Universidade de Brasília
Faculdade de Tecnologia

Algoritmos de aprendizado profundo para detecção de osteoporose em imagens odontológicas

Iago Cossentino de Andrade

TRABALHO DE GRADUAÇÃO

ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Brasília

2022

Universidade de Brasília
Faculdade de Tecnologia

Algoritmos de aprendizado profundo para detecção de osteoporose em imagens odontológicas

Iago Cossentino de Andrade

Trabalho de Graduação submetido como requisito parcial para obtenção do grau de Engenheiro de Controle e Automação

Orientadora: Profa. Dra. Mylène C.Q. Farias

Brasília
2022

Andrade, Iago Cossentino de.
A553a Algoritmos de aprendizado profundo para detecção de osteoporose em imagens odontológicas / Iago Cossentino de Andrade; orientadora Mylène C.Q. Farias. -- Brasília, 2022.
64 p.

Trabalho de Graduação (Engenharia de Controle e Automação) -- , 2022.

1. Redes Neurais Convolucionais. 2. Osteoporose. 3. Classificação. 4. Detecção. I. Farias, Mylène C.Q., orient. II. Título

Universidade de Brasília
Faculdade de Tecnologia

Algoritmos de aprendizado profundo para detecção de osteoporose em imagens odontológicas

Iago Cossentino de Andrade

Trabalho de Graduação submetido como requisito parcial para obtenção do grau de Engenheiro de Controle e Automação

Trabalho aprovado. Brasília, 05 de outubro de 2022:

Profa. Dra. Mylène C.Q. Farias,
UnB/FT/ENE
Orientadora

Prof. Dr. Marcelo A. Marotta,
UnB/IE/CIC
Examinador interno

Profa. Dra. Flavia M. G. S. A. Oliveira,
UnB/FT/ENE
Examinadora interna

Brasília
2022

Este trabalho é dedicado à Deus, aos meus pais, a minha noiva
e a todos aqueles que permaneceram e batalham nesse curso de graduação.
É uma longa jornada de amadurecimento e que todos possam se sentir
acolhidos nesse processo.

Agradecimentos

Agradeço a todos que me ajudaram até aqui, tanto fora da universidade quanto dentro. Foram muitos tijolos colocados por pessoas queridas para que eu chegasse até onde estou.

Agradeço aos meus amigos de colégio que continuaram iluminando meus dias ao longo dos anos de universidade. Em ordem alfabética: Anawana Fernandes, Gabriel Sobral, Moises Amorim e tantos outros. Obrigado por todas as experiências e partidas de jogos de tabuleiro.

Também agradeço aos bons amigos que compuseram essa jornada acadêmica comigo: Amanda dos Santos, Caio Saigg, Júlio Eduardo, Luís Felipe, Pedro Pereira, Rodrigo Zamagno, Vinícius Carvalho e tantos outros mecamigos. Muito obrigado por todas as aventuras nerds, apoios acadêmicos e aulas compartilhadas. Cá estamos nós após incontáveis semestres!

Agradeço também aos meus professores Carol Andrade e Wesley Andrade. Vocês abriram portas e me apresentaram um mundo vasto que nunca imaginei que gostaria tanto! O trabalho de vocês foi essencial para enfrentar essa jornada.

Agradeço a minha orientadora, professora Mylène Farias, pela disponibilidade, pelos conselhos e, principalmente, por toda paciência. Foram dias corridos e muito obrigado por todo o auxílio ao longo deles. Também agradeço aos professores Marcelo Marotta e Flavia Oliveira por comporem a banca e pela disponibilidade.

Agradeço à minha noiva, e quase esposa, Isabella Fernandes. Eu não poderia imaginar uma companheira mais graciosa, divertida, atenciosa, comunicativa, dançarina e gentil. Foram muitos bons momentos juntos na UnB e, agora, muitos mais fora dela. Seu suporte foi vital para realização desse trabalho e suas palavras me deram motivação para continuar construindo o nosso futuro maior e melhor juntos. Muito obrigado por esse ciclo.

Meus agradecimentos também aos meus pais, Lúcio Kleber e Andrea Cossentino. Vocês tornaram toda essa trajetória até a UnB possível e nunca me deixaram faltar nada. Muito obrigado por todos os conselhos nesse processo de amadurecimento acadêmico e profissional. Não consigo expressar nesse texto as incontáveis contribuições de vocês, mas sou profundamente grato por todo o carinho e o cuidado!

Por fim, agradeço à Deus, por ter me dado forças todos os dias, por ter me guiado e por ter me guardado.

"Uncrumpling paper balls is what machine learning is about:
finding neat representations for complex, highly folded data manifolds."
(François Chollet)

Resumo

O estabelecimento de diagnósticos médicos precoces é de grande importância para melhor tratamento de diversas doenças. Em sua grande maioria, esses diagnósticos requerem um especialista. Com o intuito de evitar imprecisões, especialistas recorrem a múltiplos exames para confirmação de doenças, o que demanda tempo e recursos. Dentre diversas doenças, um grave problema de saúde pública é a osteoporose. A osteoporose atua diminuindo a resistência óssea, predispondo o indivíduo a uma maior chance de fratura por trauma. Sendo assintomática, a osteoporose é frequentemente detectada apenas quando ocorre uma fratura, sendo assim, é importante a sua detecção precoce. Recentemente, foi encontrada alta concordância entre a análise da integridade da cortical mandibular avaliada por radiologistas experientes e maxilo-faciais e a análise de panorâmicas odontológicas efetuada por algoritmos de aprendizado de máquina. Os avanços tecnológicos nos permitem estabelecer métodos melhores de diagnósticos todos os dias, como trataremos especificamente, com inteligência artificial. O uso de algoritmos de aprendizado de máquinas (*machine learning*) se apresenta como uma ferramenta que pode auxiliar profissionais de saúde na tomada de decisões em situações clínicas complexas, poupando tempo e outros recursos. Neste trabalho, foram explorados três modelos de redes neurais convolucionais (*VGG16*, *ResNet50* e *Xception*) com diferentes camadas de profundidades e configurações de hiperparâmetros. Com a expectativa de produzir um classificador para detecção de osteoporose em panorâmicas odontológicas, foram utilizados dois conjuntos de dados fornecidos pelo Hospital Universitário de Brasília (HUB). O primeiro composto por 340 imagens panorâmicas odontológicas completas e o segundo composto por 712 imagens com regiões de interesse dessas panorâmicas. Os treinamentos das redes foram realizados em duas etapas, *feature extraction* e *fine-tuning*. Foram testadas diferentes opções de descongelamento das redes (*unfreezing*), tempos de treinamento e quantidades de camadas densas. Ao final do projeto, foi possível obter um modelo detector de osteoporose para radiografias panorâmicas odontológicas maxilo-faciais com acurácia de 88.7%. De tal maneira, reafirmando que o uso de inteligência artificial como ferramenta de uso médico é viável.

Palavras-chave: Redes Neurais Convolucionais. Osteoporose. Classificação. Detecção.

Abstract

The establishment of early medical diagnoses is of great importance for better treatment of various diseases. In order to avoid inaccuracies, specialists resort to multiple exams to confirm diseases, which demand time and resources. Among several diseases, a serious public health problem is osteoporosis. Osteoporosis acts by decreasing bone strength, predisposing the individual to a greater chance of fracture by trauma. Being asymptomatic, osteoporosis is often detected only when a fracture occurs, so its early detection is important. Recently, high agreement was found between the analysis of the mandibular cortical integrity assessed by experienced maxillofacial radiologists and the analysis of dental panoramics performed by machine learning algorithms. Technological advances allow us to establish better diagnostic methods every day, as we will specifically address, with artificial intelligence. The use of machine learning algorithms is presented as a tool to help the decision making of health professionals dealing with complex clinical situations, saving time and other resources. In this work, three models of convolutional neural networks (VGG16, ResNet50 and Xception) with different amounts of layers and hyperparameter configurations were explored. With the expectation of producing a classifier for the detection of osteoporosis in dental panoramas, two datasets provided by the Hospital Universitário de Brasília (HUB) were used. The first consists of 340 complete dental panoramic images and the second consists of 712 images with regions of interest (ROI) from these panoramics. The network training was performed in two stages, feature extraction and fine-tuning. Different options of unfreezing the networks, training times and amounts of dense layers were tested. At the end of the project, it was obtained an osteoporosis detection model for maxillofacial dental panoramic radiographs with an accuracy of 88.7%. In such a way, reaffirming that the use of artificial intelligence as a tool for medical use is viable.

Keywords: Convolutional Neural Networks. Osteoporosis. Classification. Detection.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

Softmax	*Softargmax* ou Função Exponencial Normalizada 24
AI	*Artificial Intelligence* 21
CNN	*Convolutional Neural Network* 33
DL	*Deep Learning* 21
FN	*False Negative* - Falso Negativo 46
FP	*False Positive* - Falso Positivo 46
GPDS	Grupo de Processamento Digital de Sinais 42
GPU	*Graphics Processing Unit* 27
HUB	Hospital Universitário de Brasília 19
IA	Inteligência Artificial 21
ILSVRC	*ImageNet Large Scale Visual Recognition Challenge* 43
ML	*Machine Learning* 21
MSE	*Mean Squared Error* 25
ReLU	*Rectified Linear Units* 24
ResNet	*Residual Network* 43
RMSProp	*Root Mean Squared Propagation* 29
RN	Rede Neural 21
RNC	Rede Neural Convolucional 19
ROI	*Region Of Interest* 42
RP	Radiografia Panorâmica 19
SGD	Stochastic Gradient Descent 29
TCFC	Tomografia Computadorizada de Feixe Cônico 19
TN	*True Negative* - Verdadeiro Negativo 46
TP	*True Positive* - Verdadeiro Positivo 46
UnB	Universidade de Brasília 19
VGG	*Visual Geometry Group* 43

Lista de símbolos

Sumário

1 Introdução

1.1 Contextualização

O estabelecimento de diagnósticos médicos precoces é de grande importância para o melhor tratamento de diversas doenças. Os avanços tecnológicos nos permitem estabelecer métodos mais acurados de diagnósticos todos os dias e, ultimamente, os métodos com inteligência artificial têm se tornado cada vez mais populares[1]. Em outras palavras, a crescente disponibilidade de informações médicas digitais na forma de registros eletrônicos de saúde, assim como o rápido desenvolvimento de métodos analíticos de grandes bases de dados (*big data*), o uso de algoritmos de aprendizado de máquinas (*machine learning*) se apresenta como uma ferramenta que pode auxiliar profissionais de saúde na tomada de decisões em situações clínicas complexas. Da mesma forma, o uso destas ferramentas poderá, em um futuro próximo, predizer doenças sistêmicas crônicas, pela identificação automatizada de padrões não perceptíveis ao olho humano.

Recentemente, diversos estudos têm sido desenvolvidos com o objetivo de diagnosticar a osteoporose mais precocemente. Em particular, foi encontrada alta concordância entre a análise da integridade da cortical mandibular avaliada por radiologistas experientes e maxilo-faciais e a análise de panorâmicas odontológicas efetuada por algoritmos de aprendizado de máquina. Outras pesquisas buscaram identificar padrões em imagens odontológicas para rastreamento da osteoporose, com a avaliação do padrão trabecular e cortical da mandíbula (CASTRO et al., 2020) (FRANCIOTTI et al., 2021). Assim, algumas alterações ósseas microestruturais podem ser usadas para avaliar ou detectar risco de fratura, como é proposto nesse trabalho. Essa doença é considerada um grave problema de saúde pública. A osteoporose atua diminuindo a resistência óssea, predispondo o indivíduo a uma maior chance de fratura por trauma. Sendo assintomática, a osteoporose é frequentemente detectada apenas quando ocorre uma fratura. Além disso, a fratura é um dos principais fatores de impacto socioeconômico da doença, levando muitas vezes à incapacitação do indivíduo e causando internações hospitalares onerosas. Com o envelhecimento da população mundial, o impacto de doenças no geral na população vêm sofrendo um aumento significativo em todo o mundo, incluindo o Brasil (AZIZIYEH et al., 2019) (BURGE et al., 2007). Como em outras áreas de estudo com imagens médicas, é um desafio obter variedade de imagens, tendo em vista a especificidade dos exames, a quantidade de exames que são realizados daquele tipo (CHOKSI; JEPSEN; CLINES, 2018) e a disponibilidade de acesso aos bancos de dados para realização de estudos.

[1] https://www.alliedmarketresearch.com/neural-network-market

1.2 Proposta e definição de objetivos do trabalho

1.2.1 Objetivo geral

Com base nas alterações ósseas microestruturais usadas para detectar risco de fratura e dando continuidade a estudos anteriores promissores no tema (LEE et al., 2020), este projeto possui como objetivo geral o desenvolvimento de algoritmos de redes neurais convolucionais (RNC), a partir de redes já existentes, com a finalidade de auxiliar no estabelecimento de diagnóstico precoce de osteoporose a partir de imagens odontológicas. Para tal finalidade, imagens de radiografias panorâmicas odontológicas (RP) e de tomografias computadorizadas de feixe cônico (TCFC) serão utilizadas como dados de entrada destes algoritmos para diagnosticar precocemente osteoporose. A coleta das imagens será inicialmente retrospectiva, em bancos de dados existentes no Hospital Universitário de Brasília (HUB). O acesso a esses bancos de dados é privado e não são fornecidas informações de sexo, idade, nome dos pacientes ou outras informações individualizadas.

1.3 Objetivos específicos

Os objetivos específicos desse trabalho envolvem treinar múltiplas redes neurais com estruturas diferentes, estabelecer as de melhor desempenho em acurácia e aprimorá-las alterando a estrutura de camadas. Os objetivos se dividem da seguinte maneira:

- Estabelecer banco de dados para treinamento da rede e verificar implicações de representatividade.
- Inicialmente treinar modelos compostos por redes diferentes e selecionar a de melhor acurácia entre essas redes.
- Aprimorar a rede escolhida ajustando mais detalhadamente os hiperparâmetros (taxa de aprendizado, número de épocas, tamanho de lotes e tamanho das imagens) e aumentar camadas com o intuito de obter as melhores métricas possíveis.

1.4 Apresentação do manuscrito

Este manuscrito se divide conforme exposto:

- Capítulo 2 - Fundamentação teórica: neste capítulo, são tratados aspectos contextuais teóricos necessários para entendimento do trabalho sobre aprendizado de máquinas em geral. São apresentados conceitos amplamente utilizados em inteligência artificial que não se restringem a trabalhos com imagens. São esses conceitos: tensores, perceptrons,

funções de ativação, funções de custo, funções de otimização, descida do gradiente, conjuntos de treino, teste e validação, épocas, lotes, hiperparâmetros e generalização.

- Capítulo 3 - Conceitos em Redes Neurais Convolucionais: neste capítulo, são apresentados conceitos mais específicos de aprendizado profundo voltados a Redes Neurais Convolucionais para imagens. São esses conceitos: pré-processamento de imagens, camadas de uma rede neural convolucional, *feature extraction* e *fine-tuning*.
- Capítulo 4 - Metodologia: neste capítulo, são abordadas as técnicas de treinamento e ajuste utilizadas para realização do trabalho, arquiteturas de redes neurais disponíveis, bem como as escolhidas (*VGG16*, *ResNet50* e *Xception*), procedimentos experimentais, modelagem e processos para treinamento e validação das redes.
- Capítulo 5 - Resultados: neste capítulo, são apresentados os resultados quantitativos obtidos utilizando o banco de dados disponível. Também são discutidos os desempenhos dos modelos ao longo do processo de ajuste fino até obtenção da melhor métrica atingida, após treino, nos conjuntos de validação e teste. São analisadas as métricas calculadas e as matrizes de confusão obtidas.
- Capítulo 6 - Conclusão: este capítulo inclui resumo do trabalho realizado, considerações finais e propostas de melhorias futuras. São abordadas as métricas de redes em dois conjuntos de dados, implementações que podem vir a melhorar acurácia e tempo de treinamento em trabalhos futuros e são mencionadas redes diferentes das testadas que podem desempenhar melhor para o tipo de problema enfrentado.

2 Fundamentação teórica

Tendo vista os objetivos apresentados na introdução (Capítulo 1), faz-se necessário apresentar termos e conceitos iniciais básicos para melhor compreender a temática abordada no presente trabalho. A maioria dos tópicos foi selecionada com base nos livros de Chollet (2017) e Goodfellow, Bengio e Courville (2016).

2.1 Tensores

Usualmente, os dados utilizados em sistemas de inteligência artificial são organizados em estruturas de dados de *arrays* multidimensionais de natureza numérica que são chamados de tensores. Diferentes tipos de entrada (imagem, vídeo, dados temporais) são expressos em tipos diferentes de tensores.

- Escalares (tensor dimensão 0): tensores de dimensão 0 são chamados de escalares e armazenam um único valor.
- Vetores (tensor dimensão 1): compreendem dados em uma única dimensão em sequência. Não confundir a dimensionalidade de eixos de um tensor com a dimensionalidade do vetor. Um tensor de dimensão N possui N eixos, enquanto o vetor de dimensão N possui apenas um eixo de tamanho N (N dados armazenados).
- Matriz (tensor dimensão 2): comumente conhecidos como matrizes, os tensores de dimensão 2 podem ser visualizados como retângulos de dados. Também são chamados de vetores de dados de formato: (*samples, features*).
- Séries temporais: tensor 3D de formato (*samples, timesteps, features*).
- Imagens: tensor 4D de formato (*samples, height, width, channels*).
- Vídeos: tensor 5D de formato (*samples, frames, height, width, channels*).

2.2 Perceptron

Redes Neurais (RN) são conceitos criados na década de 50 que só tiveram o seu auge de popularidade quando anos mais tarde o poder de processamento computacional permitiu a aplicação desses conceitos de maneira mais vasta (CHOLLET, 2017). O primeiro modelo de perceptron surge com Rosenblatt (1958) e são popularmente chamados de "neurônios". Perceptrons ligados em camadas originaram o nome de redes neurais ou redes neurais

profundas quando possuem muitas camadas. Estão apresentados exemplos de perceptron e multicamada de perceptrons na Figura 1 e Figura 2, respectivamente.

O objetivo com os modelos de redes organizados por camadas é aproximar uma função f de maneira a mapear entradas x a uma categoria $\hat{y}$ (FRANÇA, 2021). Sendo assim, a aproximação é dada por $\hat{y} = f(x,\theta)$, com θ sendo valores numéricos conhecidos por pesos ou parâmetros da rede.

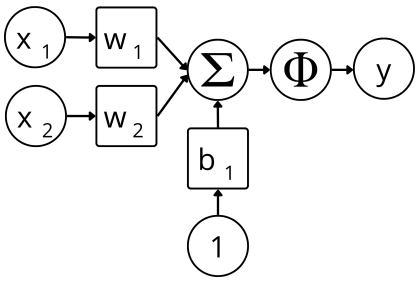


Figura 1 – Esquemático de perceptron simples. É exposto que as entradas $x = [x_1, x_2]$ são multiplicadas pelo vetor de pesos $W = [w_1, w_2]^T$ e somadas à multiplicação unitária de um viés (*bias*) b_1. Esse somatório é passado como argumento para a função de ativação Φ (seção 2.3) que introduz não-linearidade ao modelo. W e b constituem parte do conjunto de parâmetros θ ajustáveis para o aprendizado.

Fonte: Própria

O modelo básico do perceptron, de N entradas x_i, pode ser representado matematicamente por:

$$y = \Phi(xW + b) = \Phi(\sum_{i=1}^{N}(x_i w_i) + b). \tag{2.1}$$

Sistematizando o funcionamento de perceptrons multicamadas, a matriz de pesos $w_{ij}^{(k)}$ é identificada pelo i correspondente à entrada x_i, pelo j correspondente à unidade perceptron da respectiva camada e pelo k sobrescrito representando a camada respectiva. De tal modo, as seguintes equações descrevem as saídas nas camadas da Figura 2.

$$\begin{bmatrix} y_1^{(1)} & y_2^{(1)} \end{bmatrix} = \Phi(\begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} w_{11}^{(1)} & w_{12}^{(1)} \\ w_{21}^{(1)} & w_{22}^{(1)} \end{bmatrix} + \begin{bmatrix} b_1^{(1)} & b_2^{(1)} \end{bmatrix}); \tag{2.2}$$

$$\begin{bmatrix} \hat{y}_1^{(2)} \end{bmatrix} = \Phi(\begin{bmatrix} y_1^{(1)} & y_2^{(1)} \end{bmatrix} \begin{bmatrix} w_1^{(2)} \\ w_2^{(2)} \end{bmatrix} + \begin{bmatrix} b_1^{(2)} \end{bmatrix}). \tag{2.3}$$

Denotando $B^{(k)}$ como o vetor de vieses $b_j^{(k)}$, $W^{(k)}$ a matriz de pesos e Φ a função de ativação, a relação entre a Equação 2.3 e a Equação 2.3, fica demonstrada por:

$$\hat{y} = \Phi(\Phi(xW^{(1)} + B^{(1)})W^{(2)} + B^{(2)}). \tag{2.4}$$

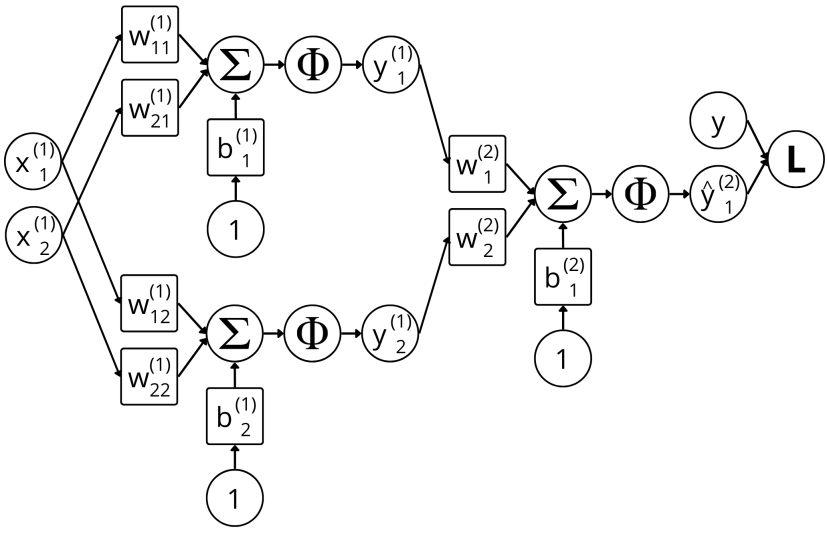


Figura 2 – Esquemático de perceptron multicamada de 2 camadas. A primeira camada possui 2 perceptrons simples e a segunda camada 1 perceptron simples. As camadas são identificadas pelo sobrescrito. A saída predita $\hat{y}$ ao final é comparada com a classe y esperada por uma função de custo L (seção 2.4).

Fonte: Própria

2.3 Funções de Ativação

Além das operações lineares que podem ser feitas com os tensores em camadas de perceptrons, como demonstrado (seção 2.2), não linearidades podem ser inseridas após as operações lineares como maneira de melhor aproximar f. Como apresentadas anteriormente (Figura 1), essas operações foram chamadas de funções de ativação (denotadas por Φ). Funções de ativação estão inseridas usualmente entre camadas do tipo densa (subseção 3.2.1). Sem as funções de ativação, os modelos de redes neurais seriam aproximadores lineares independentemente dos seus tamanhos e complexidades. Hornik, Stinchcombe e White (1989) pelo teorema da aproximação universal propõem matematicamente que modelos com não linearidades podem se aproximar de funções contínuas com precisão arbitrária. Sendo assim, fica posto que as redes de aprendizado podem agir com aproximadoras universais.

Para maior acesso ao espaço de aproximações do modelo (HORNIK; STINCHCOMBE; WHITE, 1989), são adicionadas não linearidades que serão apresentadas nas funções de ativção ReLU, *sigmoid* e *softmax* abaixo, que não são as únicas existentes, mas são as mais populares.

2.3.1 ReLU (*Rectified Linear Units*)

A função ReLU é geralmente é a escolhida para as camadas internas do tipo densa (subseção 3.2.1) e sua operação é nulificar valores negativos e passar valores positivos sem alteração. Ela faz isso tomando o maior valor entre a entrada x e 0. ReLU é denotada pela

função:

$$\text{ReLU}(x) = \max(0, x) \tag{2.5}$$

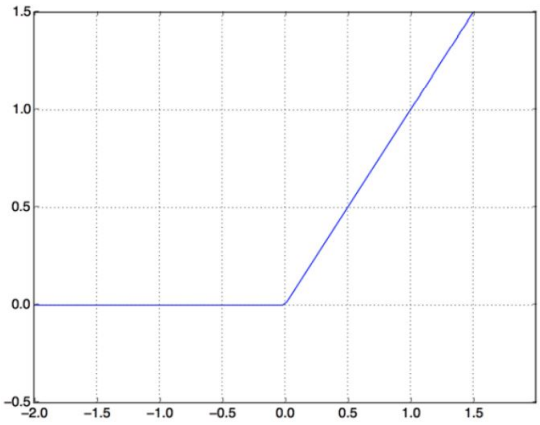


Figura 3 – Curva de ativação ReLU. Note que não há limitação para os valores de saída, portanto que sejam positivos.

Fonte: Chollet (2017)

2.3.2 Sigmoide

Juntamente com a *Softmax* (subseção 2.3.3), a Sigmoide é uma das opções de função de ativação escolhidas para serem utilizadas na camada densa final do sistema. Sigmoide é capaz de comprimir os valores da saída entre 0 e 1, sendo a função escolhida para problemas de classificação binária, como o que será tratado nesse trabalho. A interpretação do resultado dessa função é feita de que a saída 0 seria uma classe (saudável) e que a saída 1 outra classe (osteoporose, Figura 25). A função sigmoide está descrita em Equação 2.6, com a entrada x.

$$\sigma(x) = \frac{1}{1 + e^{-x}}. \tag{2.6}$$

2.3.3 *Softmax* (*Softargmax* ou Função Exponencial Normalizada)

A *Softmax* distribui um vetor de saídas de perceptrons de uma camada entre porcentagens que se somam em 1. Geralmente utilizada para classificações multiclasse, a interpretação da saída dessa função é de que os valores indicam a probabilidade de pertencimento a uma classe entre várias. A *softmax* para a entrada x_i entre todas as K entradas é calculada por:

$$softmax(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}}. \tag{2.7}$$

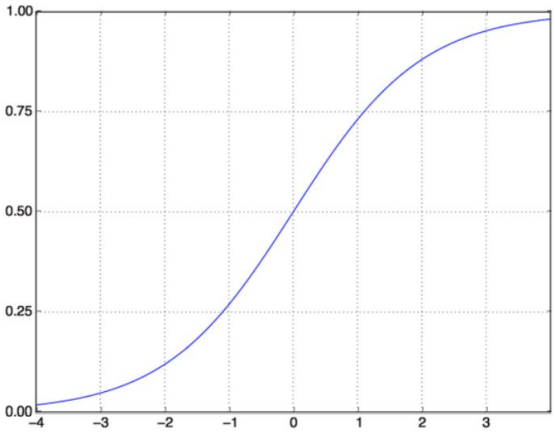


Figura 4 – Curva de ativação *Sigmoid*. Observe que a função de ativação possui saídas assíntotas a 0 e 1.

Fonte: Chollet (2017)

2.4 Funções de Custo

Ao projetarmos um sistema de aprendizado de máquinas, é necessário medir as diferenças entre os valores preditos e os valores reais de forma a otimizar o modelo. De posse dos valores de predição $\hat{y}$ e as saídas desejadas y, existem funções que calculam o erro de predição (ou custo - em inglês *loss*) adequadamente de acordo com o sistema sendo implementado. Essas são as funções de custo L, também são chamadas meramente de *loss* ou funções de objetivo. As mais populares são *binary_crossentropy* (subseção 2.4.1), *categorical_crossentropy* e a MSE[1]. A escolha das funções mais adequadas depende da aplicação da rede. Para o problema de classificação binária desse trabalho, a entropia cruzada binária (*binary_crossentropy*) foi a função escolhida (CHOLLET, 2017). A Tabela Tabela 1 lista recomendações de funções de ativação e funções de custo para cada tipo de problema.

Tabela 1 – Recomendação de função de ativação da última camada e função de custo.

Tipo do problema	**Função de Ativação (última camada)**	**Função de custo**
Classificação Binária	*sigmoid*	*binary_crossentropy*
Classificação multiclasse de *label* única	*softmax*	*categorical_crossentropy*
Classificação multiclasse com múltiplas *labels*	*sigmoid*	*binary_crossentropy*
Regressão de valores arbitrários	Nenhuma	MSE

Fonte: Chollet (2017)

[1] https://keras.io/api/losses/

2.4.1 Entropia Cruzada Binária (*Log Loss*)

A entropia cruzada binária (L_{BCE}) é aplicada em situações em que sua última função de ativação é uma sigmoide (subseção 2.3.2). Portanto, ela pressupõe saídas variando entre 0 e 1. Essa função de custo é adequada para distinguir entre duas classes (GODOY, 2022) ou para indicar presença ou não de uma classe entre várias que se pretende identificar em meio às entradas ao mesmo tempo (Tabela 1). A entropia cruzada de N predições $\hat{y}$ e as saídas desejadas y é descrita por:

$$L_{BCE} = -\frac{1}{N}\sum_{i=1}^{N}(y_i \log \hat{y}_i + (1 - y_i)\log(1 - \hat{y}_i)). \tag{2.8}$$

O Loss parcial é avaliado de acordo com a classe de y conforme apresentado abaixo e depois somado para todo N ponderadamente. De tal maneira, $L_{BCEparcial}$ mede a distância entre $\hat{y}$ e y logaritmicamente:

$$L_{BCEparcial} = \begin{cases} \log(1 - \hat{y}), & \text{se } y = 0 \\ \log \hat{y}, & \text{se } y = 1 \end{cases} \tag{2.9}$$

Para ajustes do modelo ao longo das épocas de treinamento (seção 2.6), são utilizadas as funções de custo, mas não somente. Funções de custo mensuram de alguma forma quantitativamente o quão distante as predições se encontram do esperado ao longo do treinamento, já as chamadas métricas de avaliação tratam de avaliar o desempenho do modelo como um todo. Usualmente se utiliza acurácia, mas diversas são as métricas que podem ser utilizadas [2] e, muitas vezes, os termos de métricas e *loss* são utilizadas de maneira intercambiável.

2.5 Treino, Validação e Teste

Um bom modelo de *machine learning* se desdobra em conseguir obter bons resultados em meio a dados novos, isso significa capacidade de generalização, tendo em vista que seu aprendizado é realizado anteriormente em um conjunto de dados finito. Isso é feito modelando o sistema para dados de entrada diversos e depois testando o treinamento em outro conjunto de dados.

A divisão do conjunto de dados (*dataset*), assim como em muitos outros campos da ciência, segue uma proporção guiada pelo Princípio de Pareto (PARETO; BOUSQUET; BUSINO, 1964) (PARETO, 1971). O conjunto de dados usualmente é dividido entre dois grupos (treinamento e teste) numa proporção de 80-20% ou 70-30% respectivamente (GHOLAMY; KREINOVICH; KOSHELEVA, 2018). Após isso, o sistema é ajustado ao longo de

[2] https://keras.io/api/metrics/

várias iterações (épocas, seção 2.6) utilizando os dados de treinamento. Durante esse período de ajuste, o conjunto de teste é usado como referência para acompanhar o progresso do treinamento no que seriam dados novos. A relação de compromisso de escolha (*trade-off*) entre dividir os dados de treinamento e de teste em parcelas diferentes diz respeito à variância dos estimadores calculados nesses grupos.

Também é possível utilizar três grupos de dados (treinamento, validação e teste), em que a validação é usada para acompanhar o treinamento e o grupo de teste para uma avaliação final após toda a testagem dos diversos parâmetros da rede. Isso é feito para evitar também vieses por parte do programador ao longo do ajuste de hiperparâmetros (seção 2.9). A divisão entre três grupos é feita seguindo a proporção mencionada anteriormente e dividindo o grupo de testes pela metade entre validação e teste, como nesse trabalho (70-15-15%, respectivamente).

2.6 Épocas e Lotes

Como já exposto, o aprendizado da rede neural se dá calibrando os pesos θ iteradamente até minimizar as diferenças entre o esperado e o predito pela rede. Essas iterações ocorrem computando o custo e minimizando-o repetidamente ao longo do *dataset* de treino. Essa passagem por todo o conjunto de dados é chamada de uma época (*epoch*).

Computacionalmente falando, é custoso para se treinar o modelo calculando os parâmetros de todo o conjunto de dados de uma vez só, dependendo do número de amostras. Sendo assim, as iterações ao longo de uma época muitas vezes são calculadas em lotes (*batches*). O tamanho do lote, é a quantidade de amostras que são enviadas à rede para a realização de uma iteração. Em termos gerais, para um modelo de M amostras e tamanho de lote m_b, uma época é computada após $\frac{M}{m_b}$ iterações.

O tamanho do lote e número de épocas são hiperparâmetros (seção 2.9) ajustáveis pelo programador a depender do problema que está sendo estudado. Segundo Smith (2018), o tamanho do lote tem grande impacto no tempo de treinamento, de tal forma que, quanto maior o lote, menor tende a ser o tempo de treinamento. Em contrapartida, deve ser levado em conta a memória disponível pela GPU (*Graphics Processing Unit*) sendo utilizada para treinamento do modelo.

2.7 Descida do Gradiente

Foi visto que as funções de custo (seção 2.4) são responsáveis por mensurar o quanto da predição de um modelo está distante do desejado, mas ainda não foi explicado como o ajuste dos pesos θ é feito para diminuir o *Loss*. O gradiente é um vetor tangente a uma superfície que aponta para a direção de maior incremento do valor nessa superfície, portanto,

o gradiente negativo é a direção que diminui o valor na superfície. Ele é representado pelo símbolo ∇ (nabla) e é o conceito por trás dos algoritmos de redução da função de custo L, que é diferenciável.

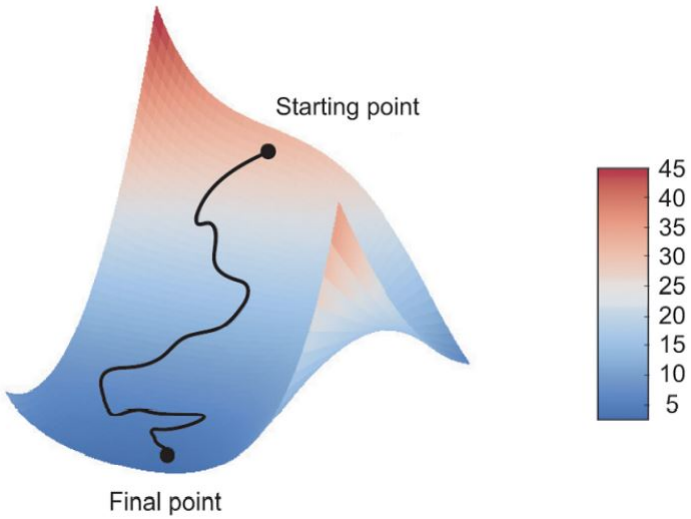


Figura 5 – Curva de descida de gradiente em superfície de *loss*. Note que a figura é meramente ilustrativa, pois o espaço dimensional de um modelo de redes neurais não se resume a três dimensões. Tendo em vista que cada peso θ do modelo compõe uma dimensão de ajuste para reduzir o *loss* e que existem modelos com milhões de pesos, uma curva realista como a representada pela figura não é humanamente visualizável geometricamente.

Fonte: Chollet (2017)

Utilizando regra da cadeia, redes neurais, através da retropropagação (*backpropagation*), realizam a descida do gradiente para ajustar os pesos das camadas para minimizar a função de custo (seção 2.4). A retropropagação propaga os gradientes para as camadas anteriores a cada iteração das épocas (seção 2.6) de modo a alterar o pesos gradualmente no sentido minimizante com uma taxa de aprendizado η. O algoritmo toma θ_n, sendo n a iteração da época, para computar θ_{n+1}, direcionando com a *loss* em $-\eta\nabla_{\theta_n}L(\theta_n)$ como na seguinte equação:

$$\theta_{n+1} = \theta_n - \eta\nabla_{\theta_n}L(\theta_n). \tag{2.10}$$

Taxa de aprendizado constitui um dos hiperparâmetros (seção 2.9) mais importantes para aprendizado adequado da rede. Os valores típicos atribuídos para η variam entre 10^{-1} e 10^{-6}, segundo Goodfellow, Bengio e Courville (2016).

2.8 Otimização

Como já comentado na seção 2.7, a descida do gradiente pode ser feita pela Equação 2.10, mas ao longo dos anos, diferentes algoritmos foram propostos e o estado da arte é obtido por algoritmos mais complexos. Métodos diferentes de propagar o gradiente en-

volvem alterar o tamanho dos lotes (seção 2.6) e alterar o algoritmo de otimização anterior (Equação 2.10).

2.8.1 Otimizações envolvendo tamanho do lote

A depender do tamanho do lote para a descida do gradiente (seção 2.7), tempos de treinamento do modelo diferentes podem ser obtidos, bem como quantidade de memória da GPU diferentes.

- Descida do Gradiente de Lote Inteiro (*Full-Batch Gradient Descent*): É a descida do gradiente realizada para todo o conjunto de dados de uma vez em uma época (RUDER, 2016). A descida do gradiente de lote inteiro promove tempos de computação das épocas menores e com menos oscilações de gradiente entre lotes, por não conterem todo o conjunto de dados em um lote só. A descida ocorre com menos perturbações de trajeto. Denotando i como o iterador entre as amostras, $L(\theta)_{(i:i+n)}$ como o custo de θ de i a $i+n$ amostras e M o total de amostras, obtemos o seguinte algoritmo:

$$\theta_{n+1} = \theta_n - \eta \nabla_{\theta_n} L(\theta_n)_{(1:M)}. \tag{2.11}$$

- Descida do Gradiente de Mini-Lotes *(Mini-batch gradient descent)*: É a descida do gradiente realizada para cada lote de m_b amostras do conjunto de dados ao longo de iterações na época (RUDER, 2016). Tem o benefício de ser computacionalmente mais leve ao equilibrar perturbações de trajetória e uso de memória pela GPU, a depender do tamanho do hiperparâmetro do lote (seção 2.6).

$$\theta_{n+1} = \theta_n - \eta \nabla_{\theta_n} L(\theta_n)_{(i:i+m_b)}. \tag{2.12}$$

- Descida do Gradiente Estocástica (*Stochastic Gradient Descent*): Também chamada de SGD é a descida do gradiente realizada para cada amostra do conjunto de dados ao longo de iterações na época (RUDER, 2016). É a descida de gradiente menos custosa computacionalmente e de trajetória mais ruidosa.

$$\theta_{n+1} = \theta_n - \eta \nabla_{\theta_n} L(\theta_n)_{(i)}. \tag{2.13}$$

2.8.2 Algoritmos de Otimização

Além de otimizar o aprendizado utilizando diferentes tamanhos de lote m_b para o algoritmo descrito na Equação 2.10, foram criados algoritmos mais eficientes e robustos que trabalham de maneira similar a esse. A plataforma do Keras oferece otimizadores[4] com

[3] https://stats.stackexchange.com/questions/153531/what-is-batch-size-in-neural-network

[4] https://keras.io/api/optimizers/

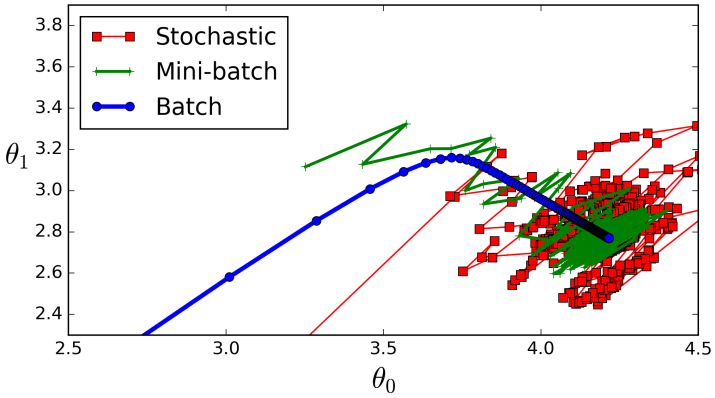


Figura 6 – Curva de comparativa entre descida do gradiente de lote inteiro (azul), descida do gradiente de mini-lotes (verde) e descida do gradiente estocástica (vermelha) de dois pesos arbitrários θ_0 e θ_1. Observa-se que a abordagem por mini-lotes se encontra em um meio termo entre a estocástica e o lote inteiro no que tange a perturbações da trajetória de minimização.

Fonte: StackExchange[3](autor original da figura desconhecido)

formulações mais complexas para tratar da descida do gradiente. Nesse trabalho foi utilizado o RMSProp.

Traduzido para o português como "propagação da raiz do valor quadrático médio", o RMSProp foi proposto por Hinton, Srivastava e Swersky (2012). Ele foi criado para ser utilizado em regime de mini-batch, sendo considerado mais robusto por ser menos influenciável por alterações bruscas de gradiente entre um lote e outro ao trabalhar com um fator quadrático médio dos gradientes de lotes passados. Essa robustez é causada pela redução do deslocamento abrupto em regiões momentâneas de crescimento de gradiente e não reduzir tanto o deslocamento em regiões de baixo gradiente.

Os parâmetros da equação do RMSProp constituem hiperparâmetros, mas não têm valores padrão alterados, mantendo os valores fornecidos pelo Keras[5]. Sendo γ um fator de média móvel e ϵ uma constante pequena para manter estabilidade, o algoritmo de RMSProp é descrito pelas seguintes regras de atualização de θ:

$$S_{\theta_n} = \gamma S_{\theta_{n-1}} + (1-\gamma)(\nabla_{\theta_n} L(\theta_n))^2 \tag{2.14}$$

e

$$\theta_{n+1} = \theta_n - \frac{\eta}{\sqrt{S_{\theta_n}} + \epsilon} \nabla_{\theta_n} L(\theta_n). \tag{2.15}$$

A taxa de aprendizado η padrão é 10^{-4}, γ padrão é 0.9 e ϵ padrão é 10^{-7}. Combinando elementos do RMSProp e mecanismos de inércia, o otimizador Adam[6] (*Adaptive Moment Estimation*) foi criado posteriormente ao RMSProp por Kingma e Ba (2014) e possui resultados de performance similares, mas é mais amplamente conhecido e usado.

5 https://keras.io/api/optimizers/rmsprop/

6 https://keras.io/api/optimizers/adam/

2.9 Hiperparâmetros

Definidos anteriormente ao processo de aprendizagem, hiperparâmetros são os parâmetros definidos pelo programador manualmente. Os mais importantes para este trabalho são os seguintes:

- Taxa de aprendizado η (seção 2.7) - Os valores típicos atribuídos para η variam entre 10^{-1} e 10^{-6}.
- Número de épocas M (seção 2.6) - Os valores típicos atribuídos para M variam entre 10 e 1000 a depender do tipo do problema sendo tratado.
- Tamanho do lote m_b (seção 2.6) - Os valores típicos atribuídos para m_b variam em potências de 2 entre 2^0 e 2^{10}. Neste trabalho, as redes neurais tiveram m_b ajustadas para não extrapolar a memória das GPUs utilizadas.
- Tamanho das imagens - Apesar de alguns modelos conseguirem inferir o tamanho das entradas, às vezes é necessário passar como parâmetro essa informação do conjunto das imagens utilizadas. Neste trabalho, foram utilizados 2 conjuntos de dados com imagens de 4019×819 pixels e 820×819 pixels, que foram redimensionadas para $1/4$ das suas dimensões originais, ou seja para 1022×204 pixels e 205×204 pixels, respectivamente.

2.10 Regressão e Classificação

Sistemas de aprendizado de máquinas são conhecidas pela versatilidade de suas aplicações, mas os problemas tratados em grande maioria se dividem em regressão e classificação. A regressão é uma abordagem utilizada quando se busca estimar (um ou múltiplos valores) contínuos com base nos dados de entrada. Por exemplo, para prever ações da bolsa de valores ou a temperatura em certo horário do dia. A classificação tem por objetivo dividir os dados de entrada em regiões, que podem ser atribuídos como pertencentes a uma ou mais classes. A classificação pode ser utilizada, por exemplo, em tarefas como diferenciar rostos, animais ou dígitos.

2.11 Generalização

A generalização de uma rede não é conseguida diretamente sem algumas experimentações, de forma que sejam atendidos alguns critérios. Durante o processo de treinamento, é esperado que o modelo se ajuste com o avanço as épocas, de modo que a rede se especialize em identificar aquele conjunto de dados cada vez mais. Nas épocas iniciais, o custo (*loss*) do conjunto de validação e treinamento será similar e alto, sendo esse o estado de *underfitting* ou de subajuste. Em maiores épocas, o modelo estará se especializando no conjunto de

treinamento, de forma que em qualquer outro conjunto de dados haverá um custo (*loss*) maior do que no conjunto de treinamento. Este é um caso de sobreajuste (*overfitting*). Um modelo com capacidade de generalizar, isto é, ajustado adequadamente, possui um menor custo (*loss*), acontecendo antes de começar na área de *overfitting*.

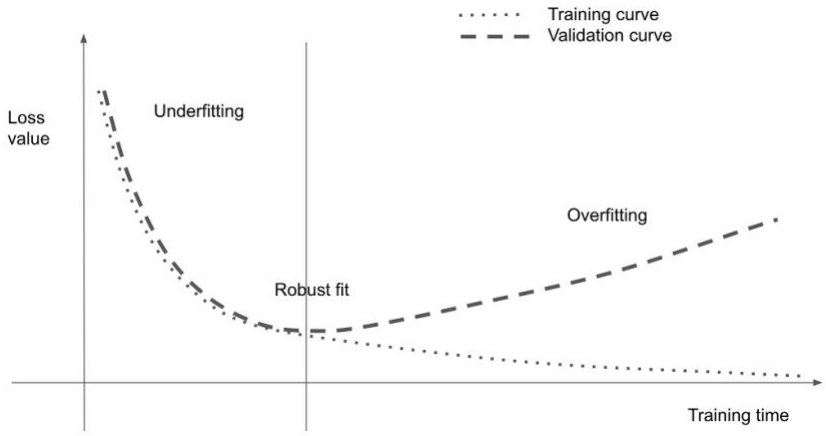


Figura 7 – Curva de *loss* no processo de treinamento e validação ao longo das épocas. Observa-se que o treinamento com o conjunto de treino, como o esperado, reduz o *loss* continuamente com as iterações. Até o momento de ajuste adequado (*robust fit*), o modelo está aprendendo características gerais que são similares tanto ao conjunto de treinamento, quanto ao de validação. Após a fase mencionada, o modelo começa a se especializar tanto no conjunto de dados de treinamento que o *loss* de validação volta a ser alto. Enquanto modelos subajustados são modelos genéricos inacurados, modelos sobreajustados são modelos acurados não genéricos. O ajuste adequado está no balanço entre acurácia e generalização na classificação de dados novos ao modelo.

Fonte: Chollet (2017)

3 Redes Neurais Convolucionais

Com vistas à plena compreensão dos aspectos teóricos abordados neste trabalho, estende-se a apresentação iniciada no Capítulo 2. O presente capítulo aborda conceitos mais específicos sobre as redes neurais para imagens as utilizadas. Algumas operações desse capítulo são mais utilizadas em imagens do que em outros tipos de dados de modelos para outros propósitos em geral.

3.1 Pré-processamento

Antes de se aplicar um modelo para aprendizado de máquinas em si, são necessários dados para fornecer a esse modelo. No que tange a um aprendizado robusto, esses dados precisam ser úteis e suficientes para que o modelo consiga extrair informações deles.

3.1.1 Verificação do *Dataset*

Para se obter um aprendizado coeso, além de dados suficientes, é importante que se faça uma verificação se esses dados fazem sentido no contexto da aplicação. São verificações importantes:

- Corretude de classes - Checar se a classe atribuída é coerente com a imagem apresentada. Por exemplo, um gato catalogado como cachorro[1].
- Balanceamento do *dataset* - Se os dados estão quantitativamente balanceados entre as classes de dados. Por exemplo, uma rede não necessariamente obteve capacidade de diferenciar dados ao ter acurácia de 90% e, entre esses dados, existir um grupo consiste de 90% dos dados totais. "Chutar"essa classe sempre obteria o mesmo resultado com aprendizado quase nenhum.
- Verificação de *outliers* - Em imagens, *outliers* consistem de áreas de segmentação que não estão ajustadas de maneira correta. Esse tipo de observação é importante para redes de segmentação de imagens, principalmente de aprendizado não supervisionado, tendo em vista que parte da avaliação do *loss* é computado nessa segmentação inadequada.

3.1.2 Aumento do conjunto de dados (*Data Augmentation*)

Por vezes, a quantidade de imagens não é suficiente para treinamento adequado de um modelo, haja vista que alguma classe é muito pequena ou até mesmo todas elas. A

[1] https://www.kaggle.com/competitions/dogs-vs-cats/data

quantidade de imagens para que uma rede se ajuste depende do tipo de problema analisado, mas, quanto mais dados para serem analisados, melhor.

Em situações como essa, a técnica recomendada seria realizar um aumento do conjunto de dados (*data augmentation*) com as imagens do próprio banco de dados. Isso pode ser realizado tomando as imagens do grupo e as duplicando com algumas modificações para que se tornem mais dados para a rede. É o processo de artificialmente gerar novos exemplos. Isso permite que a rede classifique mais imagens que são consideradas diferentes entre si. São comuns as seguintes técnicas:

- Rotação
- Translação
- *Zoom*
- Desfoque
- Alteração de contraste
- Alteração de projeção/perspectiva
- Inversão de eixo (*flipping*)
- Correção de *gamma* (ZHANG et al., 2017)
- Injeção de ruído (ZHANG et al., 2017)

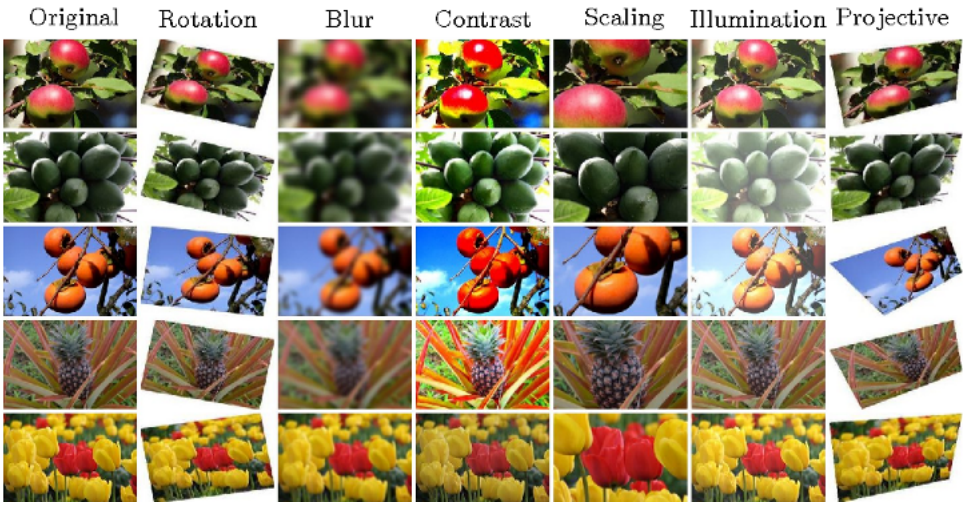


Figura 8 – Diferentes técnicas de *data augmentation* aplicadas a imagens diferentes. Não é necessário aplicar somente um tipo de técnica por vez. Podem ser utilizadas combinações de técnicas na mesma imagem nova gerada.

Fonte: Amey Gondhalekar[2], Medium

[2] https://medium.com/analytics-vidhya/data-augmentation-is-it-really-necessary-b3cb12ab3c3f

3.2 Camadas de uma *CNN*

Muitos dos modelos utilizados nesse trabalho (seção 4.2) possuem camadas em comum conectadas de maneiras diferentes. Essa seção trata de explicar as principais operações que essas camadas mais comuns realizam em redes neurais convolucionais.

3.2.1 Camada densa (*Dense*)

Como tratada no Capítulo 2, é a camada formada por Perceptrons (seção 2.2) densamente conectados. Se não acrescidos de *Droupout* (subseção 3.2.4), todas as saídas são conectadas a todas as entradas da camada seguinte.

3.2.2 Camada de convolução

Expressa em código por classes diferentes no Keras[3], essa camada é a responsável pelo nome das redes neurais convolucionais.

Convoluções são operações lineares entre uma dada entrada $x(t)$ e uma função de pesos $w(a)$ ao longo do tempo. Considerando um $w(a)$ como função de média entre valores $x(t)$ no tempo contínuo, $s(t)$ é similar a $x(t)$ suavizado (GOODFELLOW; BENGIO; COURVILLE, 2016):

$$s(t) = \int x(a)w(t-a)da. \tag{3.1}$$

Convoluções são também denotadas por asterisco da seguinte maneira:

$$s(t) = (s * w)(t). \tag{3.2}$$

Tomando t somente por valores inteiros e definindo x e w somente nos valores de t, obtemos a convolução discreta na Equação 3.3.

$$s(t) = (s * w)(t) = \sum_{-\infty}^{\infty} x(a)w(t-a). \tag{3.3}$$

Para operações em duas dimensões e utilizando terminologia de redes neurais, x é chamado de entrada e w é o kernel. Essa convolução discreta entre a imagem bidimensional I e o kernel bidimensional K, de tamanho m por n, está descrita na Equação 3.4.

$$S(i,j) = (I * K)(i,j) = \sum_{m}\sum_{n} I(m,n)K(i-m,j-n). \tag{3.4}$$

[3] https://keras.io/api/layers/#convolution-layers

Como convolução é operação comutativa (GOODFELLOW; BENGIO; COURVILLE, 2016), podemos reescrever a Equação 3.4, em que $S(i,j)$ é chamado de *feature map*:

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n). \tag{3.5}$$

Em termos de implementação mais simples, bibliotecas de inteligência artificial geralmente implementam a função de relação cruzada ou correlação cruzada (cross-correlation). Essa função é a mesma que uma convolução sem inverter o kernel K, mas muitas vezes chamada de convolução do mesmo jeito. A Figura 9 exemplifica o kernel aplicado a uma entrada genérica.

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n). \tag{3.6}$$

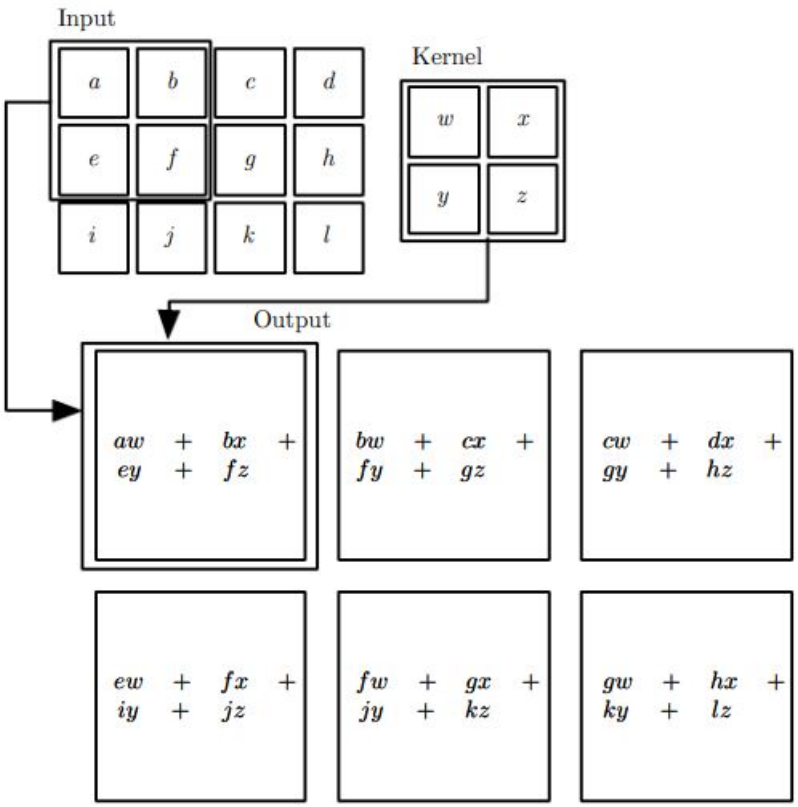


Figura 9 – Exemplo de aplicação 2D de convolução sem inversão de kernel (correlação cruzada). Note que entrada $I(i,j)$ com kernel $K(m,n)$ aplicado fornece saída S de tamanho $(i+1-m, j+1-n)$.

Fonte Goodfellow, Bengio e Courville (2016)

[4] https://mlnotebook.github.io/post/CNN1/

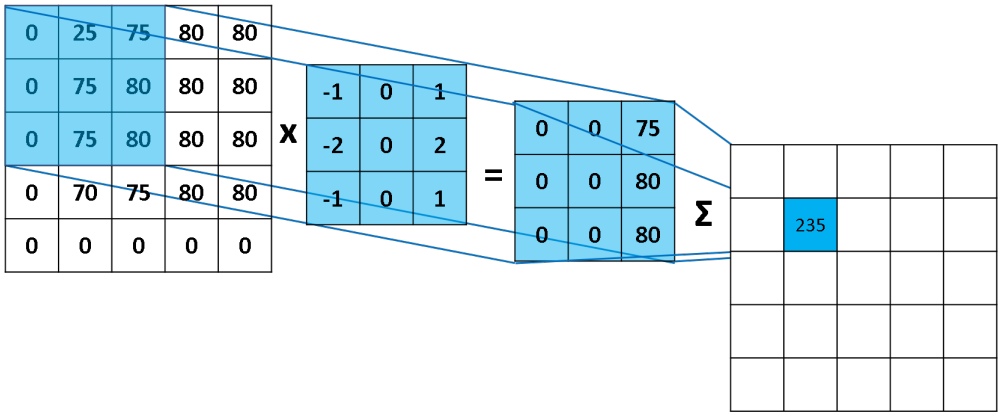


Figura 10 – Exemplo numérico de convolução 2D para obtenção de *feature map*.

Fonte: MLnotebook[4]

3.2.3 Camada de *Max-pooling (Maxpool)*

A camada de *Max-pooling*, assim como uma convolução (subseção 3.2.2) de kernel com dimensões grandes, é uma amostragem agressiva das entradas, segundo Chollet (2017). O kernel $K(m,n)$, geralmente quadrado ($m = n$), toma o maior valor de dentro do kernel. A saída S ao final da operação possui tamanho $(i/m, j/n)$, se a aplicação do kernel se deslocar a cada (m,n) entradas. O *pooling* mais popular é tomando o maior valor, porém existem outros métodos utilizando média do kernel ou menor valor do kernel.

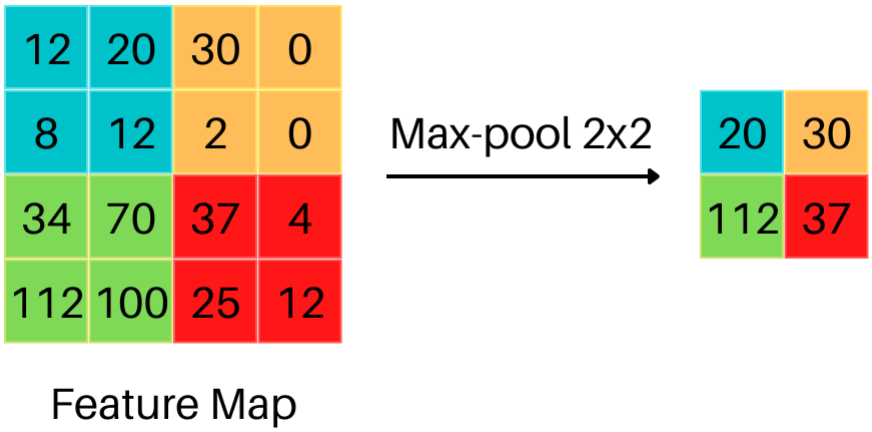


Figura 11 – Exemplo de aplicação de *max-pooling* 2D de kernel 2x2.

Fonte: baseada em Paperswitchcode[5]

[5] https://paperswithcode.com/method/max-pooling

3.2.4 *Dropout*

Segundo França (2021, p.17), regularização é "o conjunto de estratégias no campo do aprendizado de máquina que são explicitamente projetadas para reduzir o erro de teste, possivelmente às custas de aumentar o erro de treinamento". Em outras palavras, regularização é uma estratégia para obtenção de generalização (seção 2.11) do modelo.

Além de aumentar o conjunto de dados (subseção 3.1.2), *Dropout* (SRIVASTAVA, 2013) constitui de outra estratégia que pode ser utilizada para melhorar o treinamento final (SRIVASTAVA et al., 2014). A operação do *dropout* é composta por nulificar alguns perceptrons das camadas internas. Esse processo de apagar neurônios até as saídas é feito aleatória e temporariamente (GOODFELLOW; BENGIO; COURVILLE, 2016). Esse tipo de operação é considerada uma das formas mais computacionalmente baratas de regularização e que mais promove benefícios ao treinamento. A Figura 12 exemplifica o processo.

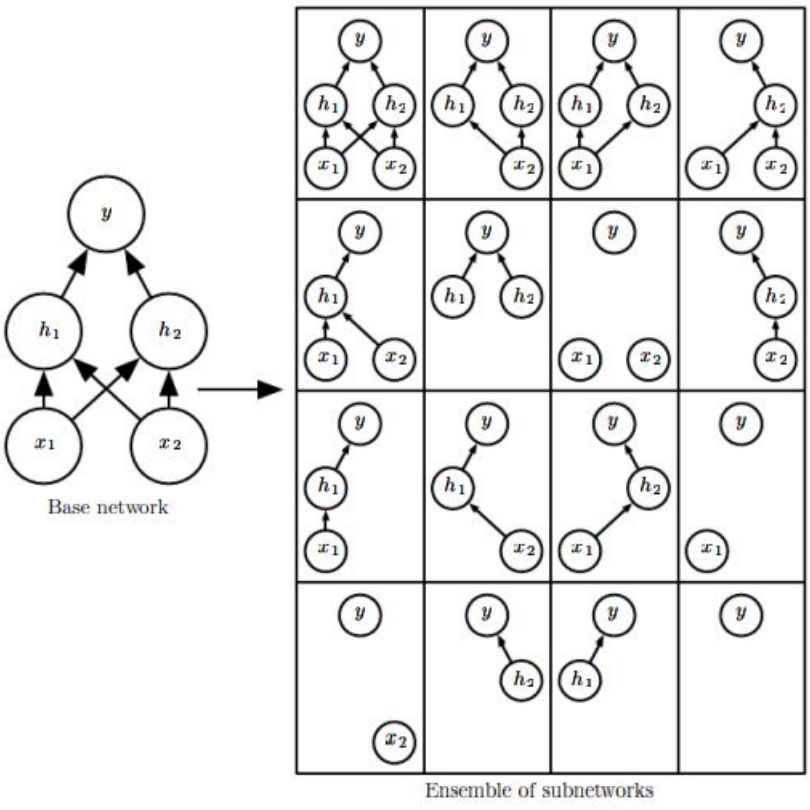


Figura 12 – Exemplo de rede com todas as combinações de *dropout* sem retirada da saída. Note que em muitos dos casos não há conexões entre as entradas $[x_1, x_2]$ e a saída y. Esse tipo de problema se torna insignificante para redes com entradas mais extensas e densidade de camadas maior, visto que a probabilidade de não haver conexão que liga alguma entrada até a saída é bem menor.

Fonte: Goodfellow, Bengio e Courville (2016)

3.2.5 Achatamento (*Flattening*)

Por fim, para permitir que camadas densas trabalhem com dados 2D vindos de camadas convolucionais ou de *max-pooling*, é realizado um processo de achatamento (*flattening*) responsável por rearranjar o *feature map*. O rearranjo é feito concatenando o final de cada linha da imagem com o início da próxima linha, formando um vetor (tensor dimensão 1, seção 2.1).

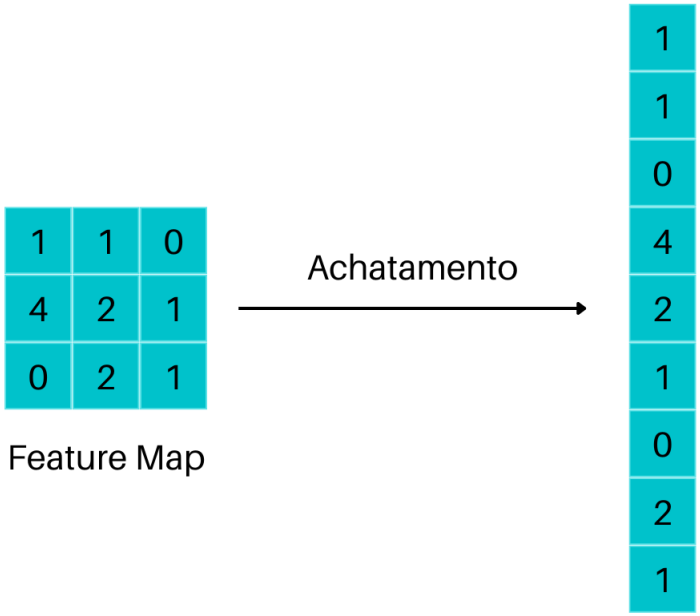


Figura 13 – Exemplo de aplicação de *flattening* de dados 2D (matriz) para 1D (vetor). Observe que nenhum dado é perdido nesse processo, somente rearranjado para entrada da próxima camada.

Fonte: baseada em SuperDataScience[6]

3.3 Treinamento e Ajuste Fino

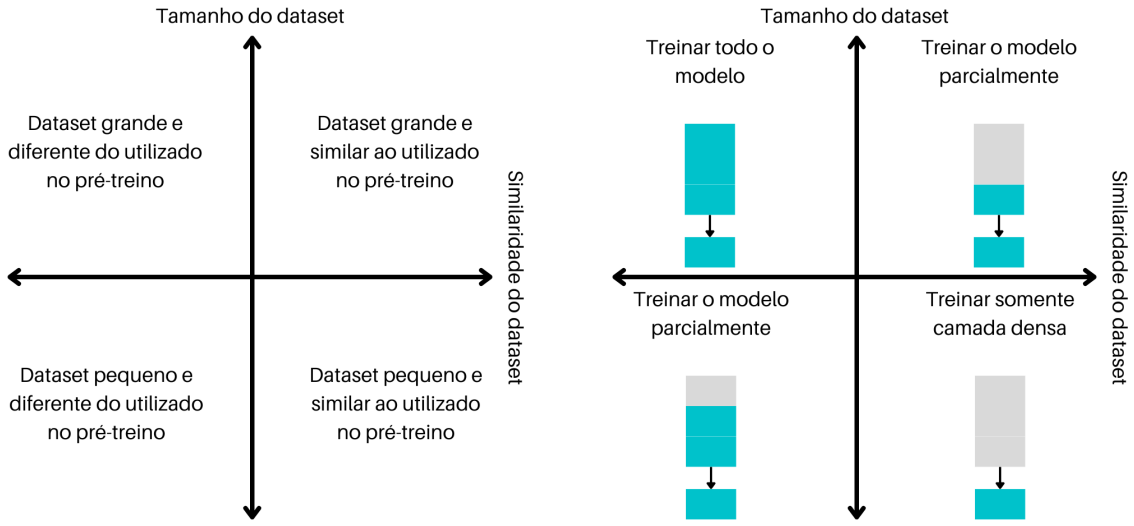
Seguindo estratégias de treinamento do estado da arte como Krizhevsky, Sutskever e Hinton (2012) ou Simonyan e Zisserman (2014), uma das sequências de treinamento de redes neurais em classificação de imagens que mais tem obtido bons resultados é a Transferência de Aprendizado (*Transfer Learning*) (PAN; YANG, 2010). A transferência de aprendizado é denotada pelo processo de aproveitar o treinamento prévio de redes neurais já testadas e reajustá-las para outros propósitos, obtendo-se boa acurácia e poupando tempo Rawat e Wang (2017).

Na prática, o modelo do usuário é "montado"utilizando como base o modelo pré-treinado que queira testar. O modelo pré-treinado possui camadas convolucionais e densas

[6] https://www.superdatascience.com/blogs/convolutional-neural-networks-cnn-step-3-flattening

com pesos θ estado da arte para classificação de um tipo de dado. O usuário, então, realiza o processo de treinamento em duas seguintes etapas:

- Ajuste das camadas densas do usuário - Como a parte convolucional (pré-treinada) está bem ajustada para classificar conjunto de dados similar, é realizado o congelamento dessa parte da rede (*freezing*). O congelamento consiste de não permitir que sejam ajustados os pesos da região congelada. Sendo assim, são calibrados somente os pesos das camadas densas adicionadas pelo usuário, que possuem os seus pesos θ inicializados com valores aleatórios. Chollet (2017) chama essa etapa de *feature extraction*.
- Ajuste fino total ou parcial do modelo - Tendo ajustado as camadas densas, o modelo passa por uma sessão de ajuste fino (*fine-tuning*). O processo consiste em descongelar (*unfreeze*) todo o modelo ou parte dele e, com uma taxa de aprendizado (seção 2.7) bem baixa, ajustar os pesos por mais algumas épocas. Essa etapa promove ajuste dos pesos reutilizados para captar representações abstratas mais relevantes para o problema em questão (CHOLLET, 2017).



(a) (b)

Figura 14 – Diagramas de sugestão de descongelamento das redes de aprendizado profundo. Em (a), temos as condições de similaridade do conjunto de dados (*Dataset Similarity*) e de tamanho do conjunto de dados (*Dataset Size*). Em (b), estão as sugestões do quanto descongelar as camadas convolucionais. Para o quadrante 1, é sugerido que se treine o modelo do zero, aproveitando somente a arquitetura da base convolucional, e, para o quadrante 4, que se treine apenas o topo denso. Quadrantes 3 e 2 devem ser ajustadas apenas as camadas de topo convolucionais (ajuste fino parcial) e todas as camadas densas.

Fonte: baseada em Pedro Marcelino[7]

[7] https://towardsdatascience.com/transfer-learning-from-pre-trained-models-f2393f124751

4 Metodologia

Neste capítulo, serão abordadas as etapas que foram realizadas e as decisões tomadas para a obtenção dos resultados. Na seção 4.1, é discutida a composição do conjunto de dados de imagens, como estão distribuídas e seus desafios. Ainda na mesma seção, trata-se das ferramentas utilizadas para programar e outras plataformas de execução dos códigos. A seção 4.2 contém informações sobre quais modelos base foram utilizados e a seção 4.3 as estratégias e decisões tomadas para treinamento deles. A etapa de aperfeiçoamento do melhor modelo com incremento de camadas densas é descrita na seção 4.4. Por fim, a seção 4.5 descreve quais métricas foram priorizadas para tomada de decisões e avaliação dos resultados.

4.1 Elaboração da base de dados

Prezando por criar um modelo que tomasse como entrada imagens radiografias panorâmicas odontológicas (RP) sem tratamento ou com tratamento mínimo. O conjunto de imagens foi fornecido em parceria com HUB[1], consistindo de dois subconjuntos. O primeiro conjunto é composto por panorâmicas odontológicas completas (sem cortes, Figura 15) de resolução espacial grande e sem qualquer processamento. Já o segundo conjunto possui imagens processadas (recortadas em um formato quadrado, Figura 16) de forma a conter apenas a região de interesse (ROI) - a região maxilo-facial. Neste conjunto, cada panorâmica do primeiro conjunto (não-processada) gera duas imagens no segundo conjunto, sendo uma imagem da região maxilo-facial esquerda e outra da região maxilo-facial direita. Após a redução das dimensões em ¼ (seção 2.9) para ambos os conjuntos, eles agora possuem imagens de 1022x204 pixels e 205x204 pixels, respectivamente. Dimensões menores permitem um treinamento mais rápido e com menos pesos. O primeiro conjunto de panorâmicas não-processadas é chamado de *Full*, enquanto que o conjunto de panorâmicas processadas é chamado de *Divided*.

O conjunto *Full* é ainda subdividido nas classes *Healthy* (saudável), *Osteopenia* e *Osteoporosis*. O estudo de osteopenia não foi escopo desse trabalho, portanto essa classe não foi utilizada. O conjunto *Divided* possuía as classes *Healthy*, *Osteoporosis* e *No-Diagnosis*. Naturalmente, essa última classe também não foi utilizada. A distribuição dos dados pode ser vista na Tabela 2. Além disso, abaixo também é apresentada a Figura 16. Esta figura exibe imagens de exemplo do conjunto *Divided* que, por sua vez, são derivadas da Figura 15, pertencente ao conjunto *Full*. Importante enfatizar que até setembro de 2022, o conjunto de dados em desenvolvimento menor possuía 340 imagens, o que não são consideradas

[1] https://www.gov.br/ebserh/pt-br/hospitais-universitarios/regiao-centro-oeste/hub-unb

muitas imagens para um treinamento de uma rede neural. Por causa disso, optou-se por realizar *Data Augmentation* (subseção 3.1.2) e um *split* dos dados em 70-15-15% (GHOLAMY; KREINOVICH; KOSHELEVA, 2018). Mais detalhes destes processos podem ser encontrados na seção 5.2.

Tabela 2 – Distribuição das classes das imagens dos conjuntos *Full* e *Divided* em quantidade de imagens e percentual do total de imagens do conjunto.

Classe		**Healthy**	**Osteoporosis**	**Osteopenia/ No Diagnosis**	**Total**
full	**Quantidade de Imagens**	169	155	16	340
	Porcentagem do Total de Imagens	49,71%	45,59%	4,71%	100,00%
divided	**Quantidade de Imagens**	370	310	32	712
	Porcentagem do Total de Imagens	51,97%	43,54%	4,49%	100,00%

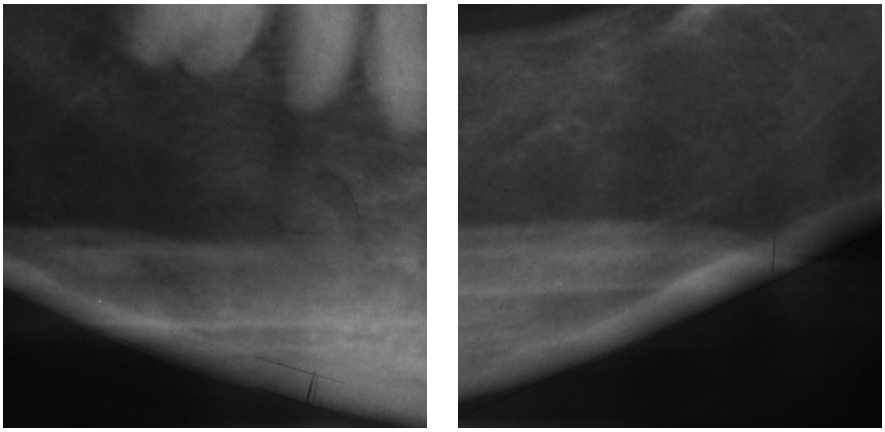
Fonte: Própria.

Figura 15 – Panorâmica odontológica completa do conjunto *Full*.

Fonte: HUB

Para codificação das redes neurais, foi utilizada a linguagem de programação Python[2] com as bibliotecas disponibilizadas do Tensorflow[3] e Keras[4]. Para testes mais simples, foi utilizado o servidor gratuito dispobilizado pelo Google Colab [5]. Para o treinamento que exigiu uma maior quantidade de horas das redes neurais, os recursos oferecidos pelo Colab não eram suficientes, portanto foram utilizadas as GPUs do servidor GPDS[6] da UnB. As especificações de hardware se encontram na seção 5.1.

2 https://www.python.org/
3 https://www.tensorflow.org/
4 https://keras.io/
5 https://colab.research.google.com/
6 http://www.gpds.ene.unb.br/



(a) (b)

Figura 16 – Exemplos de panorâmica odontológica do conjunto *Divided* da região de interesse (ROI) referente à Figura 15: (a) primeiro corte - área esquerda; (b) Segundo corte - área direita.

Fonte: HUB

4.2 Modelos utilizados

Para compor a base convolucional das redes desse trabalho, foram escolhidas 3 redes disponíveis pelo Keras[7]:

- **VGG16** (OxfordNet) - Foi criada pela Oxford (SIMONYAN; ZISSERMAN, 2014) e, entre as 3 bases escolhidas, é a menor rede neural. Venceu a competição ILSVRC (ImageNet) [8]) em 2014. Possui 16 camadas[9] que podem se visualizadas na Figura 17.
- **ResNet50** - Criada por He et al. (2015), the Residual Network 50[10] está entre as melhores redes no estado da arte. A ResNet50 possui 107 camadas, sendo 48 de convolução, 1 de *max-pooling* e 1 de *average-pooling*.
- **Xception** - Considerada a versão extrema da Inception (SZEGEDY et al., 2015), a Xception (CHOLLET, 2016) é uma rede neural de 81[11] camadas. Seu diferencial está em realizar convoluções separáveis por profundidade nos canais da imagem. Convoluções comuns são realizadas entre espaço e profundidade ao mesmo tempo. Na Xception, o processo é dividido em dois, uma convolução espacial seguida de uma convolução puntiforme na profundidade dos canais.

As arquiteturas foram utilizadas com seus pesos pré-treinados originais, resultantes do treinamento com o conjunto de dados da ImageNet, que oferece centenas de milhares de classes.

7 https://keras.io/api/applications/
8 https://www.image-net.org/
9 https://keras.io/api/applications/vgg/#vgg16-function
10 https://keras.io/api/applications/resnet/#resnet50-function
11 https://keras.io/api/applications/xception/

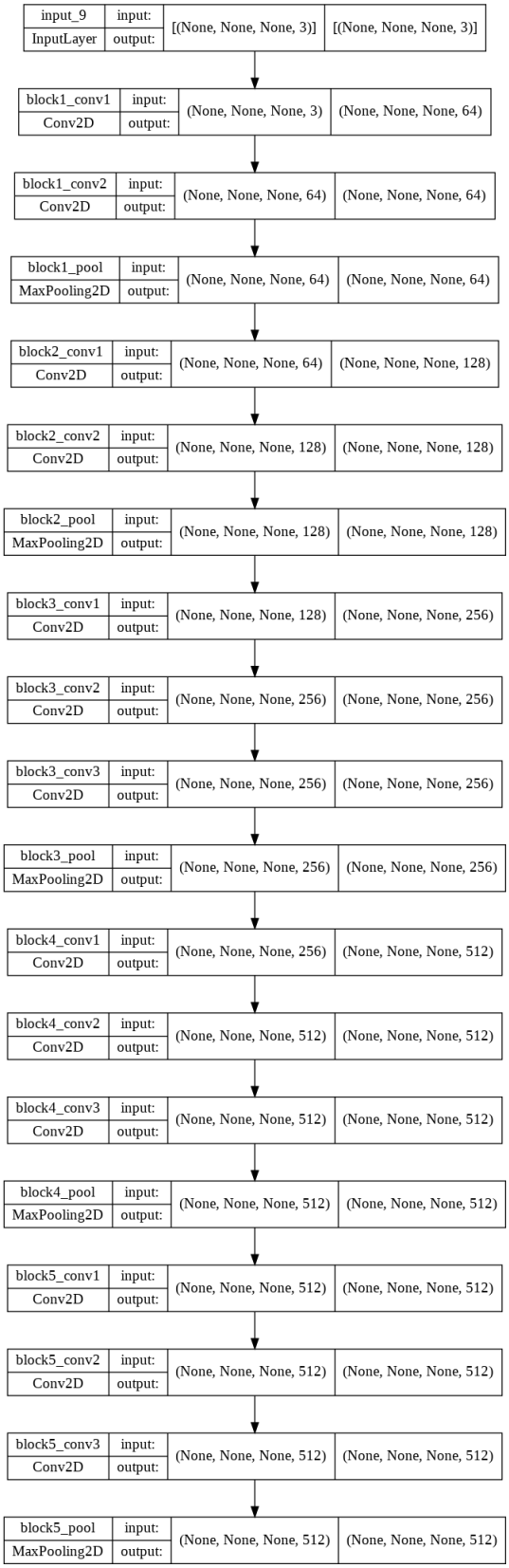


Figura 17 – As 13 camadas da base convolucional da VGG16. As demais 3 camadas densas não são utilizadas para este trabalho. Figuras das demais redes utilizadas neste trabalho não foram incluídas por serem muito extensas.

Fonte: Própria

4.3 Estratégia de treinamento

Os treinamentos foram realizados levando em conta o segundo objetivo específico do trabalho (seção 1.3), que consistiu de treinar os modelos e selecionar o de maior acurácia, de forma que foram treinadas as arquiteturas tanto para o conjunto *Full* quanto para o conjunto *Divided*. Além disso, foram testados regimes de ajuste fino (seção 3.3) tanto com descongelamento total quanto parcial do último bloco convolucional das bases. As redes eram compostas por bases de camadas convolucionais da VGG16, ou da ResNet50, ou da Xception. No topo dessa base, foi colocado um bloco de *Flatten*, uma camada densa de tamanho 256, um bloco de *Dropout* e uma última camada densa com função de ativação sigmoide (seção 3.2). O modelo foi compilado com monitoramento da perda (*loss*) de validação, função de custo de entropia cruzada binária e função de otimização RMSProp (subseção 2.8.2).

Um método de regularização (subseção 3.2.4) consiste em realizar uma interrupção precoce (*EarlyStopping*) [12] do treinamento no modelo quando o custo de validação começa a subir e o modelo atingir ajuste adequado. Treinar mais que o necessário geraria uma perda de generalização (*overfitting*). Este trabalho lidou com a generalização utilizando a configuração disponibilizada pelo Keras de *save_best_only* [13], em que a arquitetura pode ser treinada até *overfit*, mas apenas o modelo com o menor custo de validação será mantido. Sendo assim, o número de épocas escolhidas variou para as três redes, uma vez que o melhor número de épocas foi escolhido deixando o treinamento atingir o estado de sobreajuste e, posteriormente, escolhendo o número com menor custo. Quanto ao tamanho dos lotes, foram escolhidos lotes variando entre 4 e 32 de forma a não extrapolar memória das GPUs. Isso pode ser feito sem implicações nos resultados, uma vez que a função de otimização (RMSProp) tem funcionamento robusto em operação de mini-lotes.

4.4 Incremento de camadas densas

Após etapa de treinamento das redes inicial, a próxima etapa realizada focou no último objetivo específico do trabalho (seção 1.3). Ou seja, foi escolhida a rede que desempenhou melhor em acurácia (seção 4.5) tanto para o conjunto de dados *Full* quanto para o conjunto *Divided*, aumentado seu número de camadas densas. O tamanho e quantidade das camadas densas foi incrementado conforme abaixo:

- 1 camada (original, seção 4.3) - Densa(256) + *Dropout*(0.5)
- 2 camadas - Densa(128) + *Dropout*(0.5) + Densa(256) + *Dropout*(0.5)
- 3 camadas - Densa(128) + *Dropout*(0.5) + Densa(256) + *Dropout*(0.5) + Densa(512) + *Dropout*(0.5)

[12] https://keras.io/api/callbacks/early_stopping/
[13] https://keras.io/api/callbacks/model_checkpoint/

- 4 camadas - Densa(128) + *Dropout*(0.5) + Densa(256) + *Dropout*(0.5) + Densa(512) + *Dropout*(0.5) + Densa(1024) + *Dropout*(0.5)

Observe que os incrementos foram realizados gradativamente com um aumento de camadas em potências de 2. O estágio anterior, de treinar bases convolucionais diversas (seção 4.3), já abrangia uma camada densa de tamanho 256. As demais camadas adicionadas seguiram, conforme a lista anterior, em ordem até a saída da rede. A Figura 18 e a Figura 19 mostram como as redes incrementadas ficam estruturadas desde a base convolucional, nos exemplos expostos, ResNet50, até a saída da rede.

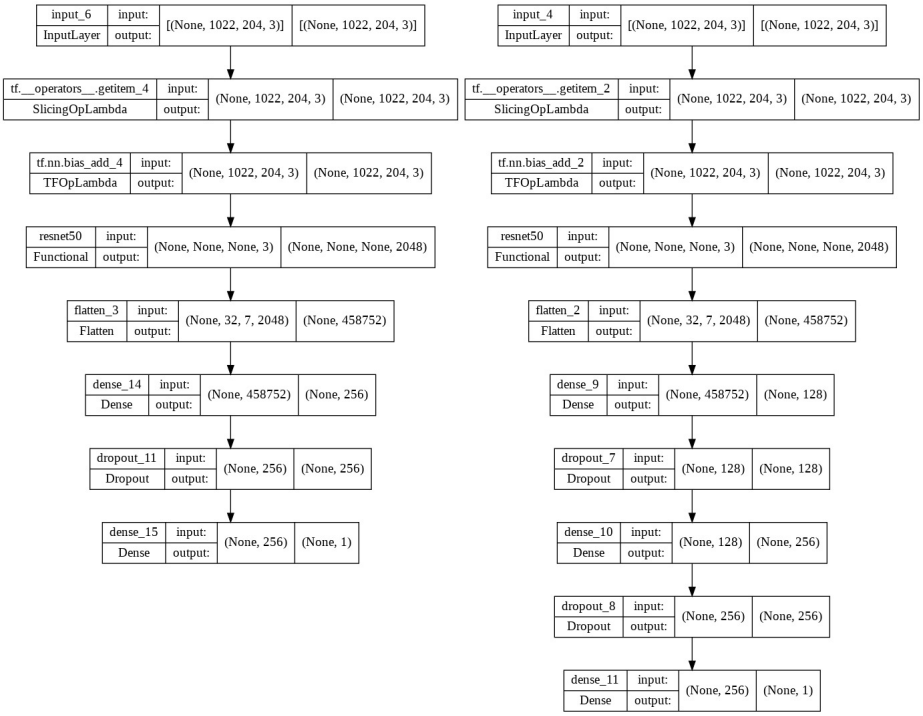


Figura 18 – Exemplos de redes neurais convolucionais com incremento de camadas densas. À esquerda, o modelo original treinado na seção 4.3. O bloco de camadas densas original está entre "dense_14"e "dropout_11". À direita, o modelo incrementado composto de 2 camadas. O bloco de camadas densas está entre "dense_9"e "dropout_8". Lembrando que a última camada densa é a camada com função de ativação sigmoide presente em todos os modelos testados.

Fonte: Própria

4.5 Métodos de Validação dos Resultados

No que tange ser uma classificação binária entre casos saudáveis (*healthy*) e com osteoporose (*osteoporosis*), muitas métricas podem ser usadas para avaliação dos resultados das redes neurais convolucionais. Essa seção trata das métricas utilizadas: Custo, Acurácia, Precisão, *Recall* e *F1-Score*, bem como matrizes de confusão como instrumentos de visualização.

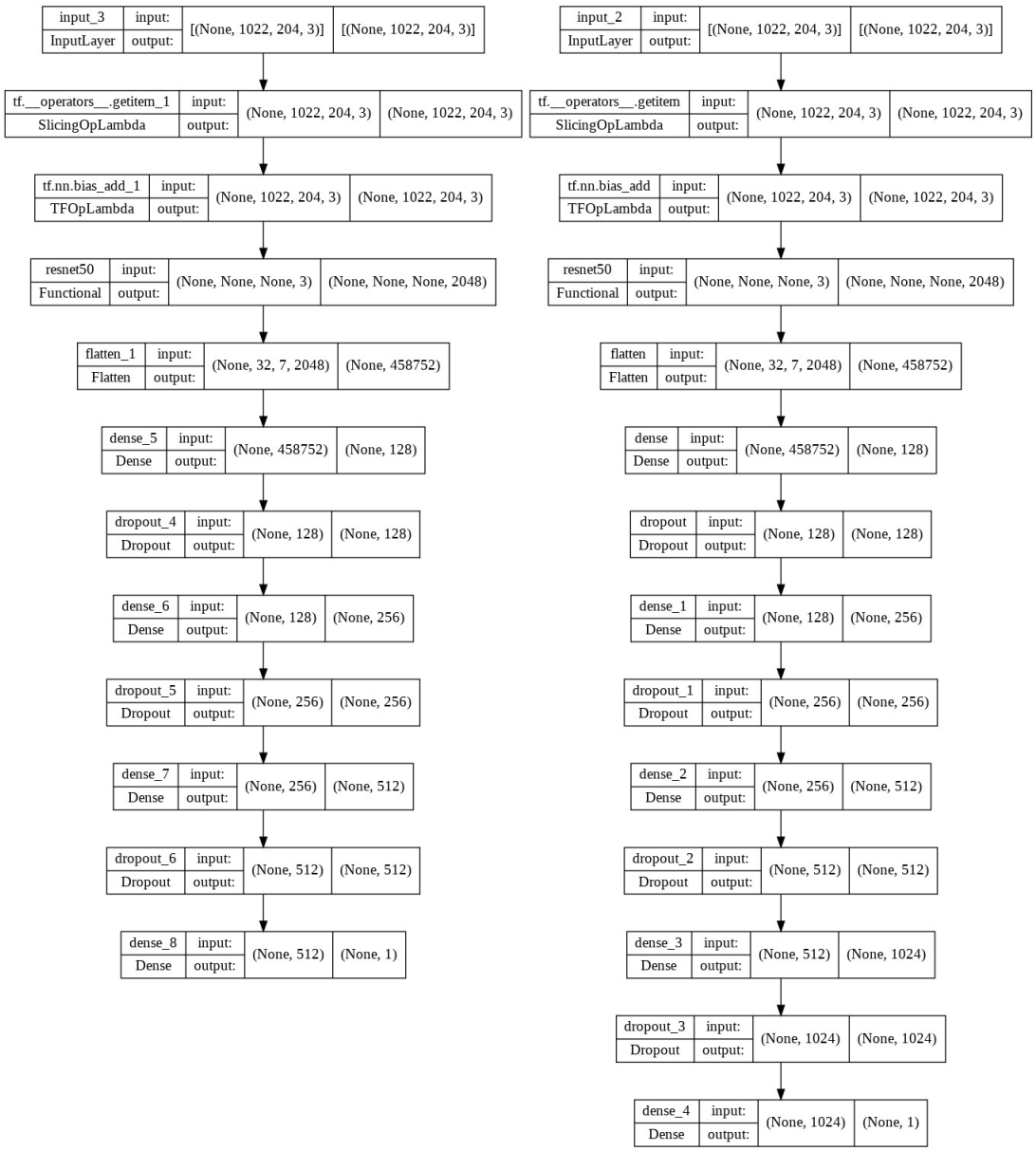


Figura 19 – Exemplos de redes neurais convolucionais com incremento de camadas densas. À esquerda: modelo incrementado composto de 3 camadas. O bloco de camadas densas está entre "dense_5"e "dropout_6". À direita: modelo incrementado composto de 4 camadas. O bloco de camadas densas está entre "dense"e "dropout_3".

Fonte: Própria

A acurácia[14] é a métrica mais importante utilizada deste trabalho. Ela é medida por casos avaliados corretamente divido por casos totais:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}. \tag{4.1}$$

onde TP são os verdadeiros positivos (osteoporose), TN os verdadeiros negativos (sem osteoporose/saudáveis), FP os falsos positivos, ou seja, negativos classificados erroneamente, e FN os falsos negativos, ou seja, positivos classificados erroneamente.

14 https://towardsdatascience.com/a-look-at-precision-recall-and-f1-score-36b5fd0dd3ec

Por sua vez, a precisão é utilizada para mensurar quantas predições positivas de fato estão corretas:

$$\text{Precision} = \frac{TP}{TP + FP}. \tag{4.2}$$

Recall, ou sensibilidade, é a métrica utilizada para avaliar quantas predições positivas estão corretas entre todos os casos positivos:

$$\text{Recall} = \frac{TP}{TP + FN}. \tag{4.3}$$

Finalmente, balanceando a sensibilidade e a precisão em situações de classes de dados desbalanceados, o *F1-Score* combina sensibilidade e precisão em uma média harmônica e é calculada da seguinte forma:

$$\text{F1-Score} = 2\frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{4.4}$$

Essa métrica penaliza desempenhos ruins em qualquer uma das métricas que a compõe, sendo muito útil para a avaliação de desempenho do modelo.

Como definido na seção 2.4, o custo (*Loss*) é uma das métricas monitoradas durante o treinamento. Juntamente com a acurácia (Equação 4.1), o custo é utilizado na decisão de qual rede treinar. O aumento do custo no treinamento indica sobreajuste.

Por fim, a matriz de confusão (seção 5.4) é uma ótima ferramenta de visualização das classificações feitas pela rede neural. Ela apresenta a distribuição dos dados entre as classes reais e as classes previstas, permitindo melhor observação do desempenho do algoritmo.

5 Resultados

5.1 Hardware

O hardware utilizado para o treinamento das redes convolucionais foi fornecido pelo servidor do GPDS[1] da UnB. Foram mais de 110 horas de treinamento e as especificações técnicas do servidor se encontram na Tabela 3.

Tabela 3 – Configurações do servidor do GPDS utilizado para treinamento dos modelos neurais de aprendizado profundo.

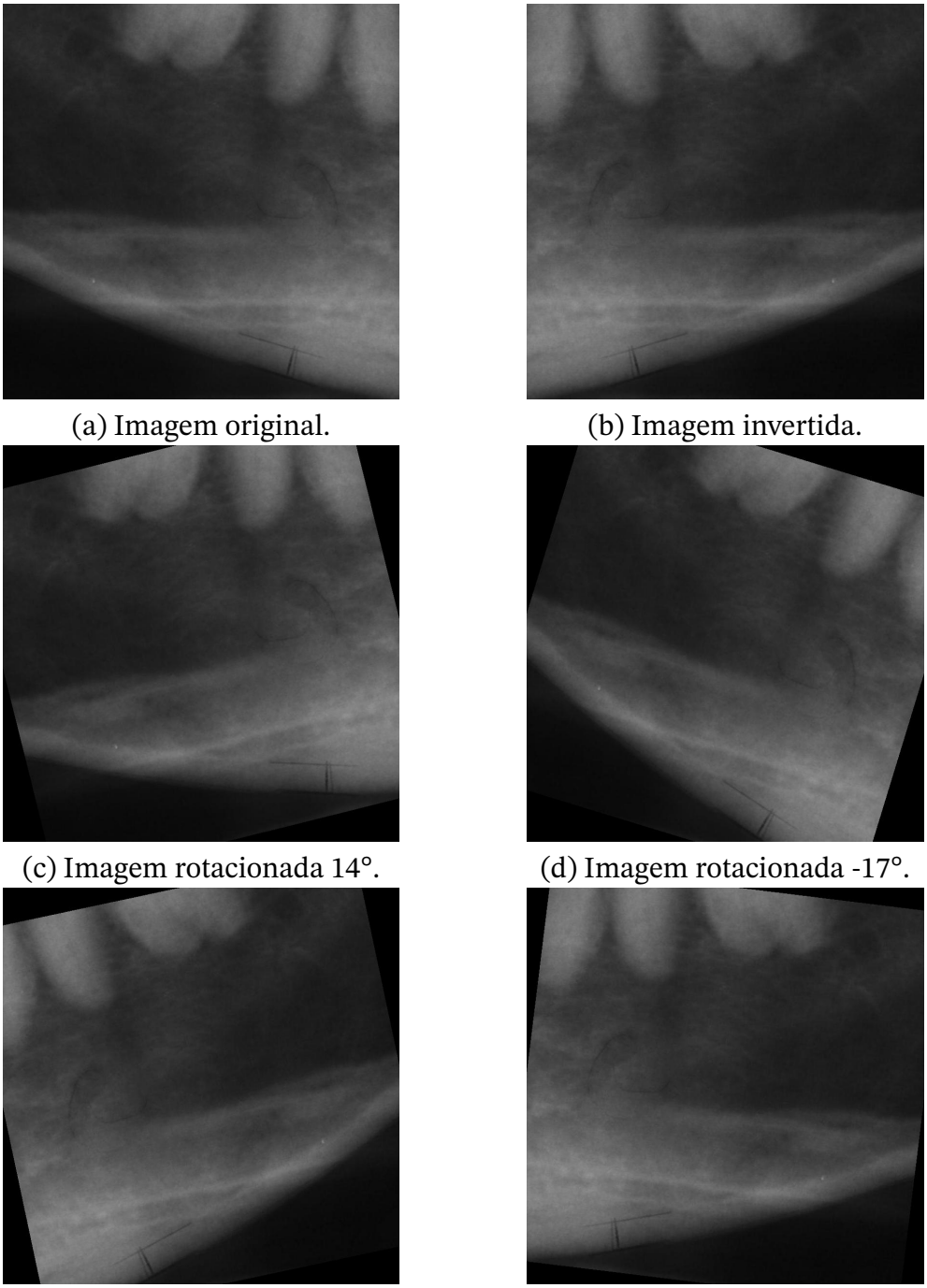
	Modelo
Sistema Operacional	Linux Ubuntu 20.04.3 LTS
Placa de Vídeo (GPU)	1x NVIDIA Quadro P6000 24GB GDDR5X, 2x NVIDIA GeForce GTX 1080 8 GB GDDR5X
Memória RAM	Inacessível via Docker
Processador	32x Intel Xeon CPU E5-2620 v4 2.10GHz
Unidade de Armazenamento	2x Samsung SSD 860 932GB SATA, 1x Seagate Barracuda HD ST4000DM004 3.6TB SATA

Fonte: Própria

5.2 Conjunto de Dados de Imagens

Como explicitado na seção 4.1, os conjuntos *Full* e *Divided* possuem imagens que foram reduzidas de tamanho para 1022x204 pixels e 205x204 pixels, respectivamente. Como também mencionado, o conjunto de dados era considerado pequeno para o treinamento do modelo e, portanto, foram aplicadas técnicas de *data augmentation*. Mais especificamente, foram aplicadas inversões horizontais, logo, no eixo vertical, nos conjuntos de dados e rotações de 5 a 15 graus aleatórias positiva e negativamente. Com isso, a quantidade de dados aumentou em 6 vezes, como poder ser observado no exemplo da Figura 20. Algumas das outras técnicas de aumento do conjunto de dados mencionadas nas seções teóricas não fazem sentido para o escopo do trabalho, portanto não foram utilizadas. A Tabela 4 mostra as quantidades dos dados aumentados e a composição dos grupos de treinamento, validação e teste dos conjuntos *Full* e *Divided*.

[1] http://www.gpds.ene.unb.br/



(a) Imagem original. (b) Imagem invertida.

(c) Imagem rotacionada 14°. (d) Imagem rotacionada -17°.

(e) Imagem invertida e rotacionada 12°. (f) Imagem invertida e rotacionada -7°.

Figura 20 – Exemplos de imagens encontradas no conjunto de dados *Divided*.

Fonte: HUB com modificações próprias

Tabela 4 – Distribuição final do conjunto de dados *Full* e *Divided* em *split* 70-15-15%

Classe		Treino		Validação		Teste		Total
		Healthy	Osteoporosis	Healthy	Osteoporosis	Healthy	Osteoporosis	
full	**Quantidade de Imagens**	709	651	152	139	153	140	1944
	Porcentagem do Total de Imagens	36,47%	33,49%	7,82%	7,15%	7,87%	7,20%	100,00%
divided	**Quantidade de Imagens**	1554	1302	333	279	333	279	4080
	Porcentagem do Total de Imagens	38,09%	31,91%	8,16%	6,84%	8,16%	6,84%	100,00%

Fonte: Própria

Os diretórios dos dados ficaram organizados conforme apresenta-se abaixo:

```
tree /f
data_folder_divided
├───test
│   ├───Healthy
│   └───Osteoporosis
├───train
│   ├───Healthy
│   └───Osteoporosis
└───val
    ├───Healthy
    └───Osteoporosis
```

5.3 Treinamento e validação dos modelos

Como explicado no Capítulo 4, os treinos se dividiram entre algumas épocas para o *feature extraction* e mais algumas épocas para o ajuste fino, ambas até o sobreajuste. A taxa de aprendizado no treinamento de *feature extraction* foi a padrão da biblioteca de 10^{-3} para o RMSProp e para o ajuste fino 10^{-5}. Alguns testes com 10^{-6} foram realizados, mas não se demonstraram bem sucedidos. A escolha do tamanho dos lotes (*batch size*) variou entre 4 e 32 amostras, a depender da GPU alocada para a rede. O funcionamento da RMSProp em mini-lotes diferentes não gerou acurácias finais muito diferentes (subseção 2.8.2), conforme o esperado. Mesmo assim, ocorreram múltiplos testes das mesmas redes com os mesmos hiperparâmetros para garantir isso. Os melhores resultados em configurações diferentes estão listados nas tabelas abaixo.

Como citada na seção 4.4, após realizar testes dos modelos somente com a camada densa de tamanho 256 e a final com ativação sigmoide, seria tomado o modelo com melhor acurácia em ambos os conjuntos de dados para serem acrescentadas mais camadas. Como exposto na Tabela 5 e Tabela 6, para uma camada densa de 256, o melhor desempenho geral foi obtido pela ResNet50. Os resultados dos incrementos de mais camadas só melhoraram o desempenho no conjunto de dados *Divided*, mas não necessariamente no conjunto *Full*.

Tabela 5 – Tabela de melhores acurácias obtidas no conjunto de dados *Full*.

Nº de camadas no topo	Modelo	Melhor acurácia	Descongelamento
1	ResNet50	0,751	parcial
	VGG16	0,696	parcial
	Xception	**0,761**	**parcial**
2	ResNet50	0,730	total
	VGG16	0,724	total
3	ResNet50	0,706	total
4	ResNet50	0,730	total

Nota: Observa-se que o ajuste fino com descongelamento parcial foi mais efetivo somente para a camada densa de 256 sozinha. Nos demais treinamentos, o ajuste fino com descongelamento total obteve resultados superiores ao descongelamento parcial.

Fonte: Própria

A escolha de não incrementar o modelo Xception se deu por não obter bons resultados no conjunto *Divided*, onde raramente obtivera acurácias maiores que 0.650, exceto na única vez que atingiu mais que 0.700 com 0.768 (Tabela 5). Quanto à VGG16, realizou-se a etapa de incrementar uma camada densa de 128, mas a acurácia reduziu em mais de 0.200 no conjunto *Divided* (Tabela 6).

Tabela 6 – Tabela de melhores acurácias obtidas no conjunto de dados *Divided*.

Nº de camadas no topo	Modelo	Melhor acurácia	Descongelamento
1	ResNet50	0,855	total
	VGG16	0,842	total
	Xception	0,768	total
2	ResNet50	0,856	total
	VGG16	0,637	total
3	ResNet50	0,873	total
4	**ResNet50**	**0,882**	**total**

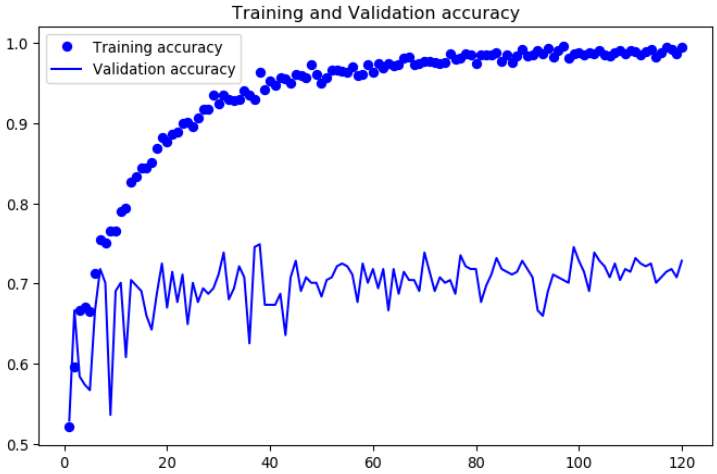
Nota: Observa-se que, para todos os modelos e seus diferentes tamanhos, o descongelamento total durante o ajuste fino foi mais efetivo.

Fonte: Própria

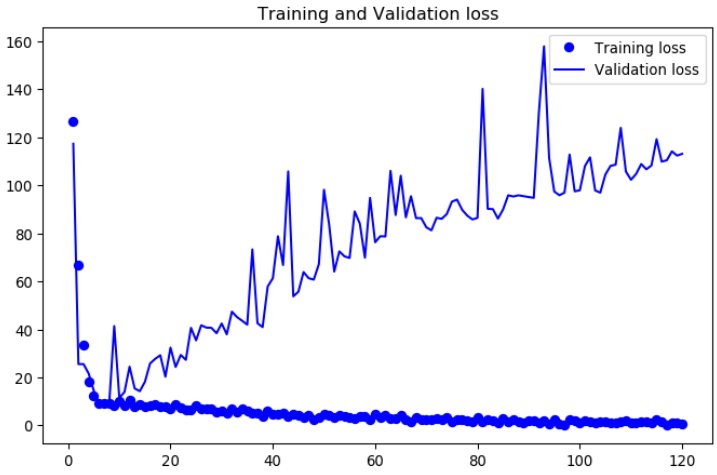
5.4 Perfil dos melhores modelos

De acordo com o que foi visto na seção anterior, os melhores modelos no quesito acurácia foram uma Xception com uma camada densa (256) e uma ResNet50 com 4 camadas densas (128 + 256 + 512 + 1024), nos conjuntos de dados *Full* e *Divided*, respectivamente. Os perfis de custo e acurácia, tanto de treino quanto de validação, ao longo das épocas para *feature extraction* e *fine-tuning* estão dispostos nas figuras a seguir.

A Figura 21 expõe o treinamento em 120 épocas da Xception congelada e a Figura 22



(a)



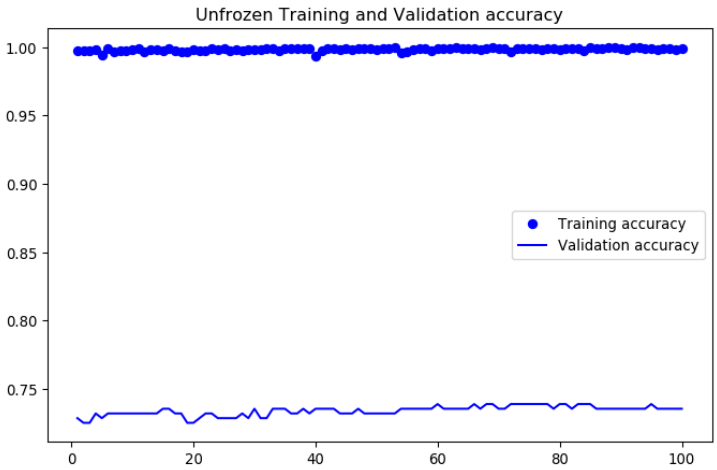
(b)

Figura 21 – *Feature extraction* da Xception. (a) O modelo começa a estabilizar na acurácia de validação por volta de 40 épocas. A acurácia no grupo de testes atinge 0.679. (b) Devido ao alto valor no *loss*, nota-se que o modelo entra em sobreajuste perto de 20 épocas, portanto não são necessárias tantas épocas de treino.

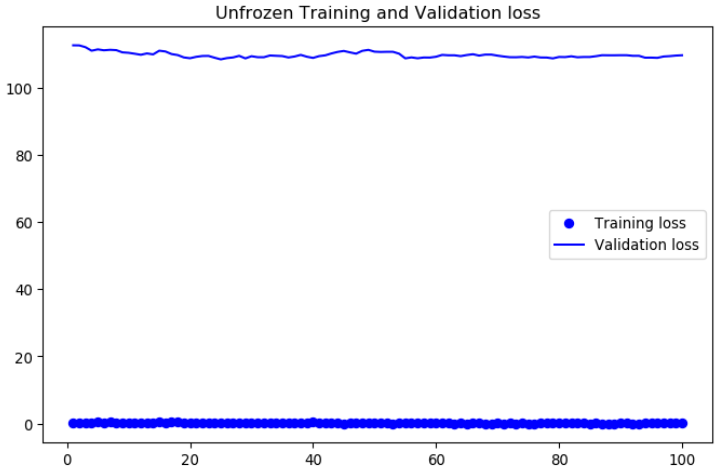
Fonte: Própria

o ajuste fino por mais 100 épocas da mesma rede. O sobreajuste da *feature extraction* ocorre por volta de 40 épocas e o *fine-tuning* se mostra sem efeito. A acurácia final obtida foi de 0.761.

A Figura 23 expõe o treinamento em 80 épocas da ResNet congelada e a Figura 24 expõe o ajuste fino por mais 80 épocas da mesma rede. O sobreajuste da *feature extraction* ocorre por volta de 30 épocas e o *loss* do *fine-tuning* se estabiliza em 50 épocas. A acurácia final obtida foi de 0.882.



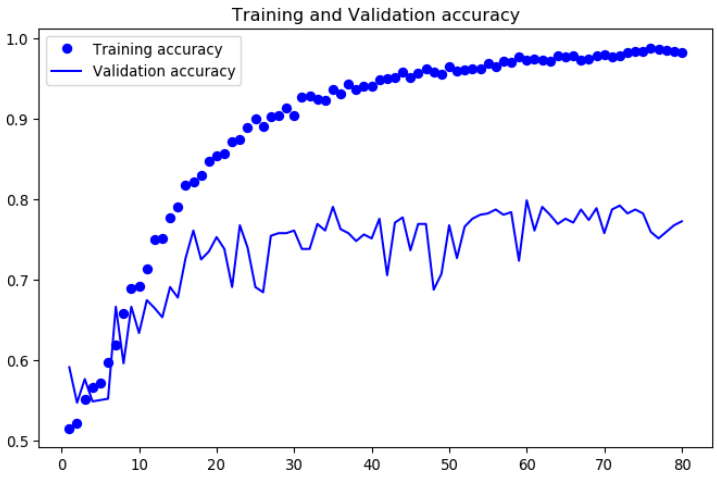
(a)



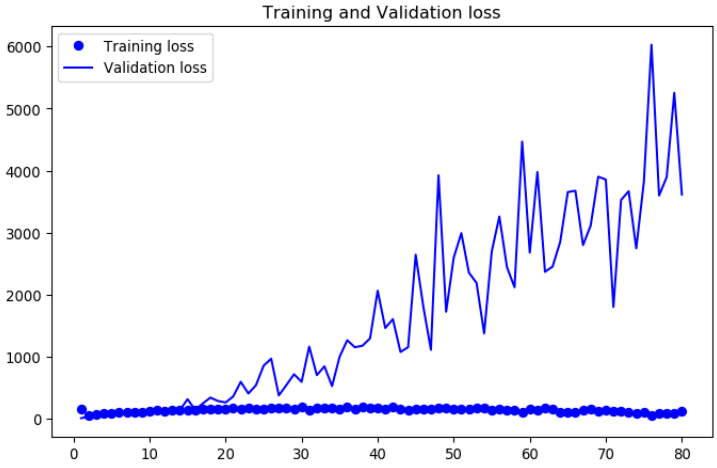
(b)

Figura 22 – *Fine-tuning* da Xception. (a) Acurácia de treino e validação. (b) *Loss* de treino e validação. Nota-se que a acurácia e *loss* praticamente não foram alteradas. O ajuste fino surtiu quase nenhum efeito. A acurácia no grupo de testes atinge 0.761, o que significa um ganho de quase 10% da etapa anterior.

Fonte: Própria



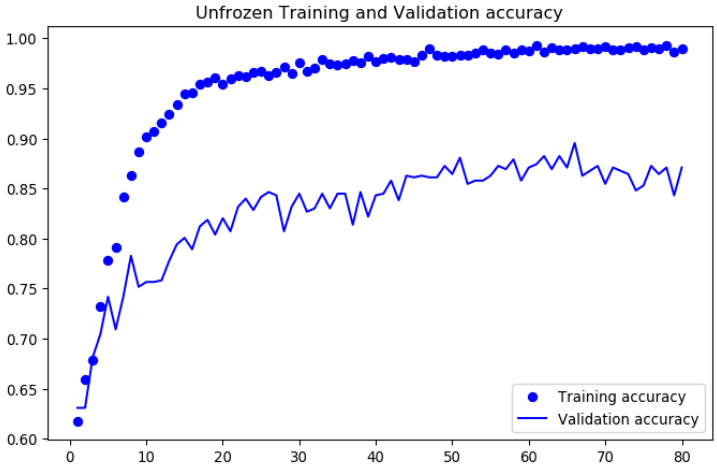
(a)



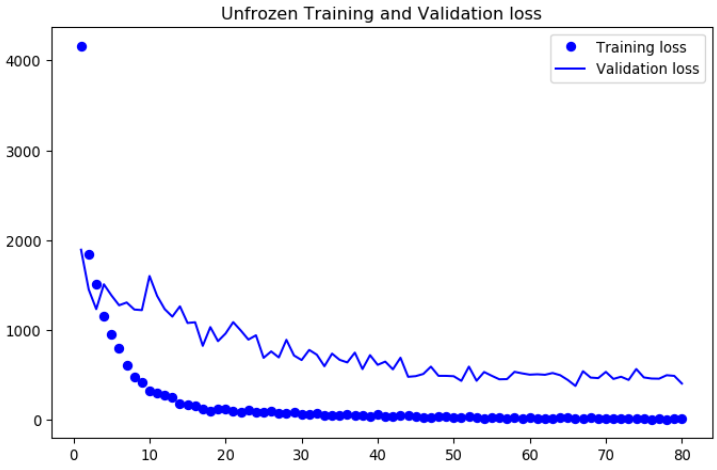
(b)

Figura 23 – *Feature extraction* da ResNet50. (a) O modelo começa a estabilizar na acurácia de validação por volta de 30 épocas. A acurácia no grupo de testes atinge 0.585. (b) Devido ao alto valor no *loss*, nota-se que o modelo entra em sobreajuste perto de 20 épocas, assim como a Xception, portanto não são necessárias tantas épocas de treino. O sobreajuste ocorre mais rapidamente devido a mais camadas densas. Isso pode ser visto na ordem de grandeza do *loss* comparado ao outro modelo.

Fonte: Própria



(a)



(b)

Figura 24 – *Fine-tuning* da ResNet50. (a) Ocorre queda na acurácia no início da etapa, como esperado, pois os pesos da base convolucional são afetados. A acurácia no grupo de testes atinge 0.882, o que significa um ganho de quase 30% da etapa anterior. (b) O *Loss* se estabiliza perto de 50 épocas.

Fonte: Própria

Para avaliação do modelo no conjunto de testes sem interferência do tamanho dos lotes no cálculo das métricas, foi utilizada a função *.predict()*[2], disponível pelo Keras. Essas predições foram utilizadas para geração da matriz de confusão e demais métricas mencionadas na seção 4.5. As matrizes e a tabela de métricas encontram-se abaixo e são elucidadas no manuscrito.

Tabela 7 – Tabela de métricas dos modelos Xception e ResNet50.

Modelo	**Métrica**	**Valor**
Xception	**Acurácia**	0,887
	Precisão	0,890
	Recall	0,871
	F1-Score	0,880
ResNet50	**Acurácia**	0,873
	Precisão	0,882
	Recall	0,832
	F1-Score	0,856

Fonte: Própria

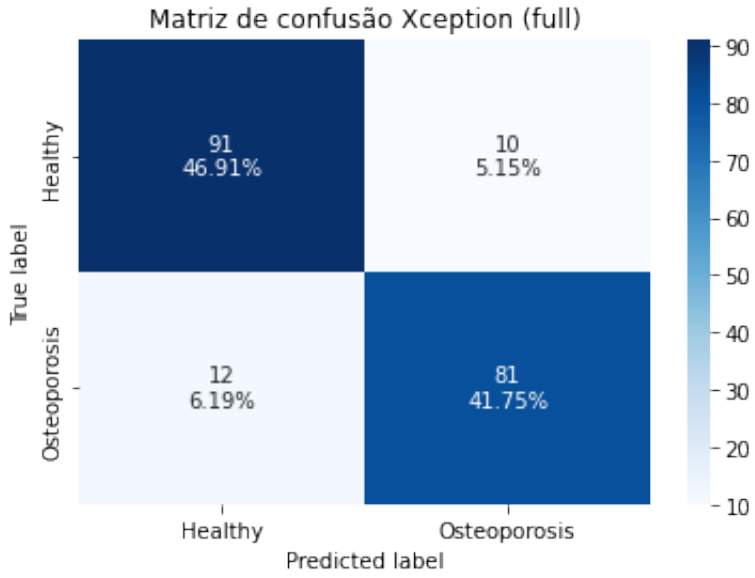


Figura 25 – Matriz de confusão da Xception.

Fonte: Própria

Ante o exposto na Tabela 7, a acurácia obtida para a Xception aumentou no cálculo sem a influência do tamanho dos lotes, até mais que as avaliações das outras redes. Não se esperava obter *F1-Score* tão alto para o número de imagens que o conjunto de dados possuía. As previsões positivas estão corretas em quase 90% das vezes (precisão). A matriz de confusão da Figura 25 foi utilizada para o cálculo das métricas e mostra que ocorreram 12 falsos negativos e 10 falsos positivos.

[2] https://www.tensorflow.org/api_docs/python/tf/keras/Model#predict

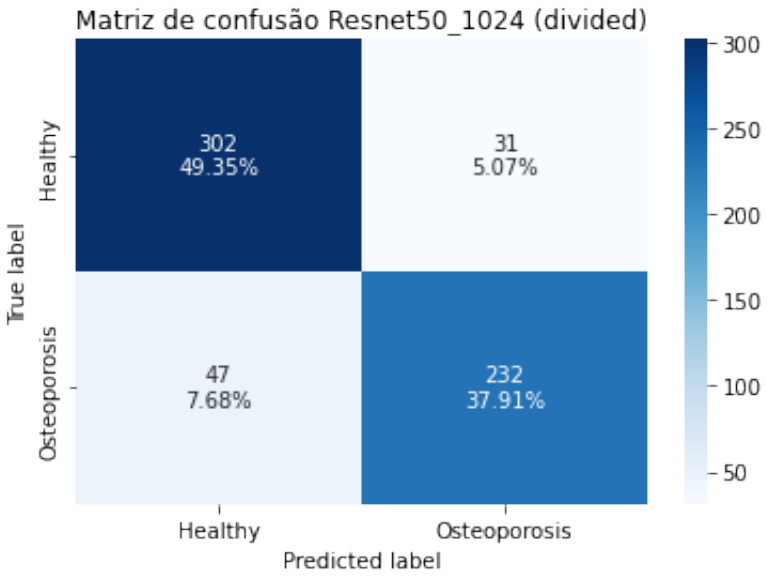


Figura 26 – Matriz de confusão da ResNet50.

Fonte: Própria

Quanto à ResNet50, entre as métricas obtidas, o *recall* foi a mais baixa, indicando maior ocorrência da falsos negativos (7.68%). Isso foi calculado utilizando a Figura 26, que apresenta 47 falsos negativos e 31 falsos positivos. Devido ao *F1-Score* e à acurácia menores que as obtidas com a Xception, segmentar uma região de interesse maxilo-facial, diferentemente do esperado, não melhorou os resultados das classificações.

6 Conclusão

Neste trabalho, foram explorados três modelos de redes neurais convolucionais (*VGG16*, *ResNet50* e *Xception*) com diferentes camadas de profundidades. A expectativa era produzir um classificador comparável ao estado da arte (KARAYIANNI et al., 2007) para detecção de osteoporose em panorâmicas odontológicas. Essa expectativa foi atingida da maneira desejada ao se atingirem resultados similares à literatura e, levando em conta as limitações com o tamanho do conjunto de dados, tornando viável a expansão desse trabalho para projetos futuros mais aplicados.

Para a produção deste estudo, utilizou-se dois conjuntos de dados, um com 340 imagens e outro com 712 imagens. O menor era composto por panorâmicas odontológicas completas e o conjunto maior com regiões de interesse dessas panorâmicas. Mais dados ainda estão sendo coletados via parceira de projetos e mais imagens só tendem a melhorar as métricas das arquiteturas. Para além do escopo do trabalho, em testes iniciais, o aumento em seis vezes promovido por *data augmentaion* ocasionou em melhoria da acurácia de 10% a 15%.

Imagens panorâmicas sem modificações de segmentação pelo usuário puderam ser avaliadas com acurácia de 88.7%. Isto é um bom resultado, comparado a trabalhos anteriores (LEE et al., 2020), considerando que muitas vezes, para detecção de osteoporose, são utilizados diversos exames diferentes para confirmação de profissionais. Quanto a selecionar uma região de interesse para auxiliar o modelo, os resultados não foram melhores que os anteriores com imagens completas. Esperava-se, com o recorte de região de interesse, que as informações consideradas ruidosas não interferissem mais na detecção e a acurácia fosse maior que da panorâmica completa. A acurácia do conjunto de dados com seleção de região de interesse (*Divided*) foi de 87.3%, similar ao conjunto de panorâmicas completas.

Em suma, ressalta-se a importância desse manuscrito de que modelos de redes neurais convolucionais são viáveis para atuarem como mais uma ferramenta de diagnósticos, ainda mais em estudos tão recentes que estão sendo explorados (FRANCIOTTI et al., 2021). Com a conclusão desse trabalho, foi possível obter um modelo detector de osteoporose para radiografias panorâmicas odontológicas maxilo-faciais. Este modelo tem capacidade de classificar corretamente 44 a cada 50 imagens, aproximadamente.

6.1 Perspectivas Futuras

A abordagem desse trabalho é uma entre diversas que podem ser escolhidas para enfrentar o desafio. Existem perspectivas diferentes que podem ser adotadas e espaço para

implementação de melhorias.

Como mencionado, os ganhos obtidos ao realizar *data augmentation* foram notáveis, portanto aumentar o banco de dados com radiografias novas e testar outras técnicas de *data augmentation* se mostram como um caminho promissor. Ainda sobre os conjuntos de dados, foi disponibilizado outro conjunto que não chegou a ser explorado. Essas imagens de 151x151 pixels consistem de regiões de interesse ainda mais específicas, como demonstrado abaixo.

Figura 27 – Exemplo de imagem de 151x151 pixels de conjunto de dados não utilizado.

Fonte: HUB

No que tange a outros modelos de redes, existem modelos que foram criados especificamente para contextos médicos e que não foram possíveis de implementar neste projeto. Apesar de utilizada popularmente para segmentação de imagens a U-Net (RONNEBERGER; FISCHER; BROX, 2015) também vem sendo aplicada para classificação (RAKHLIN; DAVYDOW; NIKOLENKO, 2018). Abordagens com a V-Net (MILLETARI; NAVAB; AHMADI, 2016) e Autoencoders (KRAMER, 1991) também parecem promissoras.

A RMSProp (subseção 2.8.2) se mostrou eficiente em sua otimização, mas outros otimizadores podem ser testados para fins de encontrar ajustes mais rápidos para o problema. A Adam (KINGMA; BA, 2014) é uma alternativa mais popular, mas todos os dias surgem ideias para novos e melhores otimizadores. A AdaBelief (ZHUANG et al., 2020) está nesse grupo.

O escopo deste manuscrito se ateve a encontrar um modelo viável, mas é uma oportunidade de projeto torná-lo um aplicativo acessível ao campo médico real e seus potenciais usuários.

Referências

AZIZIYEH, R.; AMIN, M.; HABIB, M.; PERLAZA, J.; MCTAVISH, R.; LÜDKE, A.; FERNANDES, S.; SRIPADA, K.; CAMERON, C. A scorecard for osteoporosis in four Latin American countries: Brazil, Mexico, Colombia, and Argentina. **Archives of Osteoporosis**, v. 14, dez. 2019. DOI: 10.1007/s11657-019-0622-1. Citado na p. 18.

BURGE, R.; DAWSON-HUGHES, B.; SOLOMON, D.; WONG, J.; KING, A.; TOSTESON, A. Incidence and Economic Burden of Osteoporosis-Related Fractures in the United States, 2005–2025. **Journal of bone and mineral research : the official journal of the American Society for Bone and Mineral Research**, v. 22, p. 465–75, abr. 2007. DOI: 10.1359/jbmr.061113. Citado na p. 18.

CASTRO, J.; CARVALHO, B.; MELO, N.; FIGUEIREDO, P.; MOREIRA, C.; VASCONCELOS, K.; JACOBS, R.; LEITE, A. A new cone-beam computed tomography–driven index for osteoporosis prediction. **Clinical Oral Investigations**, v. 24, set. 2020. DOI: 10.1007/s00784-019-03193-4. Citado na p. 18.

CHOKSI, P.; JEPSEN, K. J.; CLINES, G. A. The challenges of diagnosing osteoporosis and the limitations of currently available tools. **Clinical Diabetes and Endocrinology**, v. 4, 2018. Citado na p. 18.

CHOLLET, F. **Deep Learning with Python**. Manning, nov. 2017. ISBN 9781617294433. Citado nas pp. 21, 24, 25, 28, 32, 37, 40.

CHOLLET, F. Xception: Deep Learning with Depthwise Separable Convolutions. **CoRR**, abs/1610.02357, 2016. arXiv: 1610.02357. Disponível em: <http://arxiv.org/abs/1610.02357>. Citado na p. 43.

FRANÇA, C. C. **Detectando deepfakes em vídeos: uma abordagem utilizando redes neurais convolucionais residuais**. Universidade de Brasília, mai. 2021. P. 106. https://bdm.unb.br/handle/10483/30569. Citado nas pp. 22, 38.

FRANCIOTTI, R.; MOHARRAMI, M.; QUARANTA, A.; BIZZOCA, M.; PIATTELLI, A.; APRILE, G.; PERROTTI, V. Use of fractal analysis in dental images for osteoporosis detection: a systematic review and meta-analysis. **Osteoporosis International**, v. 32, jan. 2021. DOI: 10.1007/s00198-021-05852-3. Citado nas pp. 18, 59.

GHOLAMY, A.; KREINOVICH, V.; KOSHELEVA, O. Why 70/30 or 80/20 Relation Between Training and Testing Sets: A Pedagogical Explanation. In. Citado nas pp. 26, 42.

GODOY, V. D. **Deep Learning with PyTorch Step-by-Step: A Beginner's Guide: Volume I: Fundamentals**. Independently published, jan. 2022. Citado na p. 26.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. MIT Press, 2016. http://www.deeplearningbook.org. Citado nas pp. 21, 28, 35, 36, 38.

HE, K.; ZHANG, X.; REN, S.; SUN, J. Deep Residual Learning for Image Recognition. **CoRR**, abs/1512.03385, 2015. arXiv: 1512.03385. Disponível em: <http://arxiv.org/abs/1512.03385>. Citado na p. 43.

HINTON, G.; SRIVASTAVA, N.; SWERSKY, K. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. **Cited on**, v. 14, n. 8, p. 2, 2012. http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf. Citado na p. 30.

HORNIK, K.; STINCHCOMBE, M.; WHITE, H. Multilayer feedforward networks are universal approximators. **Neural Networks**, v. 2, n. 5, p. 359–366, 1989. Citado na p. 23.

KARAYIANNI, K.; HORNER, K.; MITSEA, A.; BERKAS, L.; MASTORIS, M.; JACOBS, R.; LINDH, C.; STELT, P.; MARJANOVIC, E.; ADAMS, J.; PAVITT, S.; DEVLIN, H. Accuracy in osteoporosis diagnosis of a combination of mandibular cortical width measurement on dental panoramic radiographs and a clinical risk index (OSIRIS): The OSTEODENT project. **Bone**, v. 40, p. 223–9, fev. 2007. DOI: 10.1016/j.bone.2006.07.025. Citado na p. 59.

KINGMA, D.; BA, J. Adam: A Method for Stochastic Optimization. **International Conference on Learning Representations**, dez. 2014. Citado nas pp. 30, 60.

KRAMER, M. A. Nonlinear principal component analysis using autoassociative neural networks. **AIChE Journal**, v. 37, n. 2, p. 233–243, 1991. DOI: https://doi.org/10.1002/aic.690370209. eprint: https://aiche.onlinelibrary.wiley.com/doi/pdf/10.1002/aic.690370209. Disponível em: <https://aiche.onlinelibrary.wiley.com/doi/abs/10.1002/aic.690370209>. Citado na p. 60.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. ImageNet Classification with Deep Convolutional Neural Networks. **Neural Information Processing Systems**, v. 25, jan. 2012. DOI: 10.1145/3065386. Citado na p. 39.

LEE, K.-S.; JUNG, S.-K.; RYU, J.-J.; SHIN, S.-W.; CHOI, J. Evaluation of Transfer Learning with Deep Convolutional Neural Networks for Screening Osteoporosis in Dental Panoramic Radiographs. **Journal of Clinical Medicine**, v. 9, n. 2, 2020. ISSN 2077-0383. DOI: 10.3390/jcm9020392. Disponível em: <https://www.mdpi.com/2077-0383/9/2/392>. Citado nas pp. 19, 59.

MILLETARI, F.; NAVAB, N.; AHMADI, S. V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation. **CoRR**, abs/1606.04797, 2016. arXiv: 1606.04797. Disponível em: <http://arxiv.org/abs/1606.04797>. Citado na p. 60.

PAN, S. J.; YANG, Q. A Survey on Transfer Learning. **IEEE Transactions on Knowledge and Data Engineering**, v. 22, n. 10, p. 1345–1359, 2010. DOI: 10.1109/TKDE.2009.191. Citado na p. 39.

PARETO, V.; BOUSQUET, G.; BUSINO, G. **Cours d'économie politique**. Droz, 1964. (Oeuvres complètes). ISBN 9782600040143. Disponível em: <https://books.google.com.br/books?id=s249oYSJbVMC>. Citado na p. 26.

PARETO, V. **Manual of political economy (manuale di economia politica)**. Tradução: Ann S. Schwier e Alfred N. Page. New York: Kelley, 1971. xii, 504 p. Citado na p. 26.

RAKHLIN, A.; DAVYDOW, A.; NIKOLENKO, S. Land Cover Classification From Satellite Imagery With U-Net and Lovasz-Softmax Loss. In: PROCEEDINGS of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops. Jul. 2018. Citado na p. 60.

RAWAT, W.; WANG, Z. Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review. **Neural Computation**, v. 29, p. 1–98, jun. 2017. DOI: 10.1162/NECO_a_00990. Citado na p. 39.

RONNEBERGER, O.; FISCHER, P.; BROX, T. U-Net: Convolutional Networks for Biomedical Image Segmentation. **CoRR**, abs/1505.04597, 2015. arXiv: 1505.04597. Disponível em: <http://arxiv.org/abs/1505.04597>. Citado na p. 60.

ROSENBLATT, F. The perceptron: a probabilistic model for information storage and organization in the brain. **Psychological review**, v. 65 6, p. 386–408, 1958. Citado na p. 21.

RUDER, S. An overview of gradient descent optimization algorithms, set. 2016. Citado na p. 29.

SIMONYAN, K.; ZISSERMAN, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. **arXiv 1409.1556**, set. 2014. Citado nas pp. 39, 43.

SMITH, L. A disciplined approach to neural network hyper-parameters: Part 1 – learning rate, batch size, momentum, and weight decay, mar. 2018. Citado na p. 27.

SRIVASTAVA, N. Improving Neural Networks With Dropout. U, Toronto, 2013. Master's Thesis. Citado na p. 38.

SRIVASTAVA, N.; HINTON, G.; KRIZHEVSKY, A.; SUTSKEVER, I.; SALAKHUTDINOV, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. **Journal of Machine Learning Research**, v. 15, p. 1929–1958, jun. 2014. Citado na p. 38.

SZEGEDY, C.; LIU, W.; JIA, Y.; SERMANET, P.; REED, S.; ANGUELOV, D.; ERHAN, D.; VANHOUCKE, V.; RABINOVICH, A. Going deeper with convolutions. In. Citado na p. 43.

ZHANG, Y.; DONG, Z.; CHEN, X.; JIA, W.; DU, S.; MUHAMMAD, K.; WANG, S. Image based fruit category classification by 13-layer deep convolutional neural network and data augmentation. **Multimedia Tools and Applications**, v. 78, p. 3613–3632, 2017. Citado na p. 34.

ZHUANG, J.; TANG, T.; DING, Y.; TATIKONDA, S.; DVORNEK, N. C.; PAPADEMETRIS, X.; DUNCAN, J. S. AdaBelief Optimizer: Adapting Stepsizes by the Belief in Observed Gradients. **CoRR**, abs/2010.07468, 2020. arXiv: 2010.07468. Disponível em: <https://arxiv.org/abs/2010.07468>. Citado na p. 60.