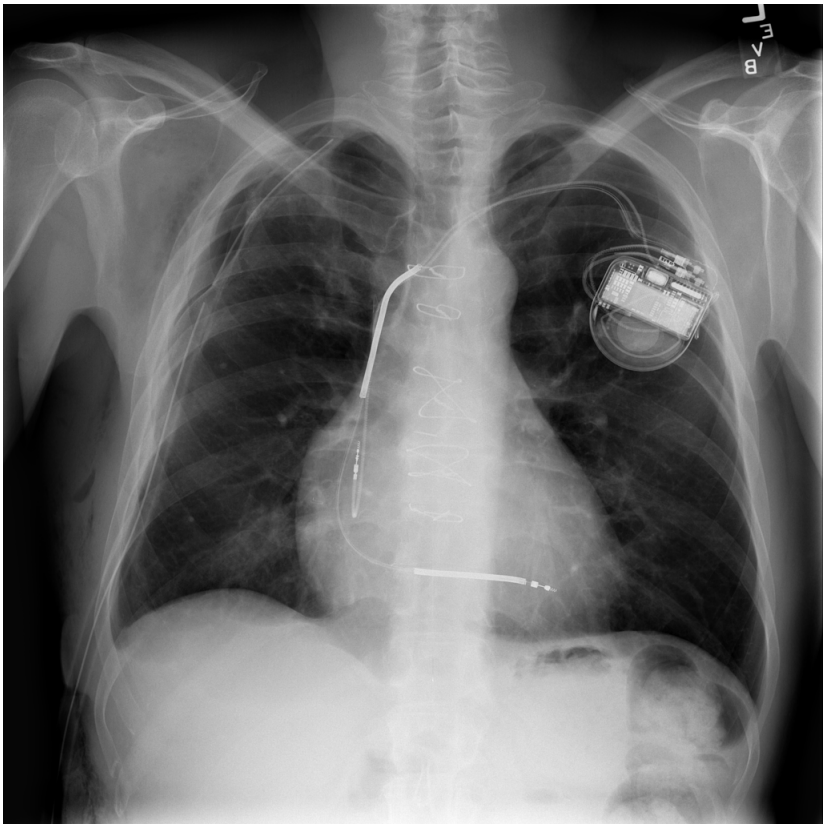


Tabela 4: Medidas descritivas das idades dos pacientes.

Medida descritiva	Valor
Valor mínimo	0
Primeiro quartil	34
Média	46,25
Mediana	48,00
Terceiro quartil	58
Valor máximo	95

Faz de suma importância mencionar, estas analises pois devido a variabilidade de gênero e idade as caixas torácicas presentes no banco de dados apresentarão tamanhos diferentes. Além disso, existe a presença de inúmeras imagens, cujo o paciente apresenta objetos em seu tórax, por exemplo como observa-se na figura 13. Vale salientar, que estas alterações podem acarretar em classificações erradas.

Figura 13: Raio-x de objeto visível na classe sem alteração patológica.



Fonte: Imagem de número 00000013_038.png.

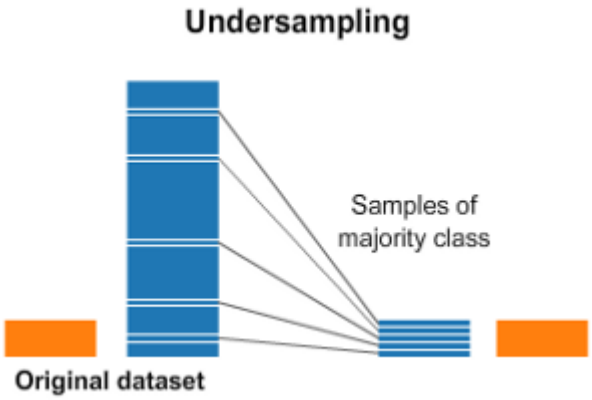
Na tabela abaixo, observa-se que a aleatorização da base em conjuntos de treino, teste e validação ocorreu com êxito, já que as patologias foram particionadas de forma proporcional. Entretanto, evidenciou-se um desbalanceamento dos dados entre as classes, desta maneira, quando necessário foi realizado o balanceamento das imagens equiparando-

as pelo conjunto com menor registro, esta técnica é conhecida como undersampling.

Tabela 5: Variabilidade no conjunto de dados.

Classificação	Base de treino	Base de teste	Base de validação
Atelectasia	2.935 (4,59%)	617 (4,50%)	663 (4,84%)
Cardiomegalia	759 (1,19%)	166 (1,21%)	168 (1,23%)
Consolidação	921 (1,44%)	186 (1,36%)	203 (1,48%)
Efusão	2.769 (4,33%)	604 (4,41%)	582 (4,25%)
Edema	417 (0,65%)	95 (0,69%)	116 (0,85%)
Enfisema	606 (0,95%)	141 (1,03%)	145 (1,06%)
Fibrose	494 (0,77%)	117 (0,85%)	116 (0,85%)
Hérnia	70 (0,11%)	17 (0,12%)	23 (0,17%)
Infiltração	6.705 (10,49%)	1.437 (10,49%)	1.405 (10,26%)
Massa	1.560 (2,44%)	286 (2,09%)	293 (2,14%)
Nodulo	1.909 (2,99%)	420 (3,07%)	376 (2,74%)
Pneumonia	230 (0,36%)	46 (0,34%)	46 (0,34%)
Pneumothorax	1.531 (2,39%)	336 (2,45%)	327 (2,39%)
Espessamento Pleural	795 (1,24%)	164 (1,20%)	167 (1,22%)
Sem Alteração	42.225 (66,06%)	9.069 (66,19%)	9.067 66,18%
TOTAL	100,00%	100,00%	100,00%

Figura 14: Técnica de balanceamento.



4.4 Modelagem

Após a preparação dos dados, ocorreu a fase de modelagem que consistiu na aplicação prática de modelos. E com o propósito abordar todas as técnicas mencionadas na revisão de literatura, foi escolhido um modelo com alto desempenho em extração de features para realizarmos a transferência de aprendizado. Este modelo foi o Very Deep

Convolutional Networks for Large Scale Image Recognition, mais conhecido como VGG19.

Como visto, é de praxe utilizar de um modelo pré-treinado, devido a sua capacidade em extrair features, porém devido aos inúmeros parâmetros presente neste modelo, o treinamento de toda as camadas exigiria uma alta demanda de poder computacional e tempo disponível. Desta maneira, optou-se por não retreinar as camadas convolucionais e verificando seu desempenho entre as classes com modelos igualmente ajustado.

Neste caso, após a extração das features por parte da VGG19, esta rede é acrescida de uma camada de Global Average Pooling, responsável por conectar essas informações a uma camada densa com 2048 neurônios. Com o propósito de se evitar o overfitting, inclui-se um dropout de 60% entre esta camada e a de saída, que contem dois neurônios com função de ativação Softmax.

A CNN do presente relatório foi configurada, com base em hiperparâmetros apresentados como usuais no livro de Chollet (2018). Sendo assim, para o otimizador de descida de gradiente estocástico, configurou-se um momentum igual a 0,90 com uma taxa de aprendizagem de 0,001. Em relação a função de perda, optou-se por utilizar a "categorical_crossentropy", cujo valor retornado é uma probabilidade de pertencer a uma das classes em análise e funciona muito bem com a função de ativação escolhida.

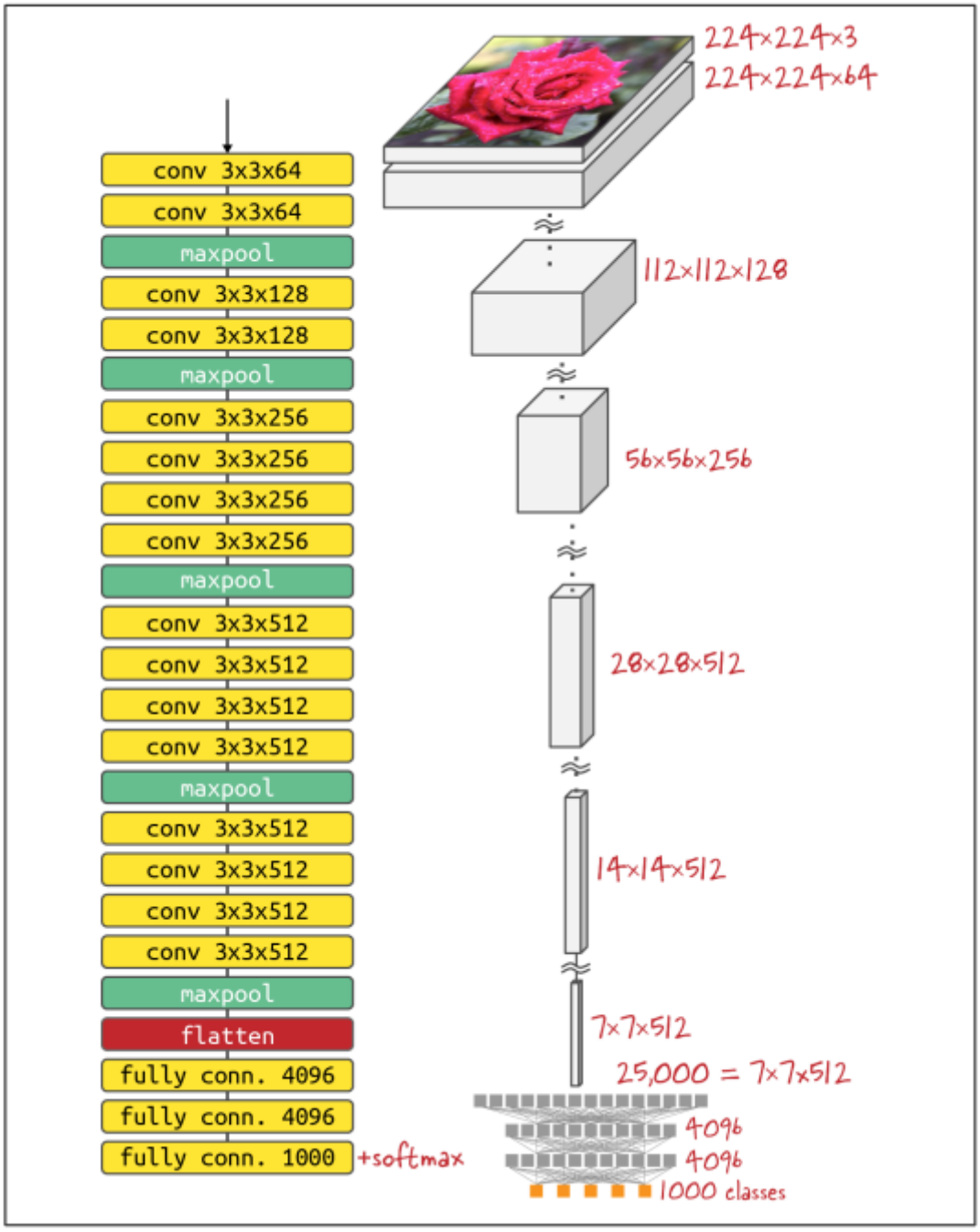
O processo de treinamento planejado para cada modelo foi de 100 épocas, porém foi configurado funções a fim de monitorar e realizar operações quando necessário. Estas funções são conhecidas como callbacks, sendo estas:

- early_stop: deve monitorar o loss no conjunto de validação e caso este mantenha-se com uma mudança inferior de 0,001 em 15 épocas seguidas o treinamento se encerra;
- reduce_lr: consiste em multiplicar a taxa de aprendizado por 0.1, caso em 12 épocas o loss no conjunto de validação não diminuísse;
- checkpoint: deve salvar apenas o melhor modelo com base no menor loss aferido no conjunto de validação.

4.4.1 Arquitetura da VGG19

Como foi de interesse apenas utilizar das camadas convolucionais da rede pré-treinada, segue abaixo a figura 15, esboçando sua estrutura geral. Observa-se o tipo de camada, tamanho dos filtros e o formato de saída. Em suma, esta rede possui 16 camadas de convolução com filtros de tamanho $3x3$ e 20.024.384 parâmetros abstendo-se das camadas da rede densa original. Na tabela 6, expõem-se a quantidade de parâmetros da rede arquitetada para o problema de classificação abordado.

Figura 15: Arquitetura da VGG19.



Fonte: Lakshmanan, Görner e Gillard (2021)

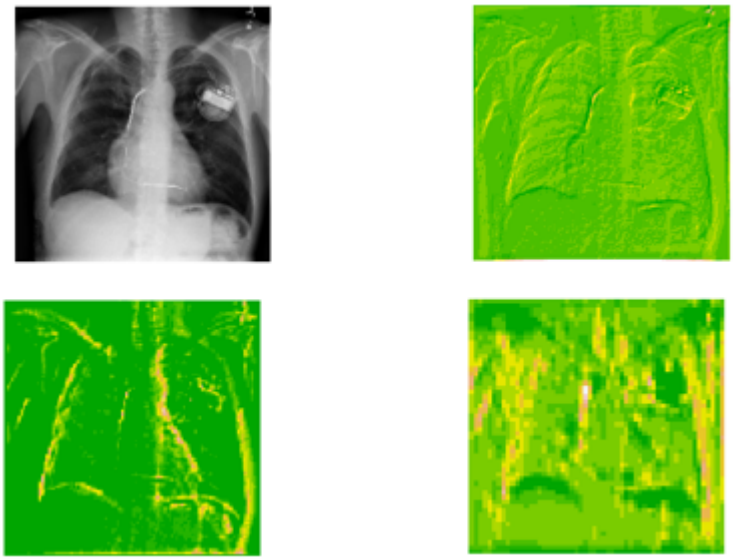
Tabela 6: Resumo da quantidade de parâmetros na rede completa.

Total de parâmetros	21.079.106
Parâmetros treináveis	1.054.722
Parâmetros não treináveis	20.024.384

4.4.2 Visão da VGG19

Após a imagem ser carregada para o modelo, esta é submetidas a diversas convoluções, gerando cada vez mais abstrações visuais dos dados de entrada como vimos na figura 8. Entretanto, vale destacar que quase sempre estas imagens não possuirão algum significado prático para os humanos. Sendo assim, foram selecionados três filtros de camadas de convolução diferentes da VGG19 e aplicou-se na primeira imagem, apenas para fins de elucidar as abstrações extraídas pela rede faz.

Figura 16: Convoluções da VGG19.



Fonte: Imagem de número 00000013_038.png.

5 Resultados

Esta seção apresentará os resultados obtidos no presente relatório, descrevendo o desempenho do modelo em cada classe. Por conseguinte, expõem-se na tabela 7, as métricas de acurácia, precisão e loss, aferidas para cada classe. Em relação a acurácia, 10 classes encontram-se superior a 60% e duas acima de 70%, sendo assim, é factível dizer que os modelos obtiveram uma performance satisfatória.

Tabela 7: Métricas de desempenho por classe.

Classe	Loss	Acurácia	Precisão
Atelectasia	0,6267	0,6467	0,6365
Cardiomegalia	0,6353	0,6235	0,5972
Consolidação	0,6102	0,6694	0,6583
Efusão	0,5669	0,7119	0,7126
Edema	0,4893	0,7895	0,7723
Enfisema	0,6334	0,6667	0,6715
Fibrose	0,6371	0,6368	0,6176
Hérnia	0,6673	0,5882	0,6250
Infiltração	0,6690	0,5835	0,5896
Massa	0,6654	0,6224	0,6378
Nodulo	0,6762	0,5833	0,5936
Pneumonia	0,6824	0,5652	0,5536
Pneumothorax	0,6250	0,6503	0,6526
Espessamento Pleural	0,6576	0,6098	0,6084

Adiante, são exibidas todas as matrizes de confusão, uma para cada classe, cujo é possível observar ótimos desempenhos em três classes: Efusão, Edema e Enfisema. Não obstante, observa-se também os quatro piores casos: Hernia, Infiltração, Pneumonia e Nódulo.

Tabela 8: Matriz de confusão para Atelectasia.

	Valor real	
Valor predito	Sem alteração	Alterado
Sem alteração	376	195
Alterado	241	422

Tabela 9: Matriz de confusão para Consolidação.

	Valor real	
Valor predito	Sem alteração	Alterado
Sem alteração	118	55
Alterado	68	131

Tabela 10: Matriz de confusão para Infiltração.

	Valor real	
Valor predito	Sem alteração	Alterado
Sem alteração	887	647
Alterado	550	790

Tabela 11: Matriz de confusão para Pneumotórax.

	Valor real	
Valor predito	Sem alteração	Alterado
Sem alteração	221	120
Alterado	115	216

Tabela 12: Matriz de confusão para a classe Edema.

	Valor real	
Valor predito	Sem alteração	Alterado
Sem alteração	72	17
Alterado	23	78

Tabela 13: Matriz de confusão para Enfisema.

	Valor real	
Valor predito	Sem alteração	Alterado
Sem alteração	96	49
Alterado	45	92

Tabela 14: Matriz de confusão para Fibrose.

	Valor real	
Valor predito	Sem alteração	Alterado
Sem alteração	65	33
Alterado	52	84

Tabela 15: Matriz de confusão para Efusão.

	Valor real	
Valor predito	Sem alteração	Alterado
Sem alteração	431	175
Alterado	173	429

Tabela 16: Matriz de confusão para Pneumonia.

	Valor real	
Valor predito	Sem alteração	Alterado
Sem alteração	21	15
Alterado	25	31

Tabela 17: Matriz de confusão para Espessamento Pleural.

	Valor real	
Valor predito	Sem alteração	Alterado
Sem alteração	99	63
Alterado	65	101

Tabela 18: Matriz de confusão para Cardiomegalia.

	Valor real	
Valor predito	Sem alteração	Alterado
Sem alteração	81	40
Alterado	85	126

Tabela 19: Matriz de confusão para Nódulo.

	Valor real	
Valor predito	Sem alteração	Alterado
Sem alteração	268	198
Alterado	152	222

Tabela 20: Matriz de confusão para Massa.

	Valor real	
Valor predito	Sem alteração	Alterado
Sem alteração	194	124
Alterado	92	162

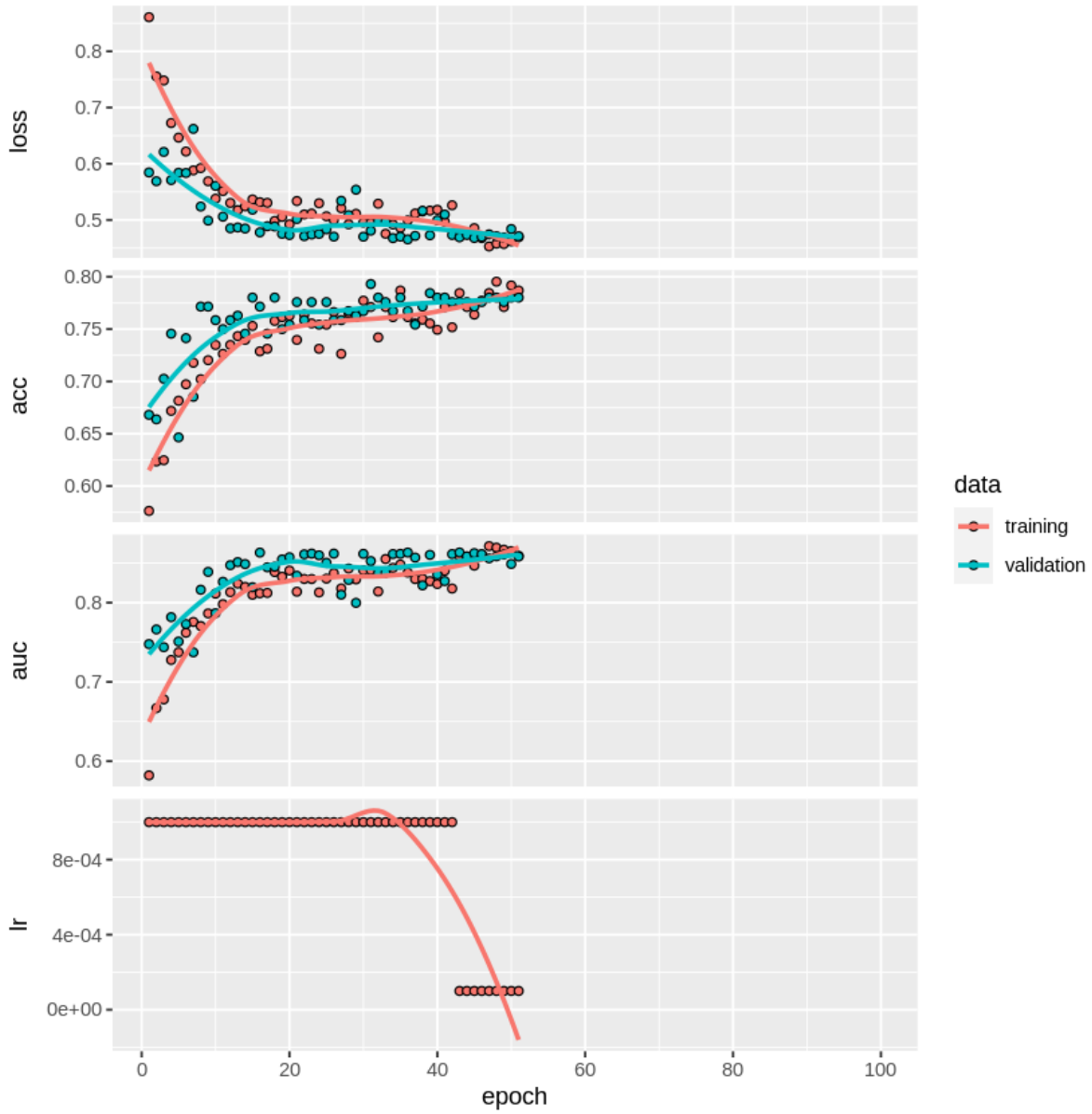
Tabela 21: Matriz de confusão para Hérnia.

	Valor real	
Valor predito	Sem alteração	Alterado
Sem alteração	8	5
Alterado	9	15

Reiterando, além de Edema possuir as melhores métricas de desempenho, seu treinamento precisou de apenas 51 épocas para alcançar estes resultados. Vale ressaltar, que seu conjunto de imagens para treino é um dos menores, contendo apenas 417 imagens. Desta maneira, fica evidente que a rede tornou-se capaz em reconhecer algum padrão característico.

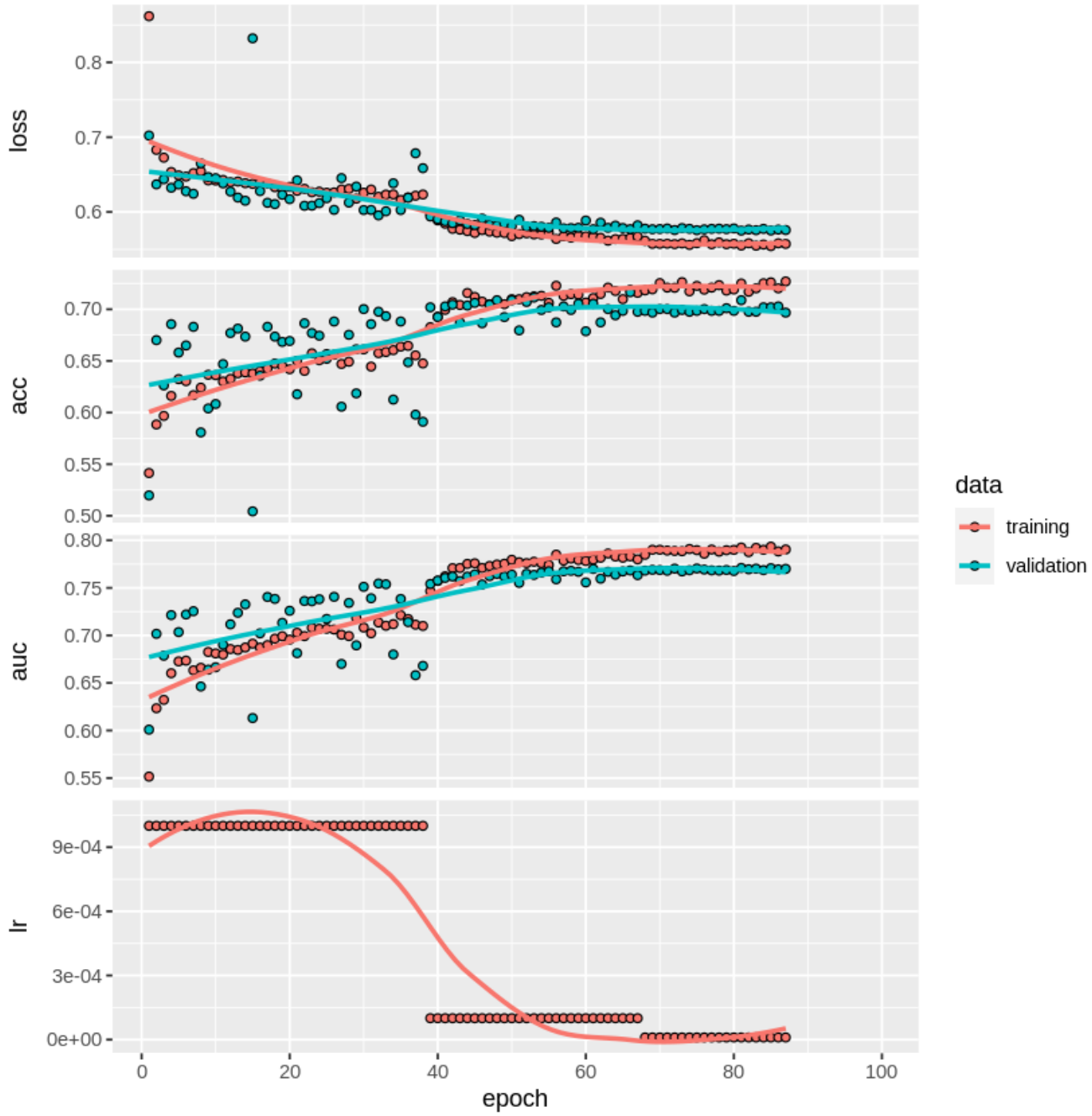
Observa-se no gráfico 17, que tanto no conjunto de treinamento quanto no de validação o loss vai decaindo a cada iteração, bem como a acurácia de treinamento aumenta. Este é o comportamento esperado para um modelo com boa capacidade preditiva, corroborado pelos mais de 78% aferidos no conjunto de teste.

Figura 17: Treinamento da rede Edema.



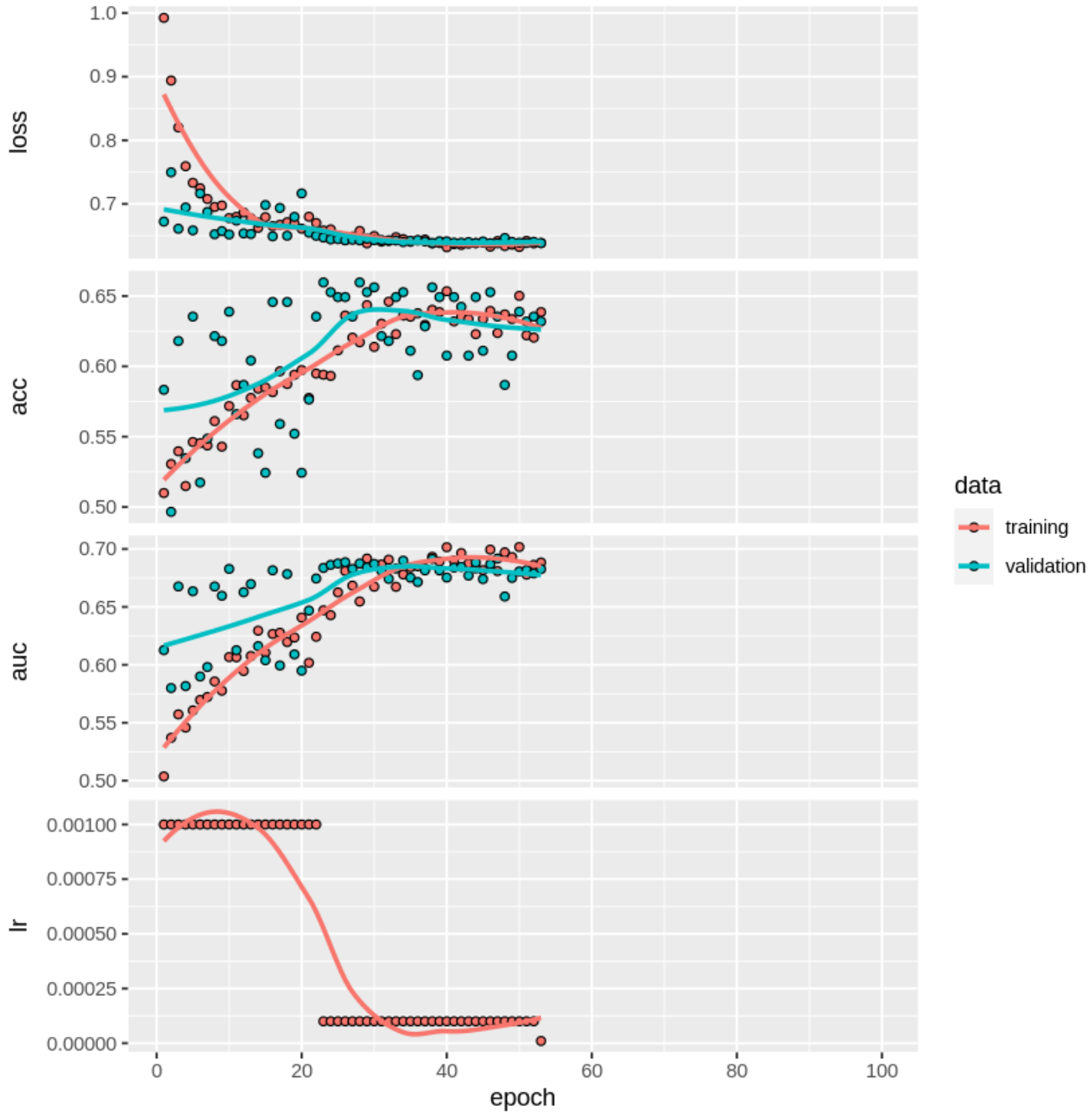
Nota-se no gráfico 18, que diferentemente da classe Edema que convergiu nas primeiras épocas para um ótimo resultado, esta classe teve um aprendizado amortizado, ou seja, o número de épocas foram fatores essenciais para se alcançar uma acurácia e precisão com mais de 70%. Em relação as imagens de treinamento, este é o terceiro maior conjunto de imagens, possuindo até $6x$ mais quando comparado a edema.

Figura 18: Treinamento da rede Efusion.



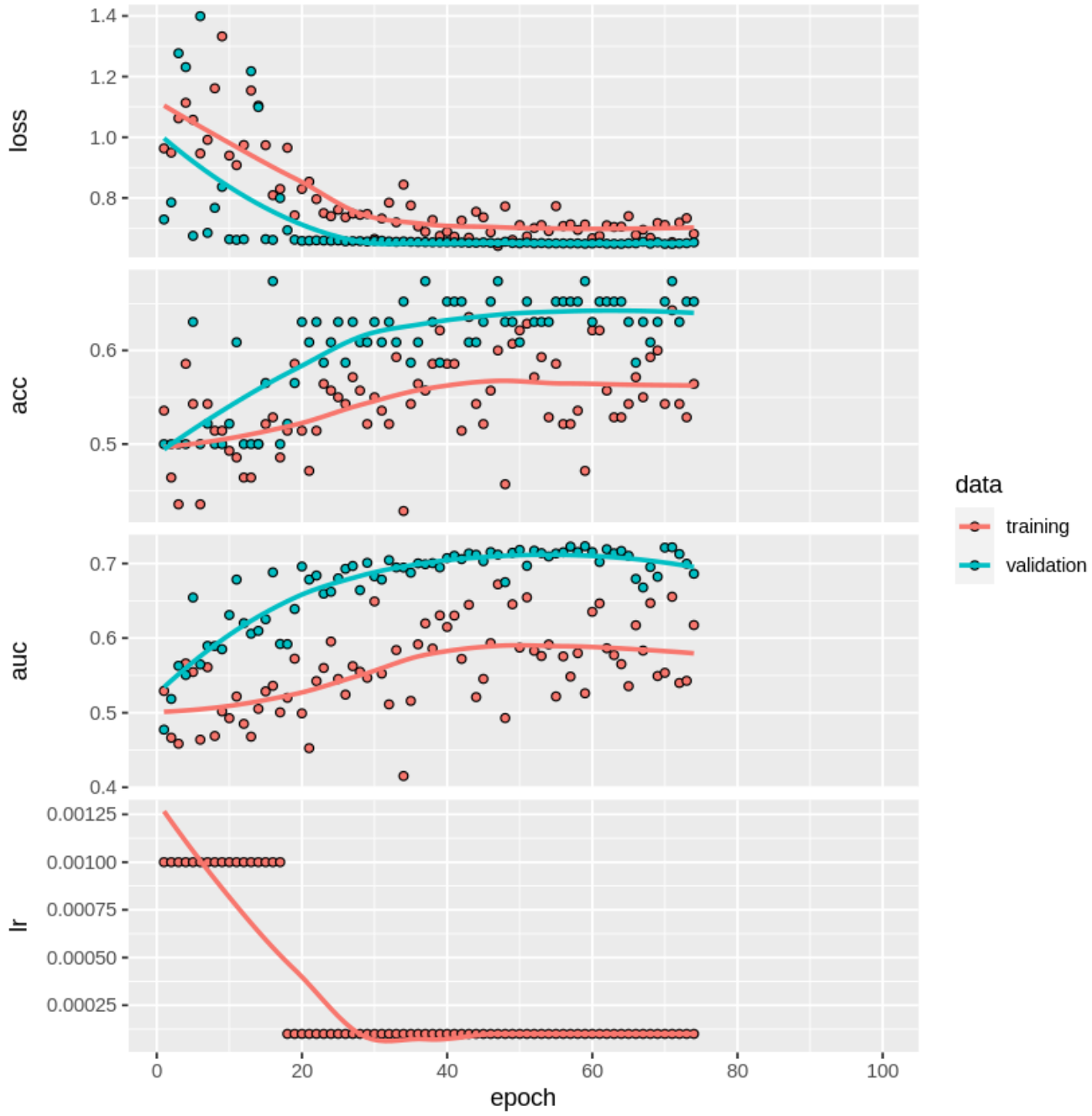
No gráfico 19, percebe-se um comportamento abrupto tanto nas métricas de desempenho quanto na taxa de aprendizagem. Na seção 4.4 definiu-se que todos os modelos seriam configurados igualmente, entretanto, como inicialmente se teve um loss bem baixo, as variações percentuais também serão baixas, logo, inicializou-se o callback que diminui a taxa de aprendizagem. Portanto, não somente neste caso, mas se para todos os modelos, fossem escolhidos hiperparâmetros mais específicos para seu treinamento, estes poderiam ter resultados ainda mais satisfatórios.

Figura 19: Treinamento da rede Enfisema.



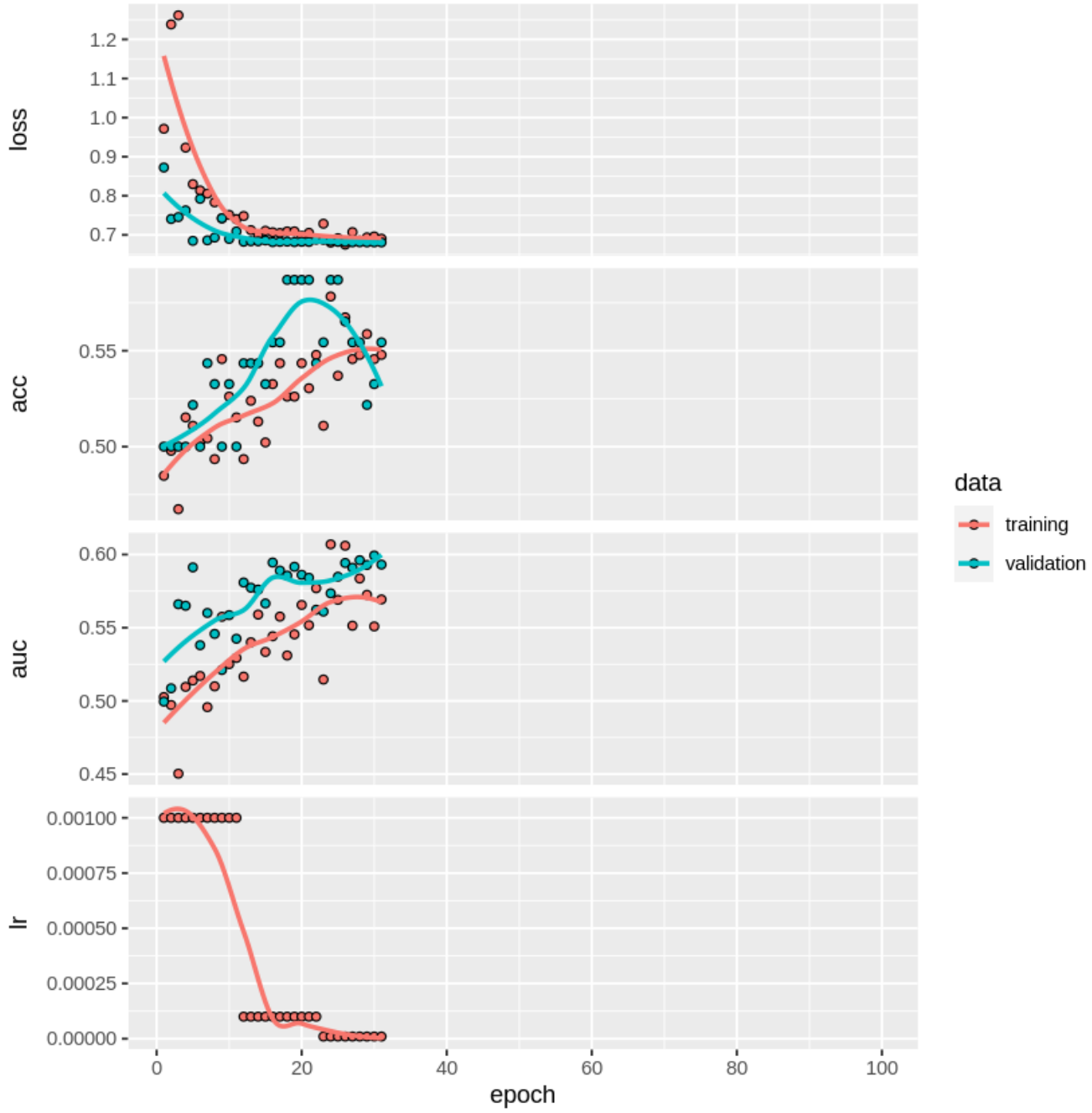
Curiosamente, no treinamento referente a Hérnia percebe-se que a rede conseguiu generalizar mais seu aprendizado no conjunto de validação do que no de treino. Um dos fatores de maior impacto, que influenciaram diretamente o aprendizado desta rede, foi que seu conjunto de treino possuía apenas 70 imagens. Por conseguinte, o melhor resultado apurado no conjunto de validação, pode ser, devido a um particionamento menos ruidoso para validação.

Figura 20: Treinamento da rede Hernia.



No treinamento da classe referente a pneumonia, observou-se um early stopping na 31 época devido a invariação do loss e semelhantemente ao supracitado, esta rede também foi prejudicada devido a limitações de imagens e modelo selecionado.

Figura 21: Treinamento da rede Pneumonia.



6 Conclusão e Trabalhos Futuros

Este relatório propôs a utilização de modelos de deep learning para ajudar a identificar doenças pulmonares a partir de radiografias de tórax. Realizando a modelagem de dados com base na transferência de aprendizado de uma CNN pré-treinada, neste caso a VGG19. Na seção de resultados, percebeu-se que a Rede Neural Convolucional desenvolvida a partir da rede pré-treinada e com camadas densas iguais, apresentou resultados positivos quanto à classificação de alguns rótulos, enquanto outros não foram satisfatórios. Isto, deve-se a configuração inflexível configurada.

É importante destacar que para os resultados não satisfatórios, existem na literatura técnicas que resultariam em melhores resultados, as quais não foram exploradas neste relatório.

De acordo com a descrição teórica detalhada, entende-se que as Redes Neurais Convolucionais são capazes de abranger uma quantidade bem superior de features através do acréscimo de filtros, porém isso resultaria numa quantidade maior de parâmetros, necessitando de maior requisição computacional em seu treino. Contudo, resultando numa melhor classificação das imagens, em virtude de sua arquitetura e relacionamentos mais complexos entre os dados e a variável resposta.

Destarte, observou-se que é possível criar ótimos modelos de classificação, como o aferido na classificação de Edema, que teve como êxito a geração de um modelo com acurácia superior a 78%. Sendo assim, em posse de modelos mais acurados, conclui-se que seu uso seria de grande valia no apoio de diagnósticos.

Em trabalhos futuros seria interessante avaliar o desempenho de modelos mais complexos e computacionalmente mais custosos aliados a inserção de variáveis como histórico do paciente. Utilizando de cluster em nuvem para otimização do treinamento e assim verificar tanto o ganho computacional como a acuracidade de novos modelos. Por fim, a questão da reprodutibilidade foi considerada ao longo do desenvolvimento dos modelos, e para permitir a análise e reprodução fiel dos resultados descritos, disponibilizou-se no apêndice os códigos implementados.

Referências

BRANCO, H. *Overfitting e underfitting em Machine Learning*. 2020. Acessado em: 07.03.2022. Disponível em: ⟨https://abracd.org/overfitting-e-underfitting-em-machine-learning/⟩.

CHOLLET, F. *Deep learning with R / Francois Chollet ; with J.J. Allaire*. Shelter Island, New York: Manning Publications, 2018. ISBN 1-63835-163-5.

ESTEVES, T. *Agrupando conceitos e classificando imagens com Deep Learning*. 2020. Acessado em: 07.03.2022. Disponível em: ⟨https://estevestoni.medium.com/agrupando-conceitos-e-classificando-imagens-com-deep-learning-5b2674f99539⟩.

GOODFELLOW, I. J.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016. ⟨http://www.deeplearningbook.org⟩.

GROUP, W. H. O. P. V. T. I.; ORGANIZATION, W. H. *Standardization of interpretation of chest radiographs for the diagnosis of pneumonia in children / World Health Organization Pneumonia Vaccine Trial Investigators' Group*. [S.l.]: World Health Organization, 2001. WHO/VB/01.35 p.

IGNOTOFSKY, R. *As Cientistas. 50 Mulheres que Mudaram o Mundo*. [S.l.: s.n.], 2017.

LAKSHMANAN, V.; GÖRNER, M.; GILLARD, R. *Practical Machine Learning for Computer Vision*. O'Reilly Media, 2021. ISBN 9781098102333. Disponível em: ⟨https://books.google.com.br/books?id=8ng5EAAAQBAJ⟩.

LEG/UFPR. *Laboratório de Estatística e Geoinformação*. Acessado em: 07.03.2022. Disponível em: ⟨http://cursos.leg.ufpr.br/ML4all/apoio/Gradiente.html⟩.

MOYER, V. A.; KENNEDY, K. A. Understanding and using diagnostic tests. *Clinics in Perinatology*, W.B. Saunders, v. 30, p. 189–204, 2003. ISSN 00955108.

POTCHEN, E. J. Measuring observer performance in chest radiology: Some experiences. *Journal of the American College of Radiology*, Elsevier, v. 3, p. 423–432, 2006. ISSN 15461440.

SAMEI, E. Why medical image perception? *Journal of the American College of Radiology*, Elsevier, v. 3, p. 400–401, 2006. ISSN 15461440.

SRIVASTAVA, N. et al. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, v. 15, n. 56, p. 1929–1958, 2014. Disponível em: ⟨http://jmlr.org/papers/v15/srivastava14a.html⟩.

WANG, X. et al. Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition(CVPR)*. [S.l.: s.n.], 2017. p. 3462–3471.

7 Apêndice

7.1 Análise descritiva

```r
### Bibliotecas dependentes:

if (!require(pacman)) {
  install.packages("pacman")
  library(pacman)
}
pacman::p_load(tidyverse)

### Leitura dos dicionario dos dados:

dici <- read.csv("./dados/CXR8/Data_Entry_2017_v2020.csv")
glimpse(dici)

### Feature Engineering:
#### Transformando a variavel FD em respostas dummy:

rotulo <- c(
  "Atelectasis",
  "Cardiomegaly",
  "Consolidation",
  "Effusion",
  "Edema",
  "Emphysema",
  "Fibrosis",
  "Hernia",
  "Infiltration",
  "Mass",
  "Nodule",
  "Pneumonia",
  "Pneumothorax",
  "Pleural_Thickening"
)

rotulo_vars <- rotulo %>%
```

```
  purrr::map(~ expr(ifelse(data.table::like(Finding.Labels, !!.x), 1, 0))) %>%
  purrr::set_names(paste0("rotulo_", gsub("\\s|/", "", rotulo)))

dici <- mutate(dici,!!!rotulo_vars)

#save(dici,file = "dici.rdata")

#### Ocorrencia dos rotulo:

dici %>% select(starts_with("rotulo")) %>%
  summarise_all( ~ sum(as.integer(.))) %>%
  pivot_longer(cols = starts_with("rotulo"),
               names_to = "Patologia",
               values_to = "freq") %>%
  mutate(frac = paste0(round(`freq` / sum(`freq`) * 100, 1), "%")) %>%
  arrange(desc(freq))

#### Multiplicidade dos rotulo:

dici  %>% select(starts_with("rotulo")) %>%
  transmute(sumVar = rowSums(.)) %>% bind_cols(dici, .) %>%
  pull(sumVar) %>% table()
```

7.2 Preparação dos dados

```
### Bibliotecas dependentes:

if (!require(pacman)) {
  install.packages("pacman")
  library(pacman)
}

p_load(tidyverse)


### Carregando dicionario da análise

load("dici.rdata")
```

```r


### Identificando local das imagens

A ideia deste código é identificar o caminho de todas
imagens, já que estas estão em pastas diferentes e o
dicionario criado na análise descritiva não traz esta
informação.

local_dataset <- "./dados/CXR8/images"
loc_img <- data.frame("local"=list.files(local_dataset, full.names = T,
recursive = T)) %>%
  mutate("Image.Index"=str_sub(local,-16))

loc_img %>% head()


### Selecionando dados unicos para o modelo


dici <- dici  %>% select(starts_with("rotulo")) %>%
  transmute(sumVar = rowSums(.)) %>% bind_cols(dici, .)

dados <- dici %>% filter(sumVar<2) %>%
  left_join(loc_img)

### Particionando em base de treinamento, teste e validação

set.seed(160006473) # Semente igual minha matricula
amostra <- sample(dim(dados)[1],size = .70*dim(dados)[1], replace = F)
base_treino <- dados[amostra,]
base_teste <- dados[-amostra,]

set.seed(160006473) # Semente igual minha matricula
amostra2 <- sample(dim(base_teste)[1],size = .50*dim(base_teste)[1],
replace = F)
base_validation <- base_teste[amostra2,]
base_teste <- base_teste[-amostra2,]

```

```
### Verificando variabilidade patológica nas bases

dados %>% select(starts_with("rotulo")) %>%
  summarise_all(~round(sum(.)/n()*100,2)) %>% reshape2::melt() %>%
  left_join(
base_treino %>% select(starts_with("rotulo")) %>%
  summarise_all(~round(sum(.)/n()*100,2)) %>% reshape2::melt()
  ,by = c("variable")) %>%
  left_join(
    base_teste %>% select(starts_with("rotulo")) %>%
  summarise_all(~round(sum(.)/n()*100,2)) %>% reshape2::melt()
  ,by = c("variable")) %>%
  left_join(
    base_validation %>% select(starts_with("rotulo")) %>%
  summarise_all(~round(sum(.)/n()*100,2)) %>% reshape2::melt()
  ,by = c("variable"))


### Criando pastas de teste, treino e validação

train_dir <- file.path(local_dataset, "train")
test_dir <- file.path(local_dataset, "test")
validation_dir <- file.path(local_dataset, "validation")
dir.create(train_dir)
dir.create(test_dir)
dir.create(validation_dir)


#### criando subpastas segundo categoria

nome_arquivos <- dici %>% select(starts_with("rotulo")) %>%
names() %>% c("Normal")

file.path(train_dir, nome_arquivos) %>%
  map(~dir.create(.x))

file.path(test_dir, nome_arquivos) %>%
  map(~dir.create(.x))

```

```
file.path(validation_dir, nome_arquivos) %>%
  map(~dir.create(.x))


#### Realocando as imagens

# Nao achei um jeito de tirar esse for ainda (operação demorada)
for (i in 1:14) {
files <- base_treino %>% filter(across(nome_arquivos[[i]])==1) %>%
pull(local)
walk2(files,
      file.path(train_dir, nome_arquivos[i]),
      ~filesstrings::move_files(.x, .y, overwrite=TRUE)
)

}
files <- base_treino %>% filter(sumVar==0) %>%
pull(local)
walk2(files,
      file.path(train_dir, nome_arquivos[15]),
      ~filesstrings::move_files(.x, .y, overwrite=TRUE)
)


for (i in 1:14) {
files <- base_teste %>% filter(across(nome_arquivos[[i]])==1) %>%
pull(local)
walk2(files,
      file.path(test_dir, nome_arquivos[i]),
      ~filesstrings::move_files(.x, .y, overwrite=TRUE)
)

}
files <- base_teste %>% filter(sumVar==0) %>% pull(local)
walk2(files,
      file.path(test_dir, nome_arquivos[15]),
      ~filesstrings::move_files(.x, .y, overwrite=TRUE)
)

```

```
for (i in 1:14) {
files <- base_validation %>% filter(across(nome_arquivos[[i]])==1) %>%
pull(local)
walk2(files,
      file.path(validation_dir, nome_arquivos[i]),
      ~filesstrings::move_files(.x, .y, overwrite=TRUE)
)

}
files <- base_validation %>% filter(sumVar==0) %>% pull(local)
walk2(files,
      file.path(validation_dir, nome_arquivos[15]),
      ~filesstrings::move_files(.x, .y, overwrite=TRUE)
)

### Cloud storage
O código abaixo envia os dados contidos nas pastas train,
test e validation para o cloud storage.

# install.packages(cloudml)
library(cloudml)
# gcloud_install()

local_dataset <- "./dados/CXR8/images"
train_dir <- file.path(local_dataset, "train")
test_dir <- file.path(local_dataset, "test")
validation_dir <- file.path(local_dataset, "validation")

gs_copy(train_dir, "gs://ventania10/train", recursive = TRUE, echo = TRUE)
gs_copy(test_dir, "gs://ventania10/test", recursive = TRUE, echo = TRUE)
gs_copy(validation_dir, "gs://ventania10/validation", recursive = TRUE,
echo = TRUE)




qtd <- dados %>% select(starts_with("rotulo")) %>%
  summarise_all(~sum(.)) %>% reshape2::melt() %>%
```

```
  left_join(
    base_treino %>% select(starts_with("rotulo")) %>%
      summarise_all(~sum(.)) %>% reshape2::melt()
    ,by = c("variable")) %>%
  left_join(
    base_teste %>% select(starts_with("rotulo")) %>%
      summarise_all(~sum(.)) %>% reshape2::melt()
    ,by = c("variable")) %>%
  left_join(
    base_validation %>% select(starts_with("rotulo")) %>%
      summarise_all(~sum(.)) %>% reshape2::melt()
    ,by = c("variable"))

local_dataset <- "./dados/CXR8/images"
train_dir <- file.path(local_dataset, "Normal_train")
validation_dir <- file.path(local_dataset, "Normal_val")
test_dir <- file.path(local_dataset, "Normal_test")


nome_arquivos <- dici %>% select(starts_with("rotulo")) %>% names()
nome_arquivos2 <- dici %>% select(starts_with("rotulo")) %>%
  names() %>%
  str_replace_all("rotulo","normal")



### Selecionando a parte de dados sem patologia para cada patologia
for (i in 1:14) {

val <- qtd %>% filter(variable==nome_arquivos[[i]]) %>%
  pull(value.x.x)

  # files <- base_treino %>% filter(sumVar==0) %>%
  #   sample_n(val) %>% pull(local)
  files <- base_teste %>% filter(sumVar==0) %>%
    sample_n(val) %>% pull(local)

  # base_treino <- base_treino[!c(base_treino£local %in% files),]
  base_teste <- base_teste[!c(base_teste$local %in% files),]
```

```
  walk2(files,
        file.path(test_dir, nome_arquivos2[i]),
        ~filesstrings::move_files(.x, .y, overwrite=TRUE)

)
}

length(list.files(test_dir,recursive = T))
gs_copy(train_dir, "gs://ventania10/Normal_train",
recursive = TRUE, echo = TRUE)
gs_copy(validation_dir, "gs://ventania10/Normal_val",
recursive = TRUE, echo = TRUE)
gs_copy(test_dir, "gs://ventania10/Normal_test",
recursive = TRUE, echo = TRUE)


dados %>%
  ggplot( aes(x=Patient.Age)) +
  geom_histogram( binwidth=3, fill="#69b3a2", color="#e9ecef", alpha=0.9) +
  theme_classic() +
  labs(x="Idade dos pacientes",
       y="")
  theme(
    plot.title = element_text(size=15)
  )
```

7.3 Treinamento

```
# esta parte foi realizada no ambiente do google colab o qual permite migrar
# pastas de forma manual, ou seja, para cada modelo em análise este era
# realocado para as pastas em uso (test_one,train_one,validation_one)
# isto deve-se ao fato de que varios modelos foram testados e criados ao longo
# do processo de desenvolvimento.

devtools::install_github("rstudio/keras")
install.packages("cloudml")

library(tidyverse)
```

```
library(keras)
library(cloudml)

system("sed -i -e 's,R_LD_LIBRARY_PATH}:${LD_LIBRARY,LD_LIBRARY_PATH}:${R_LD

tensorflow::tf_gpu_configured('GPU')

# download
gs_data_dir_local("gs://ventania10/train")
gs_data_dir_local("gs://ventania10/validation")
gs_data_dir_local("gs://ventania10/test")
gs_data_dir_local("gs://ventania10/Normal_train")
gs_data_dir_local("gs://ventania10/Normal_val")
gs_data_dir_local("gs://ventania10/Normal_test")

train_dir <- file.path(getwd(), "gs","ventania10","train_one")
validation_dir <- file.path(getwd(), "gs","ventania10","validation_one")
test_dir <- file.path(getwd(), "gs","ventania10","test_one")

### leituras das imagens/normalização

normalize_datagen <- image_data_generator(rescale = 1/255)
aug_datagen <- image_data_generator(
  rescale = 1/255,
  rotation_range = 5,
  zoom_range = 0.15,
  horizontal_flip = F,
  vertical_flip = F,
  fill_mode = "reflect"
)
train_generator <- flow_images_from_directory(
  train_dir,
  #gs_data_dir_local("gs://ventania10/train"),
  aug_datagen,
  target_size = c(224,224),
  batch_size = batch_tamanho,
  class_mode = "categorical",
  shuffle = T
)
```

```
50
51 validation_generator <- flow_images_from_directory(
52   validation_dir,
53   #gs_data_dir_local("gs://ventania10/validation"),
54   normalize_datagen,
55   target_size = c(224, 224),
56   batch_size = batch_tamanho,
57   class_mode = "categorical"
58 )
59 (bt<-length(list.files(test_dir, recursive = T)))
60 test_generator <- flow_images_from_directory(
61   test_dir,
62   normalize_datagen,
63   target_size = c(224,224),
64   batch_size = bt,
65   class_mode = "categorical"
66 )
67
68 #### rede pre-treinada
69 conv_base2 <- application_vgg19(weights = "imagenet",
70                        include_top = FALSE,
71                        input_shape = c(224, 224, 3))
72
73 #unfreeze_weights(conv_base2, from = "block5_conv1")
74 freeze_weights(conv_base2, from = "block1_conv1")
75
76 ### modelo em uso
77 model <-  keras_model_sequential() %>%
78   conv_base2 %>%
79   layer_global_average_pooling_2d() %>%
80   layer_dense(units = 2048, activation = "relu",
81   kernel_initializer = "he_normal") %>%
82   layer_dropout(rate = 0.6) %>%
83   #layer_dense(units = 512, activation = "relu",
84   #kernel_initializer = "he_normal") %>%
85   #layer_dropout(rate = 0.6) %>%
86   layer_dense(units = 2, activation = "softmax")
87
88 summary(model)
```

```
### callbacks
early_stop <- callback_early_stopping(
  monitor = "val_loss",
  min_delta = 1e-3,
  patience = 15,
  verbose = 1,
  restore_best_weights=T
)

checkpoint <- callback_model_checkpoint(
  "Modelo_pneumotora_19.h5",
  monitor = "val_loss",
  verbose = 1,
  save_best_only = T,
  mode = "min"
)

reduce_lr <- callback_reduce_lr_on_plateau(
  monitor = "val_loss",
  factor = 0.1,
  patience = 6,
  cooldown = 6
)

# funçao de perda
model %>% compile(
  loss = 'categorical_crossentropy',
  optimizer = optimizer_sgd(learning_rate =  1e-3, momentum = 0.90),
    metrics = c("acc","AUC") )

#### treinamento
history2 <- model %>% fit(
  train_generator,
  epochs = 100,
  steps_per_epoch = round(length(list.files(train_generator$directory,
  recursive = T))/batch_tamanho),
  validation_data = validation_generator,
  validation_steps = round(length(list.files(validation_generator$directory,
```

```
  recursive = T))/batch_tamanho),
  callbacks = c(checkpoint, reduce_lr, early_stop),
  verbose=1
)


### avaliaçao

plot(history2)

model %>% evaluate(validation_generator)
a <- generator_next(test_generator)
table("predito"=model %>% predict(a[1]) %>% k_argmax() %>%
as.integer,"real"=a[2]%>% k_argmax() %>% as.integer  )
```