Universidade Federal de Pernambuco

Centro de Tecnologia e Geociências

Departamento de Eletrônica e Sistemas

Graduação em Engenharia Eletrônica

Matheus Vieira Barros

Estudo de caso da classificação de tipos de câncer de pele a partir da aplicação de modelos de Redes Neurais Convolucionais.

Recife

2023

Matheus Vieira Barros

Estudo de caso da classificação de tipos de câncer de pele a partir da aplicação de modelos de Redes Neurais Convolucionais.

Trabalho de Conclusão apresentado ao Curso de Engenharia Eletrônica, do Departamento de Eletrônica e Sistemas, da Universidade Federal de Pernambuco, como requisito parcial para obtenção do grau de Bacharel em Engenharia Eletrônica.

Orientador: Prof. Dr. Marcos Antônio Martins de Almeida.

Recife

2023

Ficha de identificação da obra elaborada pelo autor,
através do programa de geração automática do SIB/UFPE

Barros, Matheus Vieira.
Estudo de caso da classificação de tipos de câncer de pele a partir da aplicação de modelos de Redes Neurais Convolucionais. / Matheus Vieira Barros. - Recife, 2023.
72 p. : il., tab.

Orientador(a): Marcos Antônio Martins de Almeida
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Pernambuco, Centro de Tecnologia e Geociências, Engenharia Eletrônica - Bacharelado, 2023.
Inclui referências, apêndices.

1. Câncer de pele. 2. Redes Neurais Convolucionais. 3. Aprendizagem profunda. I. Almeida, Marcos Antônio Martins de. (Orientação). II. Título.

620 CDD (22.ed.)

Matheus Vieira Barros

Estudo de caso da classificação de tipos de câncer de pele a partir da aplicação de modelos de Redes Neurais Convolucionais.

Trabalho de Conclusão apresentado ao Curso de Engenharia Eletrônica, do Departamento de Eletrônica e Sistemas, da Universidade Federal de Pernambuco, como requisito parcial para obtenção do grau de Bacharel em Engenharia Eletrônica.

Aprovado em: 22/09/2023

Banca Examinadora

Prof. Marcos Antônio Martins de Almeida, D.Sc

Universidade Federal de Pernambuco

Prof. João Marcelo Xavier Natário Teixeira, D.Sc

Universidade Federal de Pernambuco

Prof. Sidney Marlon Lopes de Lima, D.Sc

Universidade Federal de Pernambuco

Agradecimentos

Gostaria de expressar minha gratidão a todos que me ajudaram e que contribuiram para que eu pudesse alcançar essa conquista.

Quero agradecer ao meu professor orientador, Professor Marcos Martins, por todos os ensinamentos e direcionamentos ao longo do desenvolvimento deste trabalho assim como durante à minha graduação. Sou bastante grato por todo o conhecimento compartilhado comigo, sua sabedoria e dedicação foram fundamentais para que eu pudesse concluir este trabalho com sucesso.

Quero agradecer aos professores João Marcelo e Guilherme Melo pelo apoio e conhecimento compartilhado comigo longo da graduação, assim como por serem as pessoas que acreditaram e apoiaram desde o início o que se transformou na minha primeira experiência profissional, a Dipolum Consultoria, empresa júnior do DES. Todo suporte e dedicação empregados por cada um dos dois tem contribuição especial no início da minha jornada profissional. Quero agradecer também ao professor Sidney Lima pelo conhecimento compartilhado nas oportunidades que tivemos, seja através das disciplinas, seja através de outras interações ao longo do curso.

Quero agradecer ao meu pai, Robmar, e à minha mãe, Maria José, pois graças a eles me tornei o homem que me tornei e pude alcançar a conquista da graduação. Sou extremamente grato por todo amor e dedicação que tiveram à minha educação, em que todo esse esforço tem este trabalho como fruto desse cuidado. Sem o apoio dos meus pais nada disso seria possível. Esse trabalho é minha homenagem direta a eles e simboliza toda a gratidão que tenho pelos dois.

Quero agradecer à minha namorada e companheira, Maressa, por ser meu lugar de paz e me tranquilizar nos momentos que mais precisei. Obrigado por ficar ao meu lado e me apoiar em todas as minhas decisões, assim como me acolher nos momentos que mais dúvidas tive. Seu apoio e cuidado comigo foi essencial para que eu pudesse iniciar e concluir este trabalho tão importante, que é um marco na minha jornada profissional.

Por fim, quero agradecer aos meus irmãos, Amanda e João Victor, pelos incentivos e apoio que me deram ao longo da minha graduação. O incentivo de cada um dos dois me animava e me motivada a perseguir a conclusão desse ciclo na minha vida. O apoio dos dois fez uma enorme diferença nessa jornada.

À todos vocês, meu muito obrigado.

Resumo

O câncer de pele é uma enfermidade que resulta no surgimento de tumores consequentes de lesões contínuas no DNA de células da pele. O tratamento e cura desse tipo de doença se torna mais fácil caso o diagnóstico ocorra de forma precoce, o que pode poupar seus portadores de transtornos e consequências mais graves. Para o diagnóstico, podem ser empregadas redes neurais a fim de identificar e classificar lesões de câncer de pele registradas em imagens. Assim, este trabalho tem como objetivo realizar um estudo de caso da aplicação de modelos de Redes Neurais Convolucionais para classificação de quatro tipos de lesões resultantes do câncer de pele: Carcinoma Basocelular, Queratose Seborreica, Melanoma e Nevo Benigno. Nesse estudo de caso, os resultados alcançados pelos modelos estudados foram bastante satisfatórios. Diante dos resultados obtidos, são apresentadas linhas de trabalho futuras a partir da aplicação de Redes Neurais para diagnóstico e classificação de câncer de pele, graças à performance dos modelos propostos.

Palavras chave: Câncer de pele; Redes Neurais Convolucionais; Aprendizagem profunda.

Abstract

Skin cancer is a disease that results in tumors occurrence due to continuous damages to the skin cells DNA. Early diagnosis of this type of illness may facilitate easier treatment and cure, which can spare disorders and serious consequences to its patients. For diagnosis, neural networks can be used in order to identify and classify skin cancer lesions recorded in images. Therefore, this study aims to conduct a case study about using Convolutional Neural Networks for the classification of four types of skin cancer lesions: Basal Cell Carcinoma, Seborrheic Keratosis, Melanoma and Benign Nevi. In this case study, the results obtained by the models were quite satisfactory. From the achieved results, future lines of work are presented based in the application of Neural Networks for diagnosis and classification of skin cancer, thanks to the performance of the proposed models.

Key words: Skin Cancer; Convolutional Neural Networks; Deep Learning.

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo 1

Introdução

1.1 Contexto

O câncer de pele se trata de uma enfermidade capaz de gerar uma série de traumas e transtornos aos que acomete. Esta, consequência de diversas microagressões às células da pele ao longo da vida, quando não diagnosticada ou diagnosticada tardiamente, pode resultar inclusive em morte. Sabe-se que um dos principais fatores que resultam no surgimento de câncer de pele é a exposição prolongada e sem proteção aos raios solares [1].

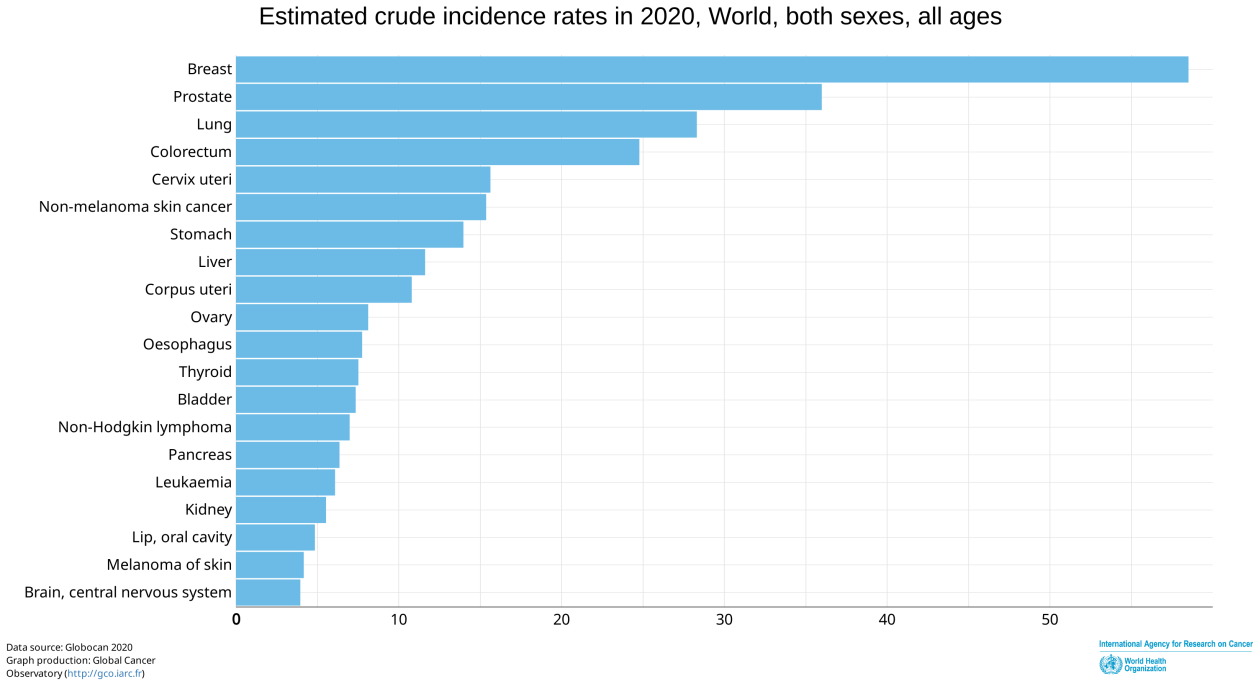
O câncer de pele surge a partir da multiplicação descontrolada de células da pele até a formação de tumores [2]. O câncer de pele pode ser dividido em duas categorias: o câncer de pele melanoma e o câncer de pele não melanoma. O primeiro consiste no câncer que tem origem a partir dos melanócitos, células responsáveis pela produção de melanina; já o segundo tem origem na células basais ou escamosas [3], o qual possui uma taxa de incidência muito maior que o câncer melanoma [1].

Segundo o Instituto Nacional do Câncer, a incidência de raios ultravioletas (UV) é o principal fator de risco para o surgimento de todos os tipos de câncer de pele, já que os raios UV provocam lesões no DNA que se acumulam ao longo do tempo [1]. Ainda segundo o INCA, a ocorrência de câncer de pele ocorre principalmente em pessoas de pele clara com idade acima de 40 anos. Além disso, segundo o Ministério da Saúde, 30% dos diagnósticos de câncer maligno no Brasil são câncer de pele, em que 3% são do tipo melanoma [2].

Segundo dados de 2020 do *Global Cancer Observatory* [4], plataforma da Organização

Mundial de Saúde que dispõe estatísticas sobre o câncer no mundo, o câncer de pele não melanoma ocupou a sexta posição entre todos os tipos de câncer em relação à taxa de incidência global, sendo 15,4 casos a cada 100 mil habitantes. Enquanto isso, o câncer de pele melanoma ocupou a décima nona posição, tendo uma taxa de incidência de 4,2 casos a cada 100 mil habitantes. Esses dados estão presentes na Figura 1.1.

Figura 1.1 – Taxa de incidência por 100 mil habitantes dos diversos tipos de câncer no mundo, dados de 2020.



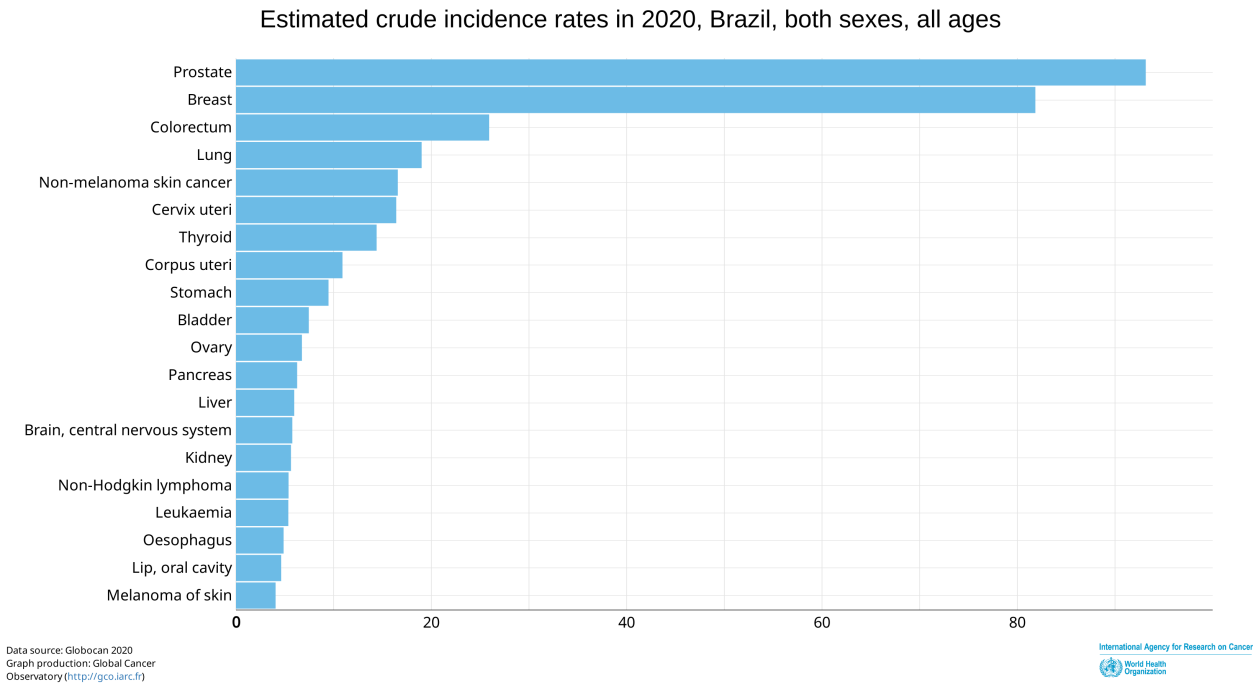
Fonte: Global Câncer Observatory [5]

Ainda segundo o *Global Câncer Observatory*, a taxa de incidência de câncer de pele não melanoma no Brasil foi de 16,6 casos a cada 100 mil habitantes, uma taxa ligeiramente maior que a taxa global. Já o câncer de pele do tipo melanoma, no Brasil, apresentou uma taxa de incidência de 4,1 casos a cada 100 mil habitantes, bastante próxima à taxa global. Essas taxas estão apresentadas no gráfico da figura 1.2.

Cabe ressaltar que o Ministério da Saúde, através do Instituto Nacional do Câncer, projeta 220.490 novos casos de câncer de pele do tipo não melanoma para cada ano do triênio 2023-2025. Esse número de casos corresponde a uma taxa bruta de 101,95 casos para cada 100 mil habitantes, o que equivale a um drástico aumento de 614% em comparação com os dados de 2020. Ainda segundo o Ministério da Saúde, o número

projetado de novos casos de câncer do tipo melanoma para cada ano do mesmo triênio é de 8.980, o que corresponde a uma taxa de 4,13 casos para cada 100 mil habitantes [6].

Figura 1.2 – Taxa de incidência por 100 mil habitantes dos diversos tipos de câncer no Brasil, dados de 2020.



Fonte: Global Câncer Observatory [5]

Em geral, as chances de cura do câncer de pele são altas, desde que o diagnóstico ocorra precocemente e da forma correta. Nesses casos, as chances de cura são da ordem de 95% [7]. Contudo, uma vez identificada uma lesão do tipo câncer de pele, uma etapa importante do processo de diagnóstico é a classificação correta de qual tipo de câncer aquela lesão previamente identificada pertence.

Nesse sentido, a tecnologia tem se mostra como um aliado fundamental. A aplicação de algoritmos e técnicas para classificação de tipos de câncer de pele a partir de imagens de lesões tem apresentado resultados promissores, com novas soluções e modelos de negócio sendo criados no mundo todo. Além disso, a aplicação de novas tecnologias para fins de classificação de câncer de pele tem o potencial de maximizar as chances de cura dessas doenças, uma vez que diagnósticos mais precisos podem ser feitos já nos estágios iniciais.

Entre as tecnologias de maior potencial, cabe um destaque especial para as redes neurais, especificamente as redes neurais convolucionais. Estas têm sido excelentes aliadas

na identificação e classificação de lesões de câncer de pele [8], [9], [10]. A partir de imagens das lesões, a aplicação das redes neurais convolucionais permite alcançar melhores taxas de diagnóstico, principalmente nos estágios iniciais, para assim maximizar as chances de cura da doença e reduzir sua taxa de mortalidade.

Por fim, existe hoje um contexto em que há disponível e de fácil acesso imagens e dados necessários para o desenvolvimento de modelos de redes neurais convolucionais, assim como da ampla disponibilidade de aparelhos capazes de hospedar aplicações com redes neurais convolucionais para classificação de câncer de pele. Dessa forma, juntando os benefícios de todos esses elementos em conjunto, este trabalho tem como principal motivação a possibilidade de estudar os potenciais da aplicação de redes neurais convolucionais na classificação dos tipos de câncer de pele.

1.2 Objetivos

Esta seção trata dos objetivos gerais e específicos deste trabalho, apresentando também o que será proposto e as metas a serem alcançadas. Além disso, é apresentado uma breve apresentação da organização deste trabalho.

1.2.1 Objetivos gerais

Este trabalho tem como objetivo desenvolver um estudo de caso da aplicação de modelos de Redes Neurais Convolucionais para fins de classificação por imagem de lesões de câncer de pele. As lesões selecionadas para este trabalho são o Carcinoma Basocelular, Queratose Seborreica, Melanoma e Nevo Benigno. Essas lesões foram selecionadas pelo fato de o Carcinoma Basocelular ser o tipo de câncer não melanoma mais comum no Brasil [2], pela Queratose Seborreica ser um tipo de lesão benigna bastante comum a partir dos 40 anos de idade [11], pelo o Nevo Benigno ser um tipo de lesão de pele que pode evoluir para o Melanoma [12], sendo o Melanoma o tipo de câncer de pele mais grave [2]. Para realização desse estudo de caso, será analisado o desempenho de três arquiteturas de Redes Neurais Convolucionais distintas: as arquiteturas VGG16, Xception e MCTC (Modelo Classificador de Tipos de Câncer), sendo esta última uma proposta de arquitetura desenvolvida pelo autor. Por fim, os resultados obtidos a partir da aplicação de cada um dos modelos na classificação desses tipos de lesão serão avaliados e comparados.

1.2.2 Objetivos específicos

- Aplicar técnicas de processamento de imagens para treinamento, validação e teste de Redes Neurais Convolucionais;
- Utilizar técnicas de Deep Learning para classificação de imagens;
- Desenvolver arquitetura de um modelo de Rede Neural Convolucional para classificação de imagens, explorando a técnica do *Transfer Learning*;
- Analisar o desempenho de Redes Neurais Convolucionais na classificação de tipos de câncer de pele;
- Obter um desempenho de pelo menos 85% de acurácia nas classificações;

1.3 Organização deste trabalho

Este trabalho está organizado em cinco capítulos. O capítulo 1 apresenta a motivação deste trabalho. O capítulo 2 realiza a fundamentação téorica necessária ao desenvolvimento do estudo de caso. O capítulo 3 apresenta em detalhes a metodologia adotada para o alcance dos objetivos gerais e específicos. O capítulo 4 compila e analisa os resultados obtidos. Por fim, o capítulo 5 apresenta as conclusões do trabalho e propõe linhas de trabalhos futuros.

Capítulo 2

Fundamentação teórica

Este capítulo faz uma síntese da teoria utilizada para o desenvolvimento do trabalho proposto. Aqui, é discutida uma visão de Inteligência Artificial na seção 2.1, discorrendo à respeito de suas definições. Na seção 2.2 é definido o conceito de *Deep Learning* e apresentado as principais estruturas das Redes Neurais. Por fim, a seção 2.3 apresenta os principais elementos relativos ao treinamento de redes neurais, desde os paradigmas de aprendizagem até as principais métricas de avaliação.

2.1 O conceito de Inteligência Artificial

Dentro da comunidade científica, há um grande debate a respeito de uma definição global para termo "Inteligência Artificial". Apesar de a expressão Inteligência Artificial, doravante também referenciada como IA neste trabalho, já existir há bastante tempo, cunhada pela primeira vez em 1956 por John McCarthy na Conferência Dartmouth [13], ainda não se tem um consenso sobre sua definição nos dias atuais. Alguns autores, tais como Wang [14], defendem definições contextuais, que levem em consideração o contexto de aplicação do sistema de IA desenvolvido, assim como dos objetivos do desenvolvimento ou da pesquisa.

Por outro lado, outros autores recomendam definições específicas para o conceito de IA, alguns inclusive extrapolando o conceito para aplicações além de computadores digitais (como Grewal, em [15]). Grewal, em seu artigo, traduz o termo IA como "um sistema mecânico de simulação de coleta de conhecimento e informação e processamento de inteligência do universo: agrupando, interpretando e disseminando estes elementos para os

elegíveis na forma de inteligência acionável".

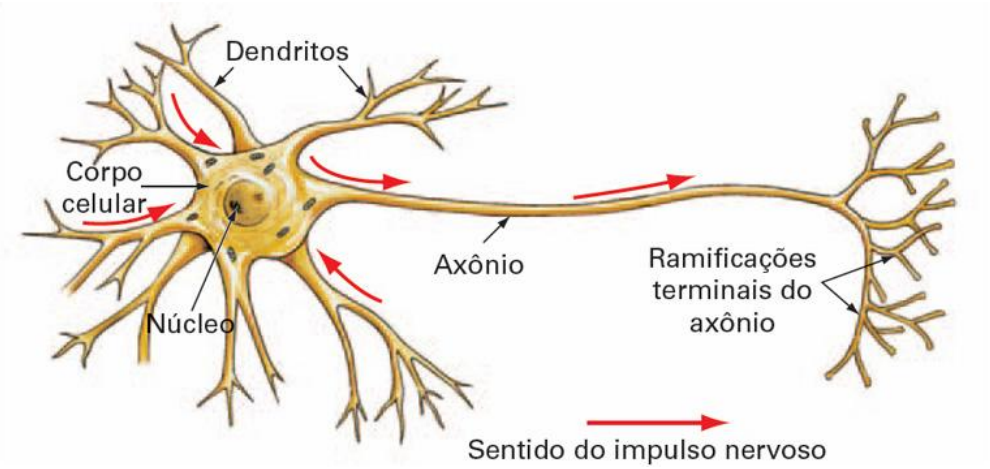
Para fins deste trabalho, não será adotada nenhuma definição formal para o termo Inteligência Artificial. Contudo, assim como Haykins em [16], será assumido que um sistema de IA deve ser capaz de fazer três tarefas: armazenar conhecimento, aplicar o conhecimento armazenado para resolver problemas e adquirir novo conhecimento através da experiência.

2.2 Deep Learning e Redes Neurais

2.2.1 Inspiração neurobiólogica

O *Deep Learning* e as redes neurais têm como inspiração principal o cérebro humano. Este possui como unidade básica de composição o neurônio, cuja estrutura apresenta três elementos principais: Dendritos, Corpo Celular e Axônio. Cada um desses elementos tem o papel de habilitar o cerébro humano a receber, processar e transmitir impulsos elétricos através do sistema nervoso para fins de realização das mais diversas atividades. A Figura 2.1 representa a estrutura de um neurônio.

Figura 2.1 – Representação da estrutura básica de um neurônio. As setas vermelhas mostram o sentido dos impulsos elétricos através do neurônio.



Fonte: Deep Learning Book [17].

Além disso, os neurônios possuem a capacidade de se conectar uns com os outros formando camadas. Essas camadas são construídas através das sinapses, isto é, estruturas funcionais básicas que permitem a interação entre neurônios. As sinapses têm papel

fundamental na capacidade plástica do cérebro, o que permite que o cérebro se desenvolva. Biologicamente falando, o aprendizado ocorre a partir da formação de novas sinapses ou da modificação de sinapses existentes.

Assim, a junção de vários neurônios interligados através de sinapses formam os circuitos cerebrais locais, ou redes neurais. Esses circuitos se constituem por camadas de neurônios responsáveis pela execução de atividades específicas em determinadas regiões do cérebro [16]. A partir da inspiração dos circuitos cerebrais que vem a ideia das redes neurais artificiais organizadas em camadas.

De forma simples e direta, cabe definir o que é uma rede neural artificial. Segundo Haykin [16], uma rede neural artificial é "um processador maciçamente paralelamente distribuído constituído de unidades de processamento simples, que têm a propensão natural para armazenar conhecimento experimental e torná-lo disponível para o uso.". Ainda segundo o mesmo autor, as redes neurais artificiais se assemelham ao cérebro em dois aspectos:

1. A rede adquire conhecimento, através de um processo de treinamento que leva à aprendizagem, a partir do ambiente em que se insere;

2. Os pesos sinápticos são utilizados para armazenar o conhecimento adquirido pela rede, sendo estes pesos as forças de conexão de neurônios de uma rede neural artificial.

Assim, o processo de aprendizagem de uma rede neural artificial funciona a partir do ajuste de seus pesos sinápticos através de algoritmo apropriado, a fim de que esta rede possa realizar determinada tarefa de forma correta [16]. Para realizar o processo de aprendizagem, são utilizados algoritmos específicos que permitem avaliar a eficácia da rede na realização da tarefa em questão e, daí, reajustar os pesos da rede para que esta possa melhorar ainda mais seus resultados.

Dessa maneira, tendo claro o conceito de rede neural artificial e a estratégia adotada para realizar a sua aprendizagem, o *Deep Learning* pode ser entendido como uma subárea da IA a qual se dedica à construção de aprendizado por modelos matemáticos a partir da experiência, isto é, aprendizado a partir da exposição a exemplos [18]. Assim, o *Deep Learning* se debruça na área do conhecimento que visa desenvolver máquinas capazes de

executar tarefas específicas, tomando como inspiração a forma como o cérebro realiza tais atividades.

Um das características do *Deep Learning* é a existência de redes neurais que possuem estruturas chamadas de *hidden layers*, ou camadas ocultas. Essas estruturas são as responsáveis pelos excelentes resultados que as redes neurais artificiais têm alcançado nas mais diversas aplicações de IA. Graças à existência das camadas ocultas, as redes neurais conseguem ter a capacidade de processamento suficiente para extrair informações relevantes na resolução de problemas complexos e não lineares as quais são aplicadas [18].

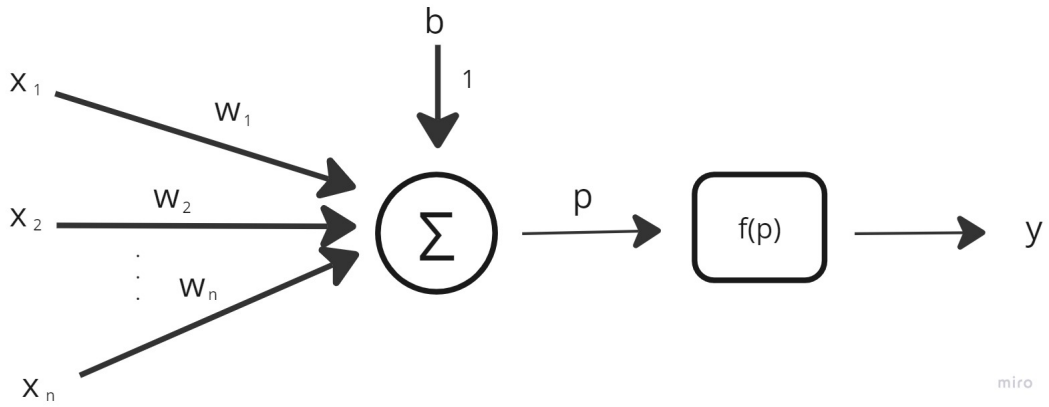
2.2.2 Neurônio artificial

O neurônio artificial se trata do elemento básico das redes neurais e consiste de um modelo matemático que visa representar a função de um neurônio biológico. De forma simples, o neurônio artificial, doravante neste trabalho chamado apenas de neurônio, é composto pelos seguintes elementos: entradas, pesos sinápticos, *bias*, função de ativação e saída. Assim, podemos definir cada um desses elementos da seguinte forma:

- **Entradas:** Correspondem aos atributos dos dados de entrada mais relevantes para resolução do problema para o qual a rede neural está sendo aplicada;
- **Pesos sinápticos:** São valores númericos ajustáveis, positivos ou negativos, que atribuem níveis de importância a cada um dos atributos de entrada dos dados utilizados. Quanto maior o valor absoluto do peso relativo a determinado atributo, maior é a relevância deste atributo na resolução do problema;
- **Bias:** Corresponde ao elemento do neurônio que permite com que a saída seja modificada independente das entradas aplicadas;
- **Somador:** Realiza a soma dos sinais de entrada, ponderados pelos respectivos pesos sinápticos, para aplicação da função de ativação;
- **Função de ativação:** Se trata de uma transformação, linear ou não-linear, aplicada à soma dos sinais de entrada, de forma que o resultado pode ser tanto a entrada de outra camada de neurônios, quanto a própria saída de uma rede neural;
- **Saída:** É resultado obtido a partir do processamento dos dados de entrada do neurônio.

O processo de obtenção da saída de um neurônio a partir das suas entradas funciona da seguinte forma: inicialmente, cada um dos atributos dos dados de entrada são multiplicados pelos respectivos pesos. Posteriormente, os resultados dessas multiplicações são somados entre si (incluindo o *bias*). Essa soma é então aplicada à função de ativação para obter a saída do neurônio para as entradas em questão. O modelo de um neurônio artificial com n entradas está representado na Figura 2.2.

Figura 2.2 – Representação esquemática de um neurônio artificial, o qual apresenta n entradas e uma única saída.



Fonte: O Autor, 2023.

Matematicamente, para um neurônio de n entradas como na Figura 2.2, podemos representá-lo da seguinte forma:

$$p = \left(\sum_{i=0}^{n} x_i * w_i \right) + b \tag{2.1}$$

$$y = f(p) \tag{2.2}$$

em que x_i é a entrada i do neurônio, com $i \in \{1, 2, ..., n\}$; w_i é o peso correspondente a entrada i; p é a soma ponderada das entradas com seus respectivos pesos sinápticos; b é o *bias* associado ao neurônio; f é a função de ativação e y a saída do neurônio.

2.2.3 O Perceptron

O Perceptron consiste de uma estrutura de rede neural a qual, a partir de várias entradas, gera como resultado uma única saída binária [16]. Em outras palavras, considerando

a Figura 2.2, que representa o modelo de um neurônio artificial, este também será o modelo de um Percetron, desde que sua saída possa assumir um entre dois valores possíveis (em geral, 0 ou 1). Matematicamente, o modelo de um Perceptron tem a seguinte forma:

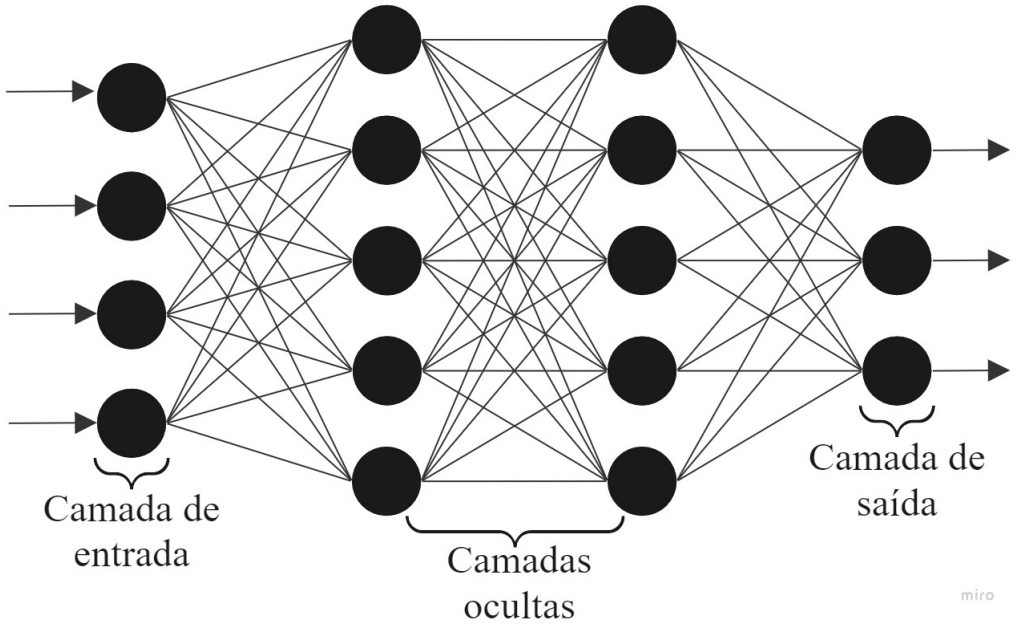
$$y = \begin{cases} 1 & \text{, se } \left(\sum_{i=0}^{n} x_i * w_i\right) + b \geq limiar \\ 0 & \text{, se } \left(\sum_{i=0}^{n} x_i * w_i\right) + b < limiar \end{cases} \tag{2.3}$$

em que x_i é a entrada i do Perceptron, com $i \in \{1, 2, ..., n\}$; w_i é o peso correspondente à entrada i; b é o *bias*, limiar é um parâmetro de referência para realização das classsificações e y é a saída do Perceptron.

Por fim, o Perceptron pode ser defindo como um modelo de neurônio não linear, ou modelo de *McCulloh-Pitts* [16]. Além disso, o Perceptron é também chamado de classificador binário, já que assume um entre dois valores possíveis com base no valor de limiar definido.

2.2.4 Estrutura das redes neurais

Figura 2.3 – Estrutura de uma rede neural. A Figura apresenta a sequência de camadas da rede, sendo estas as camadas de entrada, de saída e as camadas ocultas.



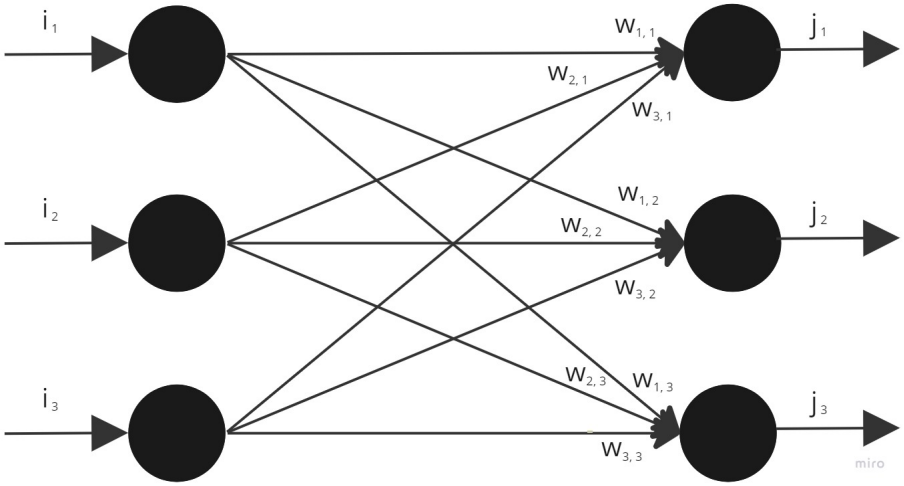
Fonte: O Autor, 2023.

Como mencionado anteriormente, as redes neurais são estruturas compostas por um

conjunto de elementos básicos, chamados neurônios artificiais, interligados entre si. A interligação desses neurônios ocorre no formato de camadas em cascata, apresentando uma característica modular, como forma de mimetizar a estrutura do cérebro humano. Esta estrutura está representada na Figura 2.3.

As camadas são conectadas entre si a partir de um conjunto de pesos sinápticos que conectam as saídas dos neurônios da camada anterior com as entradas dos neurônios da camada seguinte. Assim como para um neurônio isolado, os pesos sinápticos atribuem diferentes graus de relevância a essas conexões a partir dos valores assumidos: quanto maior o valor absoluto do peso sináptico de uma conexão, maior a importância dessa conexão para o resultado final. A Figura 2.4 apresenta os neurônios de uma rede neural conectados entre si a partir de um conjunto de pesos sinápticos:

Figura 2.4 – Representação da conexão entre neurônios de camadas em sequência através dos pesos sinápticos.



Fonte: Adaptado de Nikhil, 2017 [18].

Além disso, é possível segmentar as camadas de uma rede neural em três categorias. A primeira camada é chamada camada de entrada, a qual recebe os dados para processamento pela rede. A camada de saída é a última camada, sendo esta a camada que apresenta o resultado do processamento dos dados aplicados como entrada da rede. Por fim, as camadas intermediárias são chamadas camadas ocultas: essas camadas são as responsáveis por realizar o processamento dos dados de entrada na rede, o que envolve extração de características, reconhecimento de padrões, entre outros.

Por fim, a maneira como os neurônios são conectados entre si tem relação direta com a

performance da rede neural conforme o problema a ser resolvido. Graças a isso, existe um amplo espectro de arquiteturas de redes neurais já desenvolvidas, em que os projetistas podem selecionar aquelas as quais possuem melhor performance com base no problema que se deseja resolver. Algumas das principais arquiteturas são: Redes *Feed-Foward*, Redes Neurais Recorrentes e as Redes Neurais Convolucionais.

As Redes *Feed-Foward* e as Redes Neurais Recorrentes são apresentadas brevemente nesta seção. Já as Redes Neurais Convolucionais estão descritas em detalhes na seção 2.4.5, uma vez que esta é a arquitetura utilizada nos modelos apresentados neste trabalho.

Redes Feed-Foward

As Redes *Feed-Foward*, também denominadas *Multilayer Perceptrons* (de sigla MLP), são redes compostas por camadas de neurônios conectados entre si, em que os dados fluem em um único sentido: da camada de entrada para a camada de saída. Esse tipo de arquitetura pode ser subdividida em duas classes: totalmente conectadas ou parcialmente conectadas.

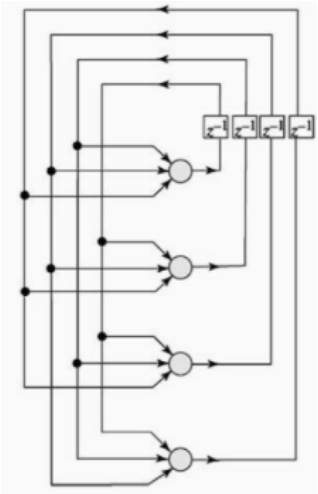
As redes totalmente conectadas possuem todos os neurônios de uma camada conectados a todos os neurônios da camada posterior. Por outro lado, as redes parcialmente conectadas são aquelas em que os neurônios de uma camada não estão conectados a todos os neurônios da camada seguinte. A Figura 2.3 representa um exemplo de uma Rede *Feed-Foward* totalmente conectada.

Redes Neurais Recorrentes

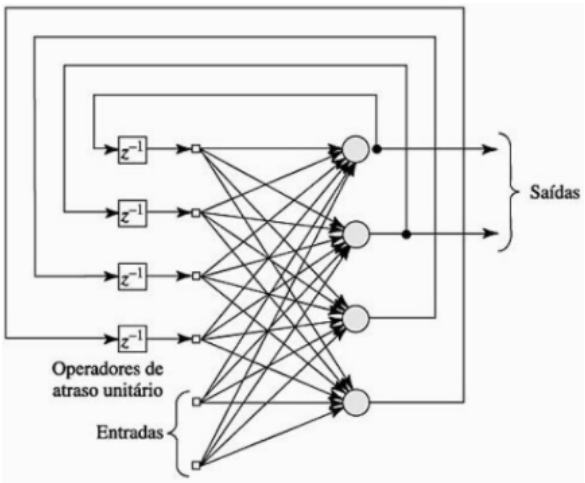
As Redes Neurais Recorrentes são aquelas em que existe pelo menos uma conexão de realimentação em sua estrutura. Esse tipo de arquitetura tem mostrado excelentes resultados quando aplicados em problemas que envolvem áudio, séries temporais e processamento de linguagem natural [19].

Além disso, as Redes Neurais Recorrentes possuem um comportamento dinâmico não linear graças à possibilidade de uso de elementos de atraso unitário em sua estrutura [16]. Em outras palavras, Redes Neurais Recorrentes podem ser pensadas como estruturas que também possuem o tempo como uma variável do modelo. Um exemplo de estrutura de Rede Neural Recorrente está na figura 2.5.

Figura 2.5 – Exemplos de arquiteturas de RNN: sem camadas ocultas na Figura 2.5 (a) e com camadas ocultas na Figura 2.5 (b).



(a)

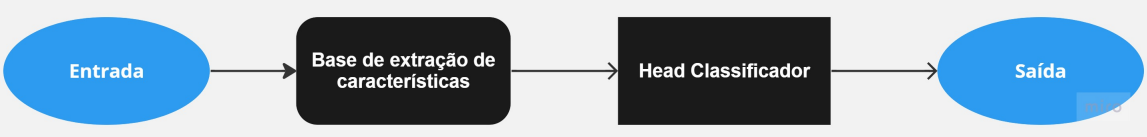


(b)

Fonte: Haykin, 2001 [16].

2.2.5 Estruturas de Redes Neurais Convolucionais

Figura 2.6 – Representação esquemática da estrutura simplificada de uma Rede Neural Convolucional.



Fonte: O Autor, 2023.

As Redes Neurais Convolucionais, ou RNC, são um tipo de arquitetura de rede neural, bastante utilizada em problemas que envolvem imagens, cuja principal característica é a realização de operações de convolução para extração das características de uma imagem. De maneira simplificada, a estrutura de uma rede neural está representada na Figura 2.6.

A Base de Extração de Características é a parte da RNC responsável por extrair as características de uma determinada imagem. Essas características podem ser elementos como bordas, formatos, texturas, iluminações, cores, etc. Contudo, esses elementos podem estar sujeitos a ruídos em imagens distintas, isto é, sujeitos a variações na iluminação, distorções, etc. Por isso, as RNCs não conseguem alcançar 100% de precisão nas classificações. A estrutura da Base de Extração de Características está ilustrada na Figura 2.7.

Figura 2.7 – Representaçao esquemática da estrutura da Base de extração de características de uma RNC.

Fonte: O Autor, 2023.

A extração dessas características é parte primordial nas aplicações que envolvem processamento de imagens, uma vez que permite reconhecer padrões, classificar imagens em classes, identificar objetos numa imagem, entre outras ações. A estrutura da Base conta com camadas de convolução e de *pooling*, assim como com funções de ativação que são utilizadas para realçar as características extraídas das imagens.

Já o *Head* Classificador é a parte da RNC responsável por realizar a classificação das imagens de entrada com base nas características extraídas na Base de Extração de Características. Esse trecho da RNC nada mais é do que uma rede do tipo *Feed-Foward*, cujo exemplo de representação está na figura 2.3.

Camada de Convolução:

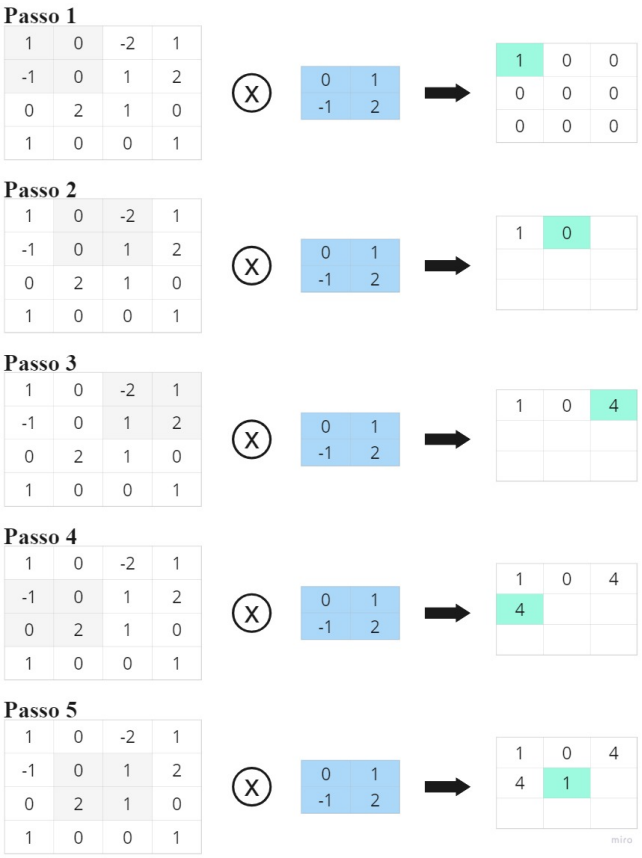
No processo de extração de características de uma imagem, a primeira operação realizada sobre os dados de entrada é a operação de convolução. Essa operação se trata de um tipo de filtragem o qual aplica um filtro à toda área da imagem de entrada para extrair os elementos constituintes dos objetos presentes na imagem [18]. Assim, é possível encontrar padrões espaciais que diferenciem objetos distintos e imagens distintas. Nesse sentido, a filtragem da operação de convolução, realizada pela camada de convolução, ocorre através de filtros chamados *kernels*.

Os *kernels* são matrizes que armazenam os pesos da Base de Extração de Características da RNC durante o processo de extração de características das imagens. No início do treinamento, esses pesos são inicializados, em geral, aleatoriamente e ajustados à medida que o treinamento ocorre para fins de otimização.

A operação de convolução ocorre pela soma ponderada dos pixels da imagem de entrada ponderados pelos pesos do *kernel* utilizado. A Figura 2.8 representa essa operação.

Como exposto na Figura 2.8, o *kernel* percorre todo o tamanho da matriz de pixels

Figura 2.8 – Passo a passo da operação de convolução com um *kernel* de dimensão 2x2 e imagem de dimensão 4x4.



Fonte: Adaptaçao de Gosh, 2020 [20].

da imagem de entrada, realizando somas ponderadas dos pixels pelos pesos do *kernel*. O resultado de cada uma das operações de soma ponderada é colocado numa nova matriz, chamada *feature map*. Os *features maps* nada mais são do que o resultado da aplicação dos *kernels* à matriz de pixels das imagens de entrada para realização da extração de características das imagens. A dimensão do *feature map* obtido após a convolução depende especialmente de dois parâmetros da operação de convolução: *stride* e *padding*.

O *stride* corresponde ao número de passos dado pelo *kernel* dentro da matriz de pixels da imagem para obter cada uma das somas ponderadas. No exemplo acima, o *stride* assumiu o valor (1,1), isto é, um passo na horizontal e um passo na vertical ao longo da matriz de pixels para obtenção da soma ponderada.

Por outro lado, o *padding* é utilizado para tratar os pixels da borda da matriz de pixels da imagem, assim como ajustar a dimensão da matriz de saída. Para esse parâmetro, existem duas possibilidades:

- **Primeira possibilidade:** Considerar o *kernel* apenas dentro da matriz de pixels da imagem, de forma que o *feature map* obtido se torna reduzido;
- **Segunda possibilidade:** Aumentar em uma unidade cada uma das dimensões da matriz de pixels da imagem original e preencher com pixels nulos a linhas e colunas adicionadas.

Assim, é possível calcular a dimensão do *feature map*, elemento resultante da operação de convolução na matriz de pixels da imagem, através das seguintes expressões:

$$y' = \left\lfloor \frac{y - k + p}{s} + 1 \right\rfloor \tag{2.4}$$

$$x' = \left\lfloor \frac{x - k + p}{s} + 1 \right\rfloor \tag{2.5}$$

Em que y' é altura do *feature map*, x' o comprimento do *feature map*, y a altura da imagem de entrada, x o comprimento da imagem de entrada, k é o tamanho do *kernel*, p é o tamanho do *padding* da convolução e s o tamanho do *stride* da convolução [18].

Função de ativação

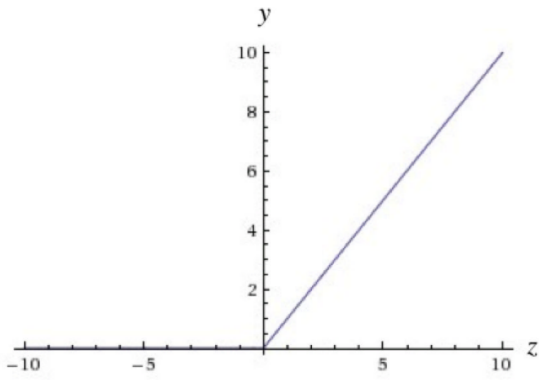
A função de ativação, aplicada aos *feature maps* obtidos na operação de convolução, tem o papel de destacar as características mais relevantes da imagem de entrada. Basicamente, a função de ativação pode ser vista como quem executa o papel de medir a importância dos pixels de uma imagem a partir de um certo critério. Além disso, segundo Haykin [16], a função de ativação restringe a amplitude do sinal de saída de um neurônio, em que, em geral, o intervalo típico da amplitudade do sinal de saída é algum entre (0, 1) ou (-1, 1).

A principal característica das funções de ativação é a sua não linearidade. Dada a complexidade nos padrões das características extraídas das imagens, a característica não linear dessas funções as tornam extremamente úteis para destacar os principais elementos que compõem as imagens. Dessa maneira, entre as funções de ativação mais comuns, pode-se destacar três: a função ReLU, a função sigmoíde e a função tangente hiperbólico.

A função de ativação ReLU assume o valor zero quando sua entrada apresenta valores negativos e valores crescentes positivos quando sua entrada é positiva. Dentro do contexto da engenharia, a função ReLU tem sido amplamente adotada em aplicações que envolvem

visão computacional [18]. Como mostra a sua curva presente na Figura 2.9, a função ReLU não apresenta um limite superior.

Figura 2.9 – Curva da função ReLU, cujo intervalo de saída varia de 0 a z, sendo z o valor de entrada da função.



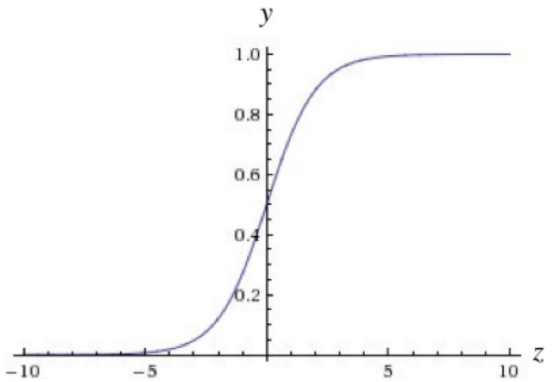
Fonte: Nikhil, 2017 [18].

Matematicamente, sua expressão é dada por:

$$f(z) = max(0, z) \tag{2.6}$$

A função sigmoide é uma função contínua cujo intervalo de valores de saída está dentro do intervalo (0, 1), isto é, apresenta limites inferior e superior, diferentemente da função ReLU. Esta é uma função estritamente crescente, em que sua curva característica está representada na Figura 2.10.

Figura 2.10 – Curva característica da função Sigmoide, para a qual o intervalo dos valores saída é (0, 1).



Fonte: Nikhil, 2017 [18].

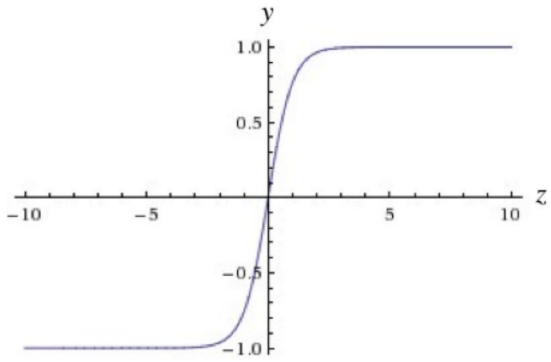
Matematicamente, sua expressão é dada pela seguinte exponencial:

$$f(z) = \frac{1}{1 + \exp\{-az\}} \tag{2.7}$$

Na expressão da função sigmoide, o parâmetro a representa a inclinação da função. Além disso, a função sigmoide, diferentemente da função ReLU, é uma função diferenciável, característica bastante importante na teoria de redes neurais [16], uma vez que os algoritmos envolvidos no treinamento de redes neurais artificiais empregam uma série de operações de diferenciação.

Por fim, a função tangente hiperbólico é um tipo de função de ativação que possui as seguintes características: é uma função ímpar, também contínua e, portanto, diferenciável, mas com intervalo de saída sendo (-1, 1). Em algumas aplicações, é interessante dispor de funções de ativação, tais como a função tangente hiperbólico, cujos valores de saída possam assumir valores negativos, o que gera benefícios analíticos [16]. A curva da função tangente hiperbólico está na Figura 2.11.

Figura 2.11 – Curva característica da função Tangente Hiperbólica cujo intervalo dos valores de saída é (-1, 1).



Fonte: Nikhil, 2017 [18].

Matematicamente, a expressão da função Tangente Hiperbólico é dada pela Equação 2.8:

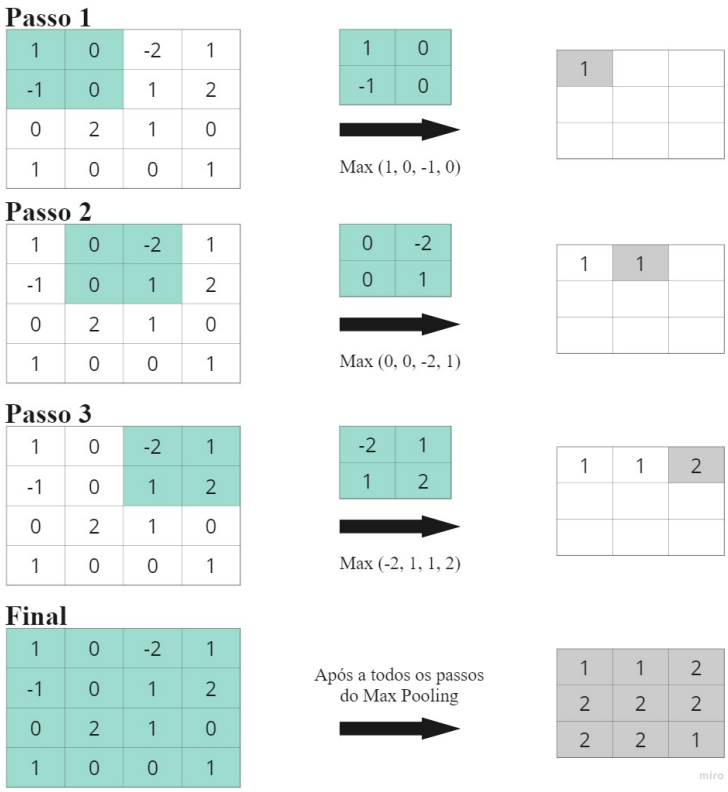
$$f(z) = tangh(z) \tag{2.8}$$

Camada de pooling

Após a aplicação da função de ativação aos *feature maps*, a etapa final consiste em reduzir a dimensionalidade dos *feature maps* obtidos e em consolidar apenas os pixels que contribuem de fato para as características da imagem, desconsiderando aqueles que não têm relevância nessa contribuição [20]. Essa operação é chamada de *pooling*. O objetivo da etapa de *pooling* é acelerar o processamento da rede neural [21], retirando os pixels que não contribuem o suficiente para extração de informação útil pela rede.

Na operação de *pooling*, os pixels ativados pertencentes a uma determinada região do *feature map* são substituídos por uma métrica específica dessa região [21]. Entre as métricas mais utilizadas, tem-se a substituição dos pixels de uma determinada região pelo pixel de maior valor ali presente, operação chamada *max pooling* [20]. Além disso, tem-se também as operações de substituição pela média dos pixels, pela norma L2 ou pela média ponderada baseada na distância de cada pixel ao pixel central. A Figura 2.12 abaixo exemplifica a execução do *pooling*.

Figura 2.12 – Passo a passo da operação de *pooling* de dimensão 2x2 utilizando a métrica *max pooling*.



Fonte: Adaptado de Gosh, 2020 [20].

Na Figura 2.12, nota-se que o tamanho da região considerada dentro do *feature map* para aplicação da operação de *pooling* é uma matriz (2x2). Essa dimensão é uma métrica chamada *pool size*. Além disso, a métrica adotada para realização do *pooling* é o *max pooling*, uma vez que os pixels de cada região sob operação de *pooling* no *feature map* são substituídos pelo pixel de maior valor.

Vale destacar que a operação de *pooling* traz para a rede neural uma propriedade de invariância à translação das características de uma imagem [22]. Em outras palavras, a operação de *pooling* permite identificar certas características de um determinado tipo de objeto (como um carro, por exemplo) independentemente de onde essas características estejam localizadas em diferentes imagens.

2.3 Treinamento de redes neurais

O treinamento de redes neurais é a etapa pela qual o modelo construído é exposto a um conjunto de dados a fim de que possa performar determinada tarefa em um ambiente específico. A partir do treinamento, a rede neural se torna apta a realizar tarefas como classificações, reconhecimento de padrões, detecção de movimentos, análise de sentimentos, entre outras várias aplicações.

Assim, Aprendizado de Máquina é a área de estudo que se debruça em desenvolver, avaliar e aprimorar técnicas de treinamento de modelos de Inteligência Artificial para aplicações em ambientes reais. Segundo Raschka [23], Aprendizado de Máquina pode ser definido "como a subárea da Inteligência Artificial que envolve o desenvolvimento de algoritmos de auto-aprendizado para aquisição de conhecimento, a partir de dados, a fim de realizar predições".

No processo de treinamento, ao ser exposto a um conjunto de dados, o modelo adquire conhecimento para realização de uma tarefa em particular. O conhecimento, segundo Fischler [24], "se refere à informação armazenada ou a modelos utilizados por uma pessoa ou máquina para interpretar, prever e responder apropriadamente ao mundo exterior".

Uma vez adquirido conhecimento, uma etapa mandatória no processo de desenvolvimento de redes neurais é o teste do modelo antes da sua aplicação em ambiente real. A avaliação do conhecimento adquirido por um modelo se dá através da análise de métricas que permitem verificar a sua performance na realização da tarefa para a qual foi treinado.

2.3.1 Paradigmas de aprendizagem

A partir da definição do Aprendizado de Máquina, existem alguns paradigmas que auxiliam os cientistas e engenheiros a escolher a metodologia de aprendizado mais adequada para o problema que se pretende resolver. Os paradigmas são:

1. **Aprendizado supervisionado:** O aprendizado supervisionado consiste em apresentar ao modelo um conjunto de dados de treinamento já previamente categorizados para que a máquina possa aprender quais características de cada um desses dados são relevantes a fim de que este seja categorizado entre as classes disponíveis [16]. Esse tipo de paradigma é amplamente adotado em problemas de classificação, por exemplo;

2. **Aprendizado não-supervisionado:** O aprendizado não-supervisionado é o paradigma no qual o modelo é apresentado a um conjunto de dados de treinamento não categorizados. Assim, o papel do modelo é identificar quais são as características que identificam os dados presentes na base de treinamento, a fim de que possa realizar tarefas como agrupamentos e reconhecimento de padrões [16];

3. **Aprendizado por reforço:** A aprendizagem por reforço é uma técnica de aprendizagem que lança mão das relações de causa e efeito em relação às decisões de um agente dentro de um determinado ambiente. O agente, neste caso, é um modelo que aprende através de tentativa e erro em busca de minimizar um determinado índice escalar de desempenho. Assim, em caso de acerto o valor desse índice é reduzido e em caso de erro, o valor do índice de desempenho aumenta [16].

2.3.2 Base de dados

O treinamento de redes neurais assume como princípio norteador o aprendizado por experiência. Para isso, é necessário que a rede neural seja de alguma forma exposta a situações a qual ela precisa aprender como se comportar para performar de maneira satisfatória. Esse princípio se fundamenta na capacidade humana de aprender através da experiência, com o objetivo de replicar para máquinas a mesma capacidade.

Para este fim, a estratégia utilizada de construir conhecimento numa rede neural é através da sua exposição a uma base de dados que materialize uma situação real, na

qual se espera que esse modelo assuma algum tipo de comportamento. Por exemplo, em problemas que envolvem classificação de dados, tema macro deste trabalho, um modelo de rede neural é exposto durante o processo de treinamento a amostras que exemplifiquem as classes e as características dos elementos que fazem parte de cada classe. Dessa maneira, este modelo consegue identificar padrões nos dados de cada uma das classes e, assim, classificar novos dados aos quais é exposto nas classes identificadas durante a etapa de treinamento.

Dentro dessa estratégia, o principal ponto avaliado ao treinar um modelo é sua capacidade de generalização. Segundo Haykin [16], a capacidade de generalização consiste na capacidade de uma rede neural de obter as saídas corretas para entradas as quais a rede não teve contato durante o treinamento.

Assim, o conjunto de dados que se tem disponível é dividido em três bases distintas: base de treinamento, base de validação e base de testes. Em geral, essas bases são divididas numa proporção de 70% dos dados para base de treinamento, 20% para validação e 10% para teste, podendo ocorrer variações. Abaixo está uma descrição de cada base e seu papel no treinamento de redes neurais.

- **Base de Treinamento:** Uma base de dados de treinamento é utilizada para auxiliar a rede neural a reconhecer padrões e elementos necessários na resolução do problema em questão;
- **Base de validação:** A base de validação é utilizada para avaliar a capacidade de generalização da rede neural durante o processo de treinamento. A base de validação, cabe salientar, é composta por dados distintos dos dados da base de treinamento;
- **Base de Teste:** A base de teste é composta por dados nunca apresentados à rede neural, a fim de avaliar a capacidade de generalização do modelo após o processo de treinamento.

2.3.3 Otimizador, função custo e taxa de aprendizagem

O otimizador é o algoritmo a partir do qual o modelo consegue reajustar seus parâmetros durante o processo de treinamento. A aprendizagem de uma rede neural acontece através do ajuste dos pesos de cada um dos neurônios, até se obter como resultado a saída

esperada para a maior parte possível dos dados utilizados como entrada durante o processo de treinamento. O ajuste dos pesos ao longo do treinamento ocorre através de um otimizador, que realiza o ajuste com o objetivo de minimizar a função custo. Por outro lado, o papel da função custo é informar ao modelo se o resultado obtido na sua saída se aproxima do resultado esperado, assim como quão distante estão o resultado esperado e o resultado obtido pelo modelo.

Em geral, para otimização em redes neurais, uma família de otimizadores bastante utilizada é a dos chamados *Gradient Descent*. O *Gradient Descent* é uma família de algoritmos que empregam o gradiente da função custo em relação aos pesos da rede neural para obterem qual o próximo conjunto de pesos que minimiza ao máximo o valor dessa função. Esses algoritmos possuem distintas abordagens de implementação, levando em consideração tempo de computação e eficiência de ajuste dos pesos para obtenção da função custo mínima. Ao final de cada época, em que uma época consiste numa única iteração sobre todos os dados da base de treinamento durante o processo de treinamento, é calculado o valor da função custo durante a iteração em questão.

Graficamente, a cada época, o *Gradient Descent* calcula a superfície da função custo em relação aos pesos e obtém o vetor gradiente no ponto que representa o conjunto de pesos da rede neural. O negativo do vetor gradiente corresponde à direção para a qual o algoritmo deve reajustar os pesos para minimizar ao máximo o valor da função custo. Esse processo é realizado iterativamente, época a época, até se encontrar o menor valor possível da função custo na superfície em particular. A Figura 2.13 representa esse passo a passo.

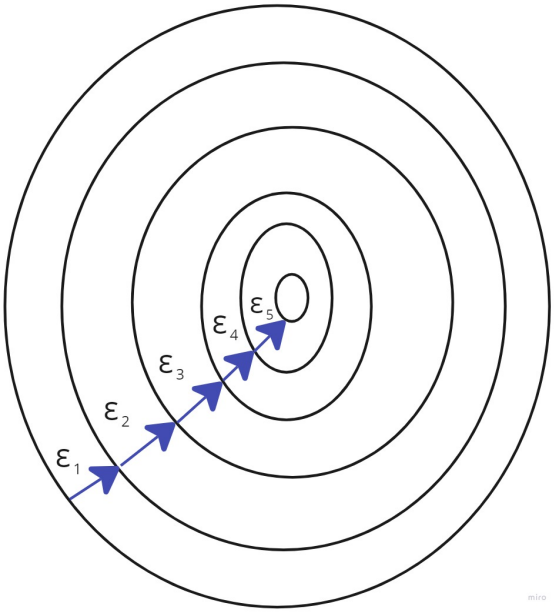
A Figura 2.13 mostra também que a dimensão do vetor gradiente, que corresponde de certa maneira à dimensão dos ajustes dos pesos, varia a cada época. Essa variação ocorre devido à existência de um hiperparâmetro das redes neurais chamado Taxa de Aprendizagem. A Taxa de Aprendizagem corresponde ao fator que determina a magnitude do reajuste dos pesos a cada época de treinamento. Matematicamente, a relação entre a Taxa de Aprendizagem, o vetor gradiente da função custo e a taxa de reajuste dos pesos é representada da seguinte maneira:

$$\Delta\omega = -\epsilon\nabla G \tag{2.9}$$

Nessa equação, $\Delta\omega$ representa o vetor de variação dos pesos da rede neural, ϵ representa

o parâmetro da Taxa de Aprendizagem e ∇G representa o vetor gradiente da função custo em relação aos pesos da rede neural.

Figura 2.13 – Representação esquemática da Descida de Gradiente para diferentes taxas de aprendizagem.



Fonte: O Autor, 2023.

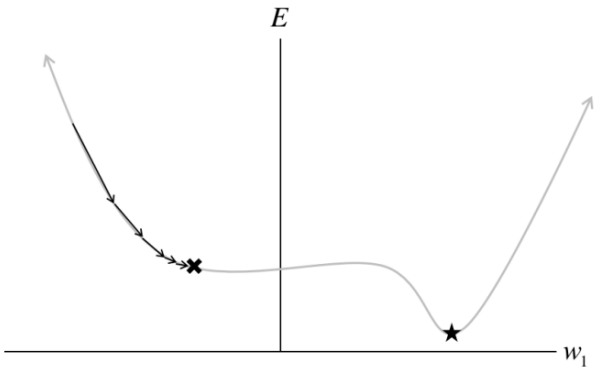
Assim, diante do exposto, as diferentes abordagens do algoritmo Descida de Gradiente recebem os seguintes nomes: *Batch Gradient Descent*, *Stochastic Gradient Descent* e *Minibatch Gradient Descent*.

Batch Gradient Descent

A abordagem do *Batch Gradient Descent*, ou BGD, ilustrada na Figura 2.14, para ajuste dos pesos, calcula a superfície da função custo em relação aos pesos da rede neural após todos os dados da base de treinamento serem submetidos ao modelo. Uma vez obtida essa superfície, o ajuste dos pesos ocorre levando em conta a posição do vetor gradiente no ponto da superfície que corresponde ao conjunto de pesos obtidos após o término de cada época de treinamento.

Essa abordagem é bastante eficiente computacionalmente, dado que a superfície da curva da função custo é calculada apenas uma vez. No entanto, esta única superfície obtida pode incorrer no problema do modelo estacionar em um mínimo local, de forma que a perfomance máxima do modelo não seja alcançada.

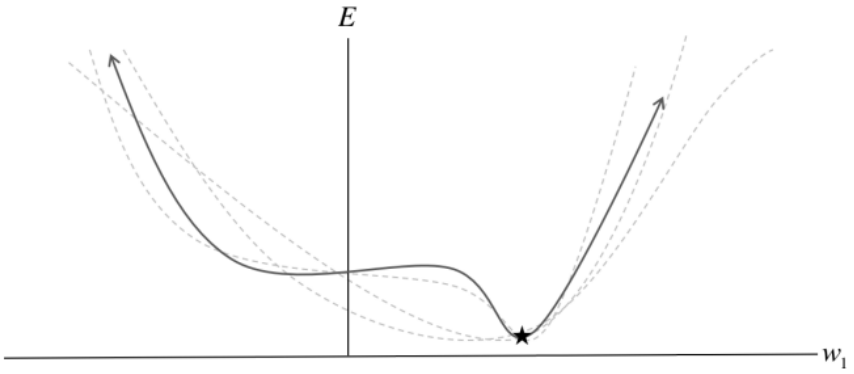
Figura 2.14 – BGD apresenta a possibilidade de alcançar uma convergência prematura, "travando"em mínimos locais. A ordenada do gráfico é a função custo e a abscissa é a o peso w_1 da RNC.



Fonte: Nikhil, 2017 [18].

Stochastic Gradient Descent

Figura 2.15 – Representação da dinamicidade das superfícies da função custo para o SGD. A ordenada do gráfico é a função custo e a abscissa é a o peso w_1 da RNC.



Fonte: Nikhil, 2017 [18].

A abordagem do *Stochastic Gradient Descent*, ou SGD, representada Figura 2.15, segue o caminho completamente oposto à abordagem do *Batch Gradient Descent*. Nessa abordagem, a superfície da função custo é calculada para cada amostra de dado presente na base de treinamento. Uma vez obtida cada superfície, o ajuste dos pesos ocorre da mesma maneira que a abordagem BGD. Assim, a cada época de treinamento, considerando uma base de dados de treinamento de tamanho N, são calculadas N superfícies da função custo, de forma que são realizados N reajustes de peso.

Contudo, dada a necessidade de obtenção da superfície da função custo para cada

amostra de dado pertencente à base de treinamento, essa abordagem apresenta um custo computacional elevado, de forma que o tempo para a execução do treinamento aumenta consideravelmente. Por outro lado, o SGD reduz bastante as chances de que o modelo treinado estacione em algum mínimo local, dada a dinamicidade das superfícieis obtidas.

Minibatch Gradient Descent

Na abordagem do *Minibatch Gradient Descent*, ou MGD, a base de treinamento é dividida em subconjuntos, chamados *minibatches*, para realização do treinamento, de forma que os dados são submetidos como entrada da rede neural *minibatch* a *minibatch*. Para cada *minibatch* colocado como entrada, é calculada uma superfície da função custo, na qual o ajuste dos pesos ocorre da mesma maneira que as abordagens do BGD e do SGD.

Essa abordagem consegue endereçar os desafios encontrados tanto com o BGD, no que concerne ao risco de convergência prematura do modelo em mínimos locais; quanto com o SGD, com seu elevado custo computacional. Assim, na prática, um outro hiperparâmento a ser definido durante a construção da estrutura da rede neural é o tamanho do *batch*, ou o tamanho dos subconjuntos disjuntos em que a base de treinamento será dividida.

2.3.4 O algoritmo de Backpropagation

Em redes neurais de múltiplas camadas, existe um algoritmo que permite o reajuste dos pesos de todas as camadas. Esse reajuste ocorre para cada batch de imagens em uma época de treinamento, a fim de encontrar o conjunto de pesos que minimiza a função custo à cada iteração sobre toda a base de treinamento. Esse algoritmo é chamado de *Backpropagation* e foi proposto pela primeira vez por David E. Rumelhart, Geoffrey E. Hinton e Ronald J. Williams [18].

O *Backpropagation* aplica derivadas parciais da função custo em relação aos pesos de cada uma das camadas de forma recursiva, para mapear a velocidade de mudança da função custo ao reajuste de cada um dos pesos da rede neural. Esse algoritmo permite que, a partir das derivadas parciais supracitadas, seja possível encontrar o melhor conjunto de pesos que resulta num valor mínimo da função custo em cada iteração. Em outras palavras, o *Backpropagation* é um algoritmo que leva à direção do vetor gradiente mais íngrime, conforme o objetivo da família de otimizadores *Gradient Descent*.

2.3.5 Underfitting e Overfitting

No processo de aprendizagem de máquina, dois conceitos são fundamentais de se levar em consideração para conclusões à respeito da qualidade dos resultados de um modelo quando aplicado à resolução de um determinado problema: *Overfitting* e *Underfitting*. Durante o treinamento, o objetivo principal desta etapa é a minimização da função custo, que informa a qualidade do modelo quando aplicado a um problema em particular. Em outras palavras, o objetivo principal é a busca de um valor ótimo para a função custo a partir de técnicas de otimização. Caso esse valor ótimo não seja alcançado após o término do treinamento, o modelo estará em um desses dois possíveis estados: *underfitting* ou *overfitting*.

Para simplificar, vale a pena introduzir dois elementos específicos para facilitar a compreensão: significado e ruído. Toda base de dados tem seus dados com atributos que são relevantes para o aprendizado do modelo (significado) e atributos que são irrelevantes para o aprendizado (ruído). O objetivo do algoritmo de um modelo é realizar o processo de aprendizagem maximizando o aprendizado de significados e minimizando o aprendizado de ruídos. Quando o modelo não aprende significado o suficiente ou aprende ruído demais, algum dos estados de *underfitting* e *overfitting* ocorreu.

O estado de *underfitting* ocorre quando o modelo não aprende significados o suficiente para alcançar seu objetivo de acordo com o problema para o qual ele é aplicado. Nesses casos, o parâmetro utilizado para avaliar a performance do modelo não alcançou seu valor ótimo. Em geral, o *underfitting* ocorre quando o modelo não é exposto a um conjunto de dados de tamanho suficiente para permitir o aprendizado de todos os elementos necessários para alcançar os objetivos desejados. Dessa maneira, o modelo deve ser apresentado a um número de dados maior, até que o valor ótimo do parâmetro de performance seja alcançado.

Por outro lado, quando o modelo aprende mais ruído do que o desejado, dizemos que esse modelo está em *overfitting*. Nesta situação, o parâmetro utilizado para avaliar o modelo também não alcançou seu valor ótimo, mas, diferentemente do *underfitting*, no *overffiting* o modelo se ajustou demais à base de treinamento. Assim, quando o mesmo modelo é empregado a uma base de teste, verifica-se que o parâmetro de performance assume um valor completamente inaquedado para os objetivos do problema. Para reduzir esse problema, existem diversas técnicas aplicadas para redes neurais, as quais serão

mencionadas mais à frente.

2.3.6 Métricas de avaliação

Uma das etapas mais importantes do processo de desenvolvimento de modelos de redes neurais é a etapa de avaliação do modelo treinado. Para essa etapa, é utilizado um conjunto de métricas que permitem avaliar a capacidade de generalização do modelo. Antes de apresentar as principais métricas, é importante definir as categorias possíveis das classificações realizadas. Assim, durante o processo de avaliação das classificações de uma rede neural, é possível categorizar cada uma das classificações em uma das quatro seguintes categorias:

- **Verdadeiros Positivos (VP):** Esse caso ocorre quando o modelo diz que um determinado dado pertence a uma classe X e de fato este dado pertence a essa classe;
- **Verdadeiros Negativos (VN):** Ocorre quando o modelo diz que determinado dado **não** pertence a uma classe X e de fato este dado não pertence a essa classe;
- **Falsos Positivos (FP):** Ocorre quando o modelo diz que um determinado dado pertende a classe X, porém ele não pertence a essa classe (ou pertence a outra classe);
- **Falsos Negativos (FN):** Ocorre quando o modelo diz que um determinado dado **não** pertence a uma classe X (ou pertence a outra classe), porém ele pertence à classe X em questão.

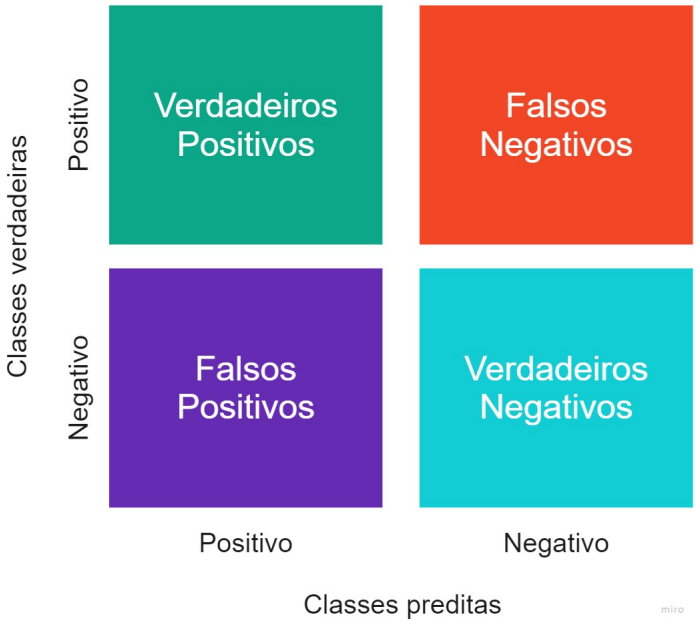
Além disso, vamos considerar que VP é o número de amostras classificadas como Verdadeiras Positivas, VN o número de amostras classificadas como Verdadeiras Negativas, FP o número de amostras classificadas como Falsas Positivas e FN o número de amostras classificadas como Falsas Negativas.

Além disso, para fins de avaliação dos resultados das etapas de treinamento e validação, foram utilizadas as métricas Acurácia e Erro. Já para avaliação dos resultados da etapa de testes, as métricas utilizadas, além de Acurácia e Erro, foram Precisão, *Recall*, Especificidade e AUC. Também foram utilizadas as Curvas ROC e a Matriz de Confusão na avaliação dos resultados desta última etapa.

Matriz de confusão

A matriz de confusão é uma ferramenta que visualmente permite avaliar a capacidade de generalização do modelo a partir da disposição numa matriz dos números de cada um dos tipos de classificações mencionados nesta subseção. A matriz na Figura 2.16 representa uma matriz de confusão.

Figura 2.16 – Matriz de confusão. A matriz dispõe de quatro quadrantes, cada um representando os possíveis resultados das classificações de um modelo.



Fonte: O autor, 2023.

Como mostra a Figura 2.16, no eixo vertical da matriz se encontram os valores quantitativos correspondentes às categorias verdadeiras dos dados colocados como entrada de um modelo. Por outro lado, no eixo horizontal, tem-se a colocação dos valores correspondentes às categorias previstas pelo modelo em relação aos dados utilizados como entrada. A avaliação do modelo se dá através da comparação dos resultados de classificação previstos pelo modelo para cada um dos dados colocados como entrada em relação às reais classes às quais esses dados pertencem. Essa comparação é feita calculando a quantidade de classificações realizadas que pertencem a cada uma das categorias possíveis apresentadas anteriormente e dispondo os resultados na matriz de confusão.

Assim, para que um modelo seja considerado como quem possui uma boa capacidade de generalização, é preciso que a diagonal principal, correspondente às classificações efeti-

vamente corretas, tenha os maiores valores possíveis, assim como os valores pertencentes aos outros dois quadrantes sejam minimizados ao máximo. Por fim, a visualização desses resultados pode ser feita tanto de maneira absoluta, quanto de maneira relativa, em termos percentuais. Quando em termos percentuais, contudo, cabe destacar que a soma dos valores percentuais em cada uma das linhas da matriz deve resultar em 100%, uma vez que cada linha representa de forma quantitativa uma categoria possível em sua totalidade (seja em classificações positivas, seja em classificações negativas).

Acurácia

Essa métrica serve para medir a taxa de classificações corretas do modelo comparativamente ao total de classificações. Matematicamente, a acurácia é calculada conforme a equação 2.10.

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.10}$$

É importante destacar que a acurácia também pode ser uma das métricas definidas como função custo durante o treinamento de modelos. Nestes casos, diferentemente do erro como função custo, o objetivo passa a ser maximizar a acurácia.

Precisão

A precisão mede a taxa de classificações Verdadeiras Positivas em relação ao total de classificações positivas. Matematicamente, a precisão é obtida através da equação 2.11.

$$P = \frac{VP}{VP + FP} \tag{2.11}$$

Recall

O *Recall*, também chamado de Sensibilidade, mede a taxa de classificações Verdadeiras Positivas em relação ao total de amostras positivas. A equação 2.12 é utilizada para o cálculo do valor do *Recall*.

$$R = \frac{VP}{VP + FN} \tag{2.12}$$

Curva ROC e AUC

Duas ferramentas de bastante importância na avaliação da performance de modelos de classificação são a curva ROC e a AUC. A curva ROC, cuja sigla vem da expressão *Receiver Operating Characteristic*, é uma ferramenta gráfica para realizar a avaliação de performance de classificação através da comparação entre duas medidas em particular: o Recall e a Taxa de Falsos Positivos. Matematicamente, a Taxa de Falsos Positivos é dada pela equação 2.13.

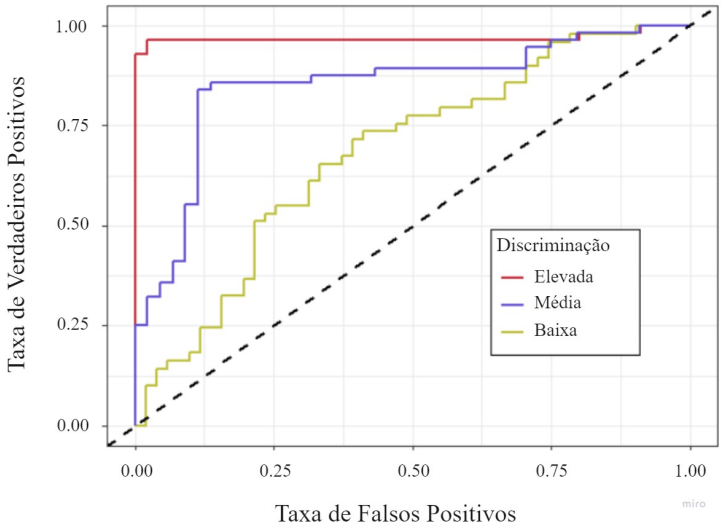
$$F = 1 - S = \frac{FP}{FP + VN} \tag{2.13}$$

Assim, a curva ROC é obtida a partir da razão entre o *Recall* e a Taxa de Falsos Positivos para diferentes valores de limiar de classificação. Esses valores limiares de classificação nada mais são do que um parâmetro percentual que indica se determinada classificação pertence ou não a uma classe. Ao realizar uma classificação, o modelo apresenta a probabilidade de a amostra classificada pertencer a uma determinada classe. Se essa probabilidade for maior que o valor de limiar, a classificação é positiva e, caso contrário, negativa.

Modelos que apresentam bons resultados são aqueles para os quais a razão entre o *Recall* e a Taxa de Falsos Positivos é a maior possível e que não varia (ou varia pouco) com a mudança do valor de limiar de classificação. A Figura 2.17 representa um exemplo de curva ROC para diferentes valores de limiar. Cabe destacar que as curvas da Figura 2.17 representam a relação entre o Recall e a Taxa de Falsos Positivos para cada uma das classes disponíveis. A possibilidade de obtenção de uma curva ROC para cada classe apresenta importância fundamental, uma vez que é possível assim avaliar a performance de um modelo nas classificações classe a classe e identificar potencias dificuldades do modelo em classificar corretamente dados de determinada classe.

Já a AUC, ou *Area Under Curve*, basicamente representa a área sob a curva ROC obtida após o treinamento. A AUC se trata de uma métrica cujo objetivo é alcançar o maior valor possível, pois nesses casos a relação entre o *Recall* e a Taxa de Falsos Positivos também é máxima. Assim como para a Curva ROC, a AUC é obtida para cada uma das classes envolvidas no problema de classificação em questão, uma vez que esta última é resultado direto das curvas ROC obtidas.

Figura 2.17 – Curva ROC para diferentes classificações. No eixo das abcissas estão os valores da Taxa de Falso Positivos e no eixo das ordenadas os valores do Recall.



Fonte: Adaptado de ZG Silva Neto, 2020 [25].

DOR

A métrica DOR, do inglês *Diagnostics Odds Rate*, também é uma métrica bastante utilizada na avaliação de modelos de classificação. Essa métrica tem sua formulação matemática dada pela equação 2.14.

$$DOR = \frac{\frac{TP}{FN}}{\frac{FP}{TN}} \tag{2.14}$$

Basicamente, essa métrica pode ser entendida como a medição das chances de uma classificação positiva para uma amostra efetivamente positiva em comparação com as chances de classificação positiva para uma amostra que não é de fato positiva [26]. Essa métrica é bastante útil para classificações binárias ou para comparação de classificações de uma classe em relação a todas as outras, como utilizado nas seções 4.3 e 4.4.

2.3.7 Conceitos adicionais importantes

Alguns outros conceitos importantes utilizados neste trabalho:

- **Transfer Learning:** *Transfer Learning* é uma técnica utilizada no treinamento de RNCs, a qual consiste em utilizar um modelo pré-treinado para realizar a tarefa de extração de características das imagens. Um modelo pré-treinado se trata de uma

RNC já treinada a partir de uma base de dados com um grande número de classes. Em geral, se usa apenas a Base de Extração de Características das redes neurais pré-treinadas, incluindo-se no seu topo o Head Classificador específico para a tarefa a qual a RNC completa vai ser aplicada. Alguns exemplos de modelos pré-treinados são: InceptionV3, AlexNet, ResNet, EfficientNet, VGG16 e VGG19;

- **Fine Tuning:** Durante o processo de treinamento usando Transfer Learning, em geral se adota o caminho em que os pesos da Base de Extração de Características são mantidos, ou "congelados". O processo de Fine Tuning consiste em, após o processo de treinamento, já com resultados satisfatórios, "descongelar" por completo ou parcialmente os pesos das camadas da RNC e retreinar a rede a partir de um conjunto de imagens totalmente novas usando um baixo valor de Taxa de Aprendizagem. Essa técnica possibilita aprimorar ainda mais a capacidade de generalização do modelo desenvolvido;

- **Limiar de classificação:** Corresponde a um valor de probabilidade considerado para classificações de imagens em classes. Resultados de classificação com probabilidade maior que o limiar de classificação são classificados como positivos em relação ao pertencimento a determinada classe. Caso contrário, como negativos;

- **Função Erro Entropia Cruzada:** Função erro utilizada para classificações multiclasses, a qual mede a distância da distribuição de probabilidade entre o valor previsto e o valor real. Quanto menor for essa diferença, melhor a classificação realizada.

Capítulo 3

Metodologia

Este capítulo fala da metodologia empregada para o desenvolvimento deste trabalho. Na seção 3.1, é apresentada a sequência de etapas percorridas para alcance dos modelos com capacidade de generalização satisfatória. As seções 3.2 a 3.6 descrevem todas as atividades realizadas em cada uma das etapas. Na seção 3.7 são apresentadas as ferramentas utilizadas neste trabalho.

3.1 Etapas

Este trabalho foi construído seguindo a sequência de etapas abaixo:

1. **Obtenção dos dados:** Esta etapa consiste na obtenção dos dados que serão utilizados para o treinamento das redes neurais utilizadas;

2. **Pré-processamento dos dados:** Nesta etapa, foi aplicada uma série de transformações úteis e necessárias aos dados utilizados para os ciclos de treinamento, validação e teste das redes neurais implementadas;

3. **Definição e construção dos modelos:** Nesta etapa, foram selecionados os modelos pré-treinados VGG16 e Xception para comparação com o MCTC, construído pelo autor;

4. **Treinamento dos modelos:** Nesta etapa, os modelos mencionados foram treinados a partir dos dados processados;

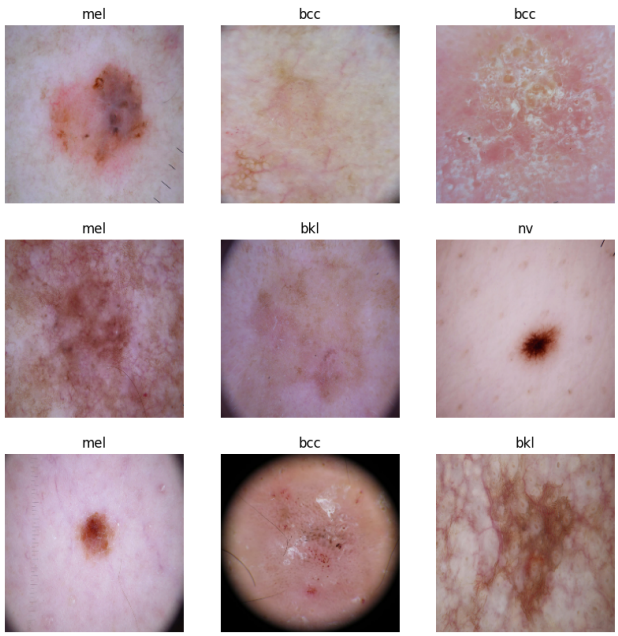
5. **Ajustes e aprimoramentos:** A partir dos resultados do treinamento, foram realizados ajustes nos parâmetros dos modelos a fim de aprimorar as suas respectivas capacidades de generalização.

3.2 Obtenção dos dados

Para fins de treinamento dos modelos VGG16, Xception e MCTC, foram escolhidas imagens da *International Skin Imaging Collaboration* (ISIC). A ISIC é originada a partir de uma parceria entre empresas e universidades para tornar mais simples e amplamente conhecidos os principais tipos de lesões de pele e assim tornar o processo de diagnóstico mais fácil e mais rápido [27]. O ISIC tem desempenhado um papel fundamental no desenvolvimento de técnicas e algoritmos de Inteligência Artificial para o diagnóstico de câncer de pele, inclusive em seus estágios iniciais.

A base de dados do ISIC conta com imagens de diferentes tipos de classes de lesão, mas possui um foco especial em lesões do tipo Melanoma. As imagens obtidas estão em arquivos do tipo JPEG com dimensão de 600x450. No total, foram utilizadas 4396 imagens, sendo 75% desse total dedicado a treinamento, 20% para validação e 5% para teste. A Figura 3.1 exemplifica essas lesões.

Figura 3.1 – Amostra da base de imagens utilizada no treinamento, validação e testes dos modelos.



Fonte: O Autor, 2023.

Além disso, cabe destacar que para cada uma das etapas de treinamento, validação e testes, cada uma das classes foram apresentadas ao modelo tendo o mesmo número de imagens por classe. Em outras palavras, todo trabalho foi realizado com uma base de dados balanceada, sendo um quarto do total de imagens para cada etapa a quantidade de imagens de cada uma das classes.

Cada uma das siglas na Figura 3.1 representa um dos tipos de lesão. A sigla *bcc* representa Carcinoma Basocelular, *bkl* representa Queratose Seborreica, *mel* para Melanoma e *nv* para Nevo Benigno.

3.3 Pré-processamento dos dados

Uma das etapas mais importantes em aplicações de *Machine Learning* é o pré-processamento da base de dados utilizada para treinamento, validação e teste dos modelos. Esse pré-processamento envolve uma série de operações como transformações nos dados, conversões de tipos de arquivos, codificações, etc. Quando os dados são imagens, estas exigem uma série de operações específicas para que estejam aptas a serem usadas como dados de entrada do modelo, a fim de que este possa extrair das imagens as características mais relevantes para a classificação.

Neste trabalho foram realizadas as seguintes operações sobre a base de dados:

1. Conversão para RGB;
2. Ajuste de dimensões;
3. Data Augmentation;
4. Normalização;
5. Codificação.

As imagens foram submetidas como entrada para os modelos no padrão RGB, já que os modelos utilizados neste trabalho exigem que as imagens sejam convertidas para matrizes do tipo RGB. Além disso, conforme especificação para implementação dos modelos, quando utilizadas para treinamento dos modelos Xception [28] e MCTC [29], as imagens foram redimensionadas para um tamanho de 299x299, sendo aplicadas como entrada

desses modelos no formato 299x299x3. Na Figura 3.1, as imagens se encontram redimensionadas no formato 299x299x3, pois foram utilizadas para treinar os modelos Xception e MCTC. Por outro lado, conforme as especificações para implementação do modelo VGG16 [30], as imagens foram redimensionadas para um tamanho de 224x224, sendo aplicadas como entrada deste modelo no formato 224x224x3.

O redimensionamento é uma das ações mais necessárias do pré-processamento dos dados. O motivo dessa afirmação é que essa etapa serve para evitar que a capacidade de processamento do hardware computacional utilizado não seja ultrapassada, de modo que o treinamento do modelo seja executado por completo. Além disso, a base de imagens ISIC é formada por imagens obtidas de fontes diversas, como universidades, laboratórios e empresas. Assim, o redimensionamento das imagens aplica um padrão aos dados que facilita o processo de aprendizado dos modelos aplicados para a tarefa de classificação.

Em sequência ao redimensionamento, uma técnica bastante empregada para ampliar o número de exemplos de dados aplicados como entrada dos modelos é o *Data Augmentation*. Essa técnica consiste em implementar uma série de transformações visuais às imagens que resultam numa base de dados ainda maior e com dados aprimorados. As transformações possíveis de serem aplicadas são as mais diversas, o que inclui rotações, translações, alterações de brilho, ampliações, etc.

Além disso, algumas das operações de *Data Augmentation* têm o objetivo de evidenciar algumas características das lesões, a afim de tornar o processo de extração de características das imagens mais eficiente. Cabe destacar que as operações de *Data Augmentation* desempenham também o papel de tornar as classificações das lesões invariáveis à posição da lesão na imagem. Isso ocorre pois as diversas operações tais como rotações, translações, entre outras, colocam uma mesma lesão em diferentes posições em diferentes imagens, o que gera essa característica de invariância à posição nas classificações. Neste trabalho, as transformações de *Data Augmentation* aplicadas estão indicadas na Tabela 3.1.

A penúltima transformação foi a normalização dos pixels das imagens. Como as imagens estão em escala RGB, isto é, são matrizes cujos valores variam de 0 a 255, existe a necessidade de colocar esses valores na mesma escala de forma que as predições dos modelos não sejam prejudicadas. A aplicação de dados não normalizados no treinamento dos modelos pode resultar em prejuízos na classificação, dado que um longo intervalo de valores de entrada pode influenciar na definição dos pesos durante o ciclo de treinamento,

Tabela 3.1 – Operações de *Data Augmentation* aplicadas à base de imagens utilizada para treinamento dos modelos.

Operações de *Data Augmentation*	
Espelhamento horizontal	Sim (180^o)
Alteração na altura	0.1
Alteração na largura	0.1
Rotação	30^o
Ampliação	0.125 vezes

Fonte: O Autor, 2023.

de forma que os modelos não realizem as melhores classificações.

Por fim, para fins de classificação e obtenção dos resultados, foi aplicada uma codificação a cada uma das classes de câncer a serem classificadas. Essa codificação permite que o modelo possa realizar as classificações matematicamente, o que não seria possível caso as classes fossem distinguidas de forma nomeada. Os códigos aplicados a cada uma das classes estão na Tabela 3.2:

Tabela 3.2 – Codificação de cada uma das classes de Câncer da base de dados deste trabalho. para fins de treinamento e classificação

Codificação das classes		
Tipo de câncer	Sigla	Código
Carcinoma Basocelular	bcc	0
Queratose Seborreica	bkl	1
Melanoma	mel	2
Nevo Benigno	nv	3

Fonte: O Autor, 2023.

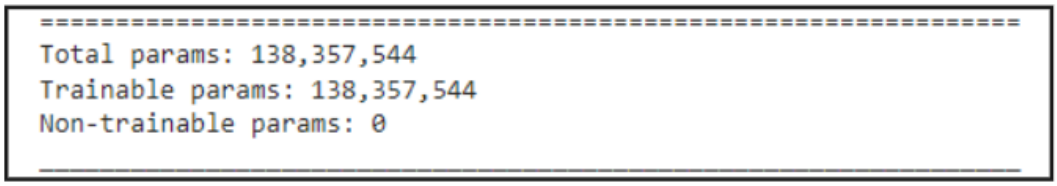
A codificação foi realizada fazendo uso da classe *ImageDataGenerator* do TensorFlow, a qual cria *batches* de imagens com *Data Augmentation* em tempo real. Essa classe possui o método *flow_from_directory*, o qual cria uma classe chamada *DirectoryIterator*, que atribui uma codificação a cada uma das categorias dos dados. Essa codificação é obtida através do atributo *class_indices*, que retorna um dicionário na qual as chaves são os nomes das categorias e os valores o código de cada uma das categorias.

3.4 Definição e construção dos modelos

Como mencionado na seção 1.2, foram selecionados três modelos para avaliação: o VGG16, o Xception e o MCTC. Esses modelos foram treinados e testados em lotes que progrediam de 400 imagens por simulação. Em outras palavras, as simulações foram realizadas iniciando com uma base de 400 imagens e progredindo o tamanho dessa base em passos de 400 imagens.

O modelo VGG16 é um modelo construído em 2014 por Simonyan e Zisserman [31] para avaliar a performance de Redes Neurais Convolucionais com filtros de convolução, ou *kernels*, de tamanho 3x3. Esse modelo aplica esses filtros em sequência, o que reduz bastante o custo computacional comparado com modelos com filtros de convolução de dimensão 5x5 e 7x7. O VGG16 alcançou as primeira e segunda posições, respectivamente, em desafios de localização e classificação no *Imagenet Challenge 2014* [31]. Além disso, o modelo VGG16 implementado apresenta um total de 16 camadas, sendo 13 dessas camadas como camadas de convolução e as outras 3 como camadas totalmente conectadas. A Figura 3.2 apresenta o número de parâmetros do modelo VGG16 neste trabalho.

Figura 3.2 – Número total de parâmetros treináveis e não-treináveis do modelo VGG16 após a realização do treinamento.



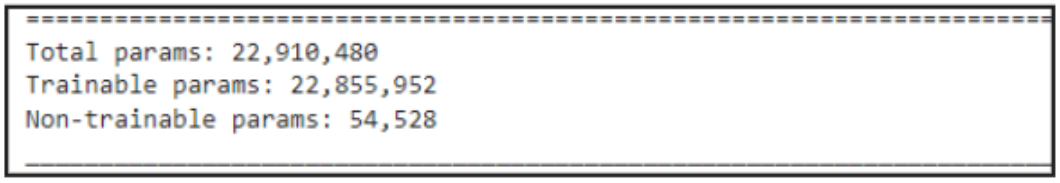
Fonte: O Autor, 2023.

O modelo Xception é um modelo desenvolvido por François Chollet [32] em 2017, publicado no artigo *Xception: Deep Learning with Depthwise Separable Convolutions*, o qual lança mão das operações de convolução separáveis em profundidade para realizar a extração de características das imagens de entrada. O modelo Xception conseguiu alcançar desempenho ligeiramente superior ao modelo Inception V3 quando treinado na base *ImageNet* e desempenho bastante superior quando treinado em uma base de imagens com mais de 350 milhões de amostras e 17.000 classes [32].

A arquitetura do modelo é dividida em três etapas principais. A primeira etapa da

arquitetura, o *Entry flow*, corresponde à etapa na qual os dados são introduzidos ao modelo, sendo essa etapa percorrida apenas uma vez. A etapa *Middle flow* corresponde à etapa intermediária, na qual os dados passam por essa etapa 8 vezes até chegar à última etapa da arquitetura, o *Exit flow* [32], também percorrida apenas uma vez. Essa arquitetura apresenta uma estrutura para a Base de Extração de Características com 36 camadas de convolução, seguida de apenas uma camada de regressão logística para classificação, uma vez que os autores do artigo se restringiram a avaliar o modelo Xception apenas na tarefa de classificação. A Figura 3.3 apresenta o número total de parâmetros do modelo.

Figura 3.3 – Número total de parâmetros treináveis e não-treináveis do modelo Xception após a realização do treinamento.



Fonte: O Autor, 2023.

Por fim, o último modelo presente neste trabalho foi o MCTC. O MCTC é um modelo que faz uso da técnica de *Transfer Learning*, em que a arquitetura utilizada para desempenhar o papel de Base de Extração de Características foi o Inception V3 [33]. O Inception V3 foi uma arquitetura introduzida pelos pesquisadores Szegedy, Vanhoucke, Ioffe e Shlens, a qual faz uso de módulos *Inceptions* para redução de custos computacionais das operações de convolução. Esses módulos consistem de uma sequência de camadas as quais contém filtros de dimensões 1x1, 3x3 e 5x5. O modelo MCTC apresenta em sua estrutura 93 camadas de convolução, advindas da Base de Extração de Características da arquitetura do InceptionV3. Ao topo da Base de Extração de Características, foi incluído um Head Classificador, com 4 camadas densas, para classificação das imagens, cuja estrutura está abaixo:

```
1  tf.keras.layers.Flatten(),
2  tf.keras.layers.BatchNormalization(),
3  tf.keras.layers.Dense(512, activation='relu'),
4  tf.keras.layers.Dropout(rate=0.2),
```

```
5   tf.keras.layers.BatchNormalization(),
6   tf.keras.layers.Dense(512, activation='relu'),
7   tf.keras.layers.Dropout(rate=0.3),
8   tf.keras.layers.BatchNormalization(),
9   tf.keras.layers.Dense(128, activation='relu'),
10  tf.keras.layers.Dropout(rate=0.2),
11  tf.keras.layers.BatchNormalization(),
12  tf.keras.layers.Dense(num_classes, activation='softmax')
```

A camada *Flatten* desempenha o papel de colocar os dados de entrada num formato adequado à entrada da camada totalmente conectada. A camada de *BatchNormalization* é a responsável por normalizar os dados vindos em lote, enquanto que as camadas *Dropout* servem para reduzir o *overfitting* aplicando a técnica de regularização *dropout* a certos percentuais da camada totalmente conectada. Por fim, uma sequência de três camadas densas, com respectivamente 512, 512 e 128 neurônios, com funções de ativação do tipo ReLU, junto a uma camada final com 4 neurônios (valor do parâmetro *num_classes*) e função de ativação *softmax* concluem a estrutura do Head Classificador. A figura 3.4 apresenta o número total de parâmetros do modelo MCTC.

Figura 3.4 – Número total de parâmetros treináveis e não-treináveis do modelo MCTC após a realização do treinamento.

```
=================================================================
Total params: 23,193,508
Trainable params: 23,152,676
Non-trainable params: 40,832
_________________________________________________________________
```

Fonte: O Autor, 2023.

3.5 Treinamento dos modelos

Para alcance da convergência do modelo durante o treinamento, algumas técnicas foram empregadas com base em ferramentas providas pelo Keras. A primeira técnica foi o emprego do método *EarlyStopping*, que consiste num método que encerra o treinamento após uma determinada métrica de acompanhamento não alterar seu valor por um certo

número de épocas de treinamento. A segunda técnica fez uso da ferramenta *ReduceLROnPlateau*, a qual consiste em um método que reduz a Taxa de Aprendizagem por um fator pré-determinado caso o valor de uma determinada métrica de acompanhamento não seja alterado após um número de épocas selecionado.

A terceira técnica empregada lançou mão do método *ModelCheckpoint*, o qual, durante o processo de treinamento, salva os pesos do modelo que alcançou melhor desempenho em relação a uma determinada métrica de acompanhamento selecionada. No caso deste trabalho, foi definido que o modelo que alcançasse o menor valor de função erro sobre a base de validação seria salvo. Assim, o modelo de melhor desempenho seria armazenado para testes posteriores.

Por fim foram escolhidos o algoritmo de otimização SGD e a função custo sendo a função erro *Sparse Categorical Crossentropy* para treinamento. Além disso, as métricas acompanhadas durante o processo de treinamento foram a acurácia e o erro.

3.6 Validação e testes

Após a conclusão do treinamento de cada um dos modelos, a avaliação de performance foi realizada em duas fases. A primeira fase consiste na análise da curva de aprendizagem do modelo em relação às bases de treinamento e validação. Essa análise foi feita para verificação da performance do modelo no treinamento, observando a ocorrência de *underfitting* ou *overfitting*. O objetivo principal dessa fase é de avaliar se o modelo conseguiu atingir valores satisfatórios de erro e acurácia, assim como verificar se nenhum dos dois casos de *underfitting* e *overfitting* ocorreu.

A segunda fase consiste no teste dos modelos treinados através da base de teste. Essa fase tem o objetivo de avaliar a capacidade de generalização dos modelos obtidos, a fim de inferir sua potencial aplicação em contextos reais de uso. Para avaliação dos resultados dos testes dos modelos, foram utilizadas uma série de métricas: Erro, Acurácia, Precisão, *Recall*, *F1-Score*, Matriz de Confusão, Curva ROC, AUC e DOR.

Assim, a partir dos resultados das avaliações, os parâmetros de treinamento eram ajustados a fim de se alcançar resultados satisfatórios de treinamento, validação e teste dos modelos. Então, os resultados foram então compilados e serão apresentados no Capítulo 4 deste trabalho.

3.7 Ferramentas e ambientes utilizados

Neste trabalho, foram usadas as seguintes ferramentas:

- **TensorFlow:** O TensorFlow se trata de um conjunto de pacotes e módulos Python, de código aberto, utilizado para construir, treinar e testar modelos de *Deep Learning*, lançado pelo Google em 2015. De maneira pragmática, o TensorFlow é uma biblioteca Python que permite a construção de grafos de fluxos de dados para computação[18];

- **Google Colaboratory:** O Google Colaboratory se trata de um ambiente do Google que permite a escrita de código Python, em geral para aplicações de *Machine Learning*, Ciência de Dados e IA. Essa ferramenta conta com acesso a GPUs e TPUs que facilitam o treinamento de redes neurais, tornando esse processo muito mais veloz graças à capacidade de computação desses tipos de hardwares. Através do Google Colaboratory foi utilizada a GPU T4 da NVIDIA para treinamento, validação e teste dos modelos;

- **Keras:** Keras é uma API escrita em Python, utilizada para aplicações de *Machine Learning*, com foco em *Deep Learning*, executada através do TensorFlow.[34];

- **Python:** Python é uma linguagem de programação orientada a objetos com uma ampla gama de aplicações em áreas como desenvolvimento web, sistemas embarcados e Inteligência Artificial. Sua ampla comunidade e a existência de uma série de bibliotecas e frameworks que facilitam a construção de modelos de *Machine Learning* foram as razões para escolha dessa linguagem.

Capítulo 4

Resultados

Na seção 4.1 é feita uma síntese a respeito das simulações, apresentando uma visão dos hiperparâmetros selecionados para treinamento e dos parâmetros dos modelos treinados. Na seção 4.2 é exposto os resultados de performance dos modelos treinados neste trabalho, apresentando os números obtidos para acurácia, erro e número de épocas para conclusão do treinamento. Na seção 4.3 é feita uma análise individual das simulações de melhor desempenho de cada modelo. Por fim, na seção 4.4, é feita uma análise comparativa entre as simulações de melhor desempenho de todos os modelos.

Para seleção da simulação de melhor desempenho para cada um dos modelos, foi levado em consideração a meta de acurácia estabelecida na seção 1.2. Dito isto, será selecionada como simulação de melhor desempenho aquela que tiver maior valor de acurácia alcançado entre todas as simulações realizadas. Cabe ressaltar que, exceto as curvas de aprendizagem e a tabela 4.2, todos os gráficos e tabelas apresentados neste capítulo foram obtidos aplicando cada um dos modelos treinados à base de teste, a afim de avaliar sua capacidade de generalização.

4.1 Simulações

Para as simulações realizadas, o tamanho do *batch* para treinamento foi definido como 32. Além disso, o valor da Taxa de Aprendizagem inicial foi estabelecido como 0,007. Esse valor era reduzido por um fator de 0,7 caso o erro sobre a base de validação não fosse reduzido após 5 épocas de treinamento.

A cada redução do valor do erro, o modelo que corresponde ao mínimo erro obtido

era salvo, como mencionado na seção 3.5, através do método *ModelCheckpoint*. Caso não houvesse nenhuma redução do valor do erro sobre a base de validação após 16 épocas de treinamento, a simulação era finalizada através do método *EarlyStopping* do Keras. Cabe destacar que o número máximo de épocas de treinamento definido foi de 600, apesar de as simulações encerrarem após uma média de 60 épocas nas condições aqui estabelecidas para alcance da melhor convergência possível dos modelos.

4.2 Resultados obtidos

Tabela 4.1 – Resultados sobre a base de teste das simulações para diferentes tamanhos da base de imagens utilizada.

Nº total de imagens	Nº de imagens de teste	MCTC		VGG16		Xception	
-	-	Acurácia(%)	Erro	Acurácia(%)	Erro	Acurácia(%)	Erro
400	20	30	1,47	50	1,12	75	0,74
800	40	25	1,48	70	0,73	75	0,56
1200	60	57	0,87	65	0,69	83	0,54
1600	80	75	0,68	73	0,71	75	0,71
2000	100	81	0,46	73	0,58	79	0,61
2400	120	68	0,76	77	0,57	80	0,57
2800	140	81	0,78	76	0,56	81	0,50
3200	160	78	0,63	76	0,59	82	0,54
3600	180	77	0,64	81	0,46	81	0,48
4000	200	83	0,56	81	0,47	84	0,59
4396	220	85	0,47	84	0,48	83	0,48

A Tabela 4.1 apresenta os resultados de erro e acurácia obtidos a partir das simulações utilizando a base de teste para cada um dos modelos estudados, considerando as progressões do tamanho da base conforme descrito na seção 3.4. É importante ressaltar que cada simulação foi executada uma vez para obtenção dos resultados descritos na Tabela 4.1. Ao todo, foram 11 simulações executadas por modelo.

A coluna de cabeçalho "Nº de imagens"representa o tamanho integral da base de imagens utilizada em cada uma das simulações, enquanto a coluna "Nº de imagens de teste"representa o número de imagens da base de teste. Cabe destacar que na 4.1, os

valores de acurácia foram obtidos comparando as classificações corretas em relação a todas as outras classificações. Além disso, o valor do erro foi obtido a partir do cálculo da entropia cruzada das classificações.

Na Tabela 4.2, tem-se o número de épocas necessárias para se alcançar os requisitos de finalização do treinamento, definidos na seção 3.6. Na coluna de cabeçalho "Nº de imagens", tem-se o tamanho integral da base de imagens utilizada em cada uma das simulações neste trabalho.

Tabela 4.2 – Número de épocas de treinamento até a convergência para diferentes tamanhos de base de imagens utilizada.

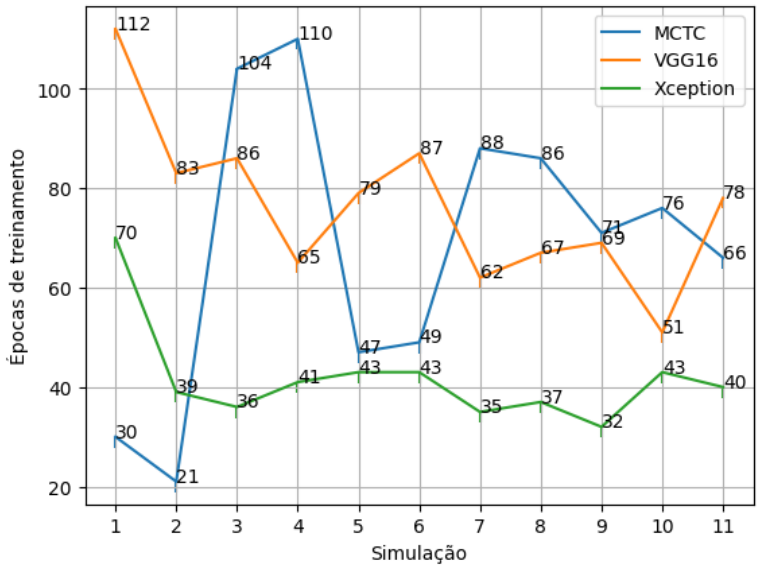
Épocas de treinamento por simulação			
Nº de imagens	MCTC	VGG16	Xception
400	30	112	70
800	21	83	39
1200	104	86	36
1600	110	65	41
2000	47	79	43
2400	49	87	43
2800	88	62	35
3200	86	67	37
3600	71	69	32
4000	76	51	43
4396	66	78	40

Esta tabela tem o intuito de apresentar a velocidade de convergência durante a etapa de treinamento de cada um dos modelos para cada uma das simulações. A análise da velocidade de convergência do treinamento assume um papel bastante relevante no sentido de economia de recursos computacionais e financeiros, principalmente quando se faz uso de infraestrutura computacional em nuvem fornecida por terceiros para treinamento.

Uma outra visulização sobre a velocidade de convergência de cada um dos modelos é apresentada no gráfico da Figura 4.1. Esse gráfico apresenta o número de épocas de treinamento para convergência (eixo das ordenadas) para cada simulação realizada (eixo das abcissas). Como se pode verificar pelas curvas, o modelo Xception (curva verde) apresenta a maior velocidade de convergência entre todos os modelos. Por outro lado, os

modelos MCTC (curva azul) e VGG16 (curva laranja) apresentam velocidades de convergência similares, levando, em média, o mesmo número de épocas de treinamento para alcance da convergência.

Figura 4.1 – Curvas das velocidades de convergência de cada um dos modelos avaliada através do número de épocas de treinamento.



Fonte: O Autor, 2023.

É importante destacar que, para fins de treinamento e validação, os percentuais da quantidade de imagens utilizadas estão definidos na seção 3.3. Isto é, o número de épocas que consta na Tabela 4.2, também dispostos nas curvas do gráfico da Figura 4.1, para atingir os requisitos necessários para finalização do treinamento correspondem, respectivamente, ao uso de 75% do conjunto integral de imagens em cada uma das simulações para fins de treinamento e 20% para fins de validação.

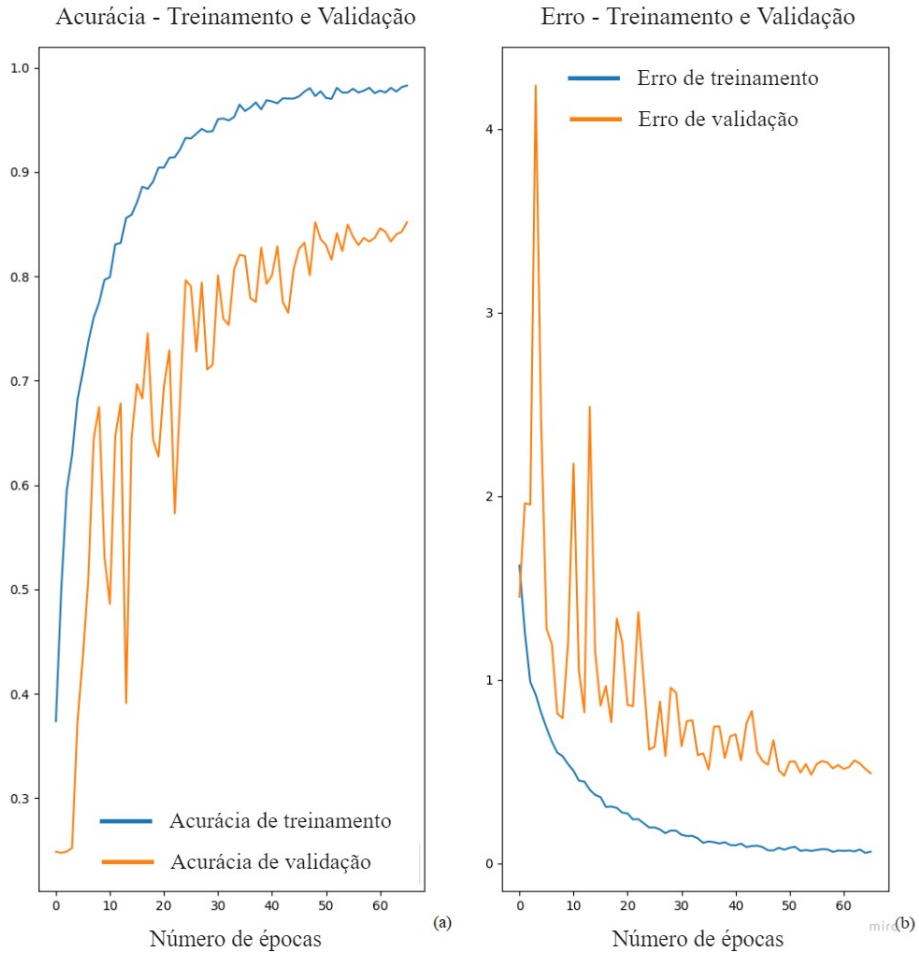
4.3 Análise das simulações de melhor desempenho

Nesta subseção, serão analisadas as simulações de melhor desempenho de cada um dos modelos. Para seleção dessas simulações, será escolhida aquela que apresentar maior valor de acurácia, sendo o erro a métrica de desempate caso haja mais de uma simulação com o mesmo valor de acurácia. Nestes casos, será selecionada a simulação com menor erro.

Para o MCTC e para o VGG16, cada um teve a última simulação como a simulação de melhor desempenho, a qual foi executada com uma base de 4.396 imagens. Por outro lado, para o modelo Xception, a simulação selecionada como a de melhor desempenho foi a penúltima, a qual foi executada com uma base de 4.000 imagens.

4.3.1 Modelo MCTC

Figura 4.2 – Curvas de aprendizagem do MCTC: Acurácia (a) e Erro (b) em relação às bases de treinamento e validação.



Fonte: O Autor, 2023.

As curvas de aprendizagem do modelo MCTC estão representadas nos gráficos da figura 4.2. O gráfico na Figura 4.2 (a) representa a evolução da acurácia do modelo ao longo do processo de treinamento. Este gráfico aponta que o modelo conseguiu alcançar uma acurácia superior a 80% em relação às imagens utilizadas para validação, no qual a

curva apresenta um comportamento cada vez mais suave a cada época à medida que o treinamento chega próximo de sua conclusão.

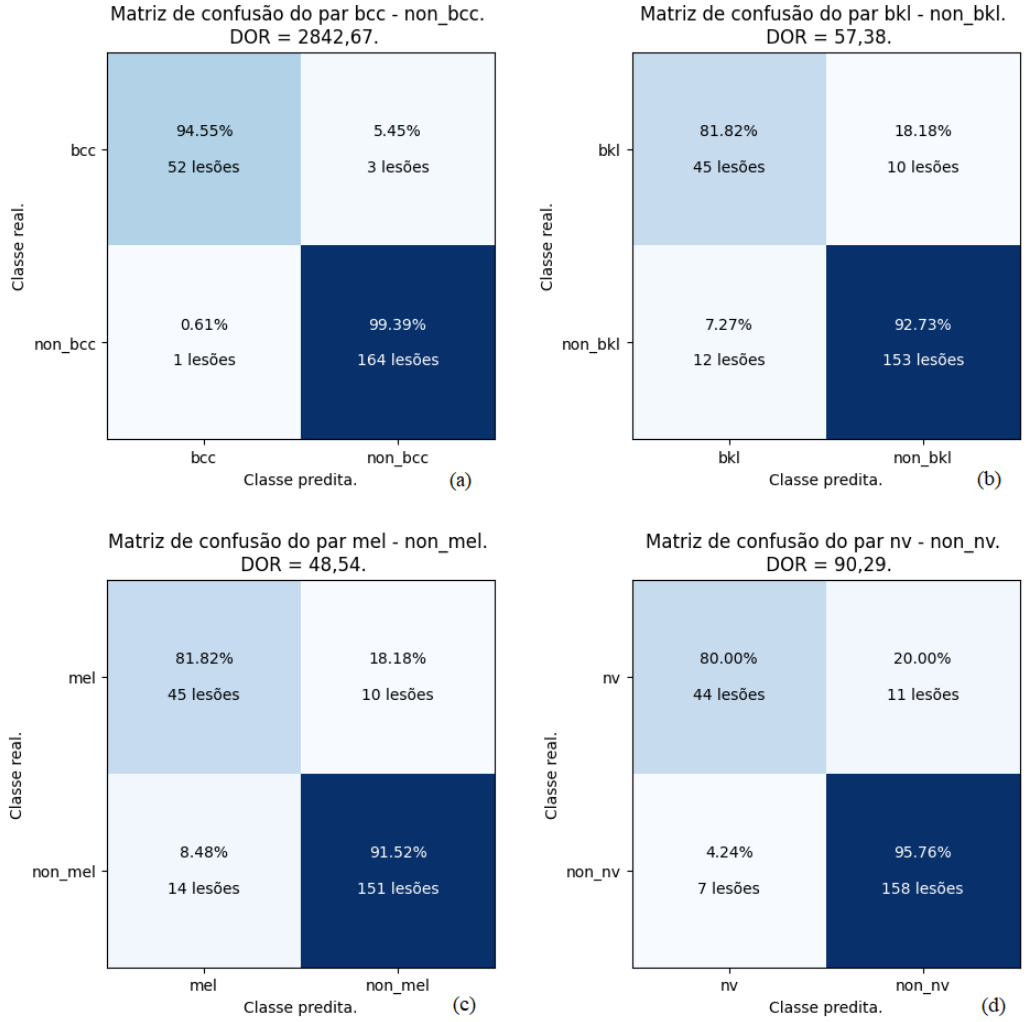
Além disso, o gráfico na Figura 4.2 (b) apresenta a evolução do erro também durante o processo de treinamento. Como aponta este gráfico, o comportamento da curva do erro para a base de validação assume uma característica cada vez mais suave a cada época à medida que o treinamento chega próximo do fim, chegando em valores próximos de 0,5. Além disso, como se pode notar, as curvas de erro em relação às bases de treinamento e validação se estabilizam e não divergem uma da outra à medida que o treinamento chega próximo do fim, o que aponta que as técnicas descritas na seção 3.5 e 3.6 e utilizadas na metodologia do trabalho foram úteis para evitar a ocorrência de *overfitting* durante o processo de treinamento.

A Figura 4.3 apresenta uma visão do desempenho do MCTC, através de matrizes de confusão, na sua capacidade de identificar e diferenciar uma determinada classe de lesão em relação a todas as outras. As matrizes de confusão (a), (b), (c) e (d) da Figura 4.3 são, respectivamente, uma avaliação da capacidade de diferenciação do modelo das doenças Carcinoma Basocelular, Queratose Seborreica, Melanoma e Nevo Benigno.

Como mostra a Figura 4.3 (a), o Carcinoma Basocelular foi a lesão em que o modelo apresentou melhor capacidade de identificação, alcançando uma taxa de acerto de 94,55%. Além disso, o modelo também consegue definir bem quando uma lesão não é do tipo Carcinoma, uma vez que este apresentou 99,39% de taxa de acerto em classificar corretamente lesões que não eram desta classe. A performance do modelo em identificar e classificar corretamente o Carcinoma Basocelular pode ser verificada também através do DOR dessa classificação, a qual alcançou um valor de 2.842,67.

Por outro lado, o modelo apresentou certa dificuldade em identificar o Melanoma e diferenciá-lo das outras lesões, como aponta a matriz da Figura 4.3 (c). Isso se deu pois o Melanoma é um tipo de lesão que é consequência direta da evolução de lesões do tipo Nevo Benigno [12]. Dessa maneira, muitas das características do Melanoma se assemelham às características do Nevo Benigno, o que dificulta a diferenciação e, por consequência, a classificação. Assim como para o Nevo Benigno, a diferenciação entre o Melanoma e a Queratose Seborreica também é desafiadora. Esses dois tipos de lesões apresentam características clínicas que se assemelham bastante, de forma que a diferenciação visual também se torna complexa [35].

Figura 4.3 – Matrizes de confusão, obtidas a partir da base de testes, da classificação pelo MCTC de um tipo de lesão em relação a todas as outras. Ao todo, foram usadas 220 imagens, sendo 55 por classe.

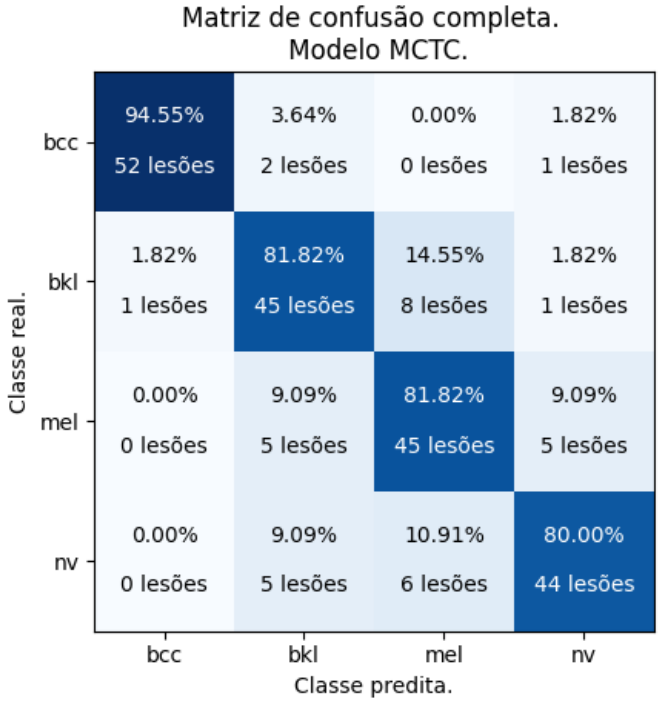


Fonte: O Autor, 2023.

A Figura 4.4 apresenta a matriz de confusão completa do MCTC, levando em conta todas as classes. A matriz presente nesta Figura consolida os resultados já observados na Figura 4.3. Além de apresentar a taxa de acertos para o Carcinoma Basocelular de 94,55%, a matriz de confusão da Figura 4.4 também aponta que as classificações incorretas de lesões do tipo Carcinoma foram como lesões do tipo Queratose Seborreica e Nevo Benigno, assumindo, respectivamente, os seguintes percentuais: 3,64% e 1,82%.

Para o caso da diferenciação entre a Queratose Seborreica e o Melanoma, a matriz de confusão aponta que 14,55% das imagens com lesões da classe Queratose Seborreica foram classificadas como sendo da classe Melanoma. Além disso, 9,09% das imagens com

Figura 4.4 – Matriz de confusão completa para o MCTC obtida através da base de teste, com 220 imagens.



Fonte: O Autor, 2023.

lesões do tipo Melanoma foram classificadas como sendo da classe Queratose Seborreica.

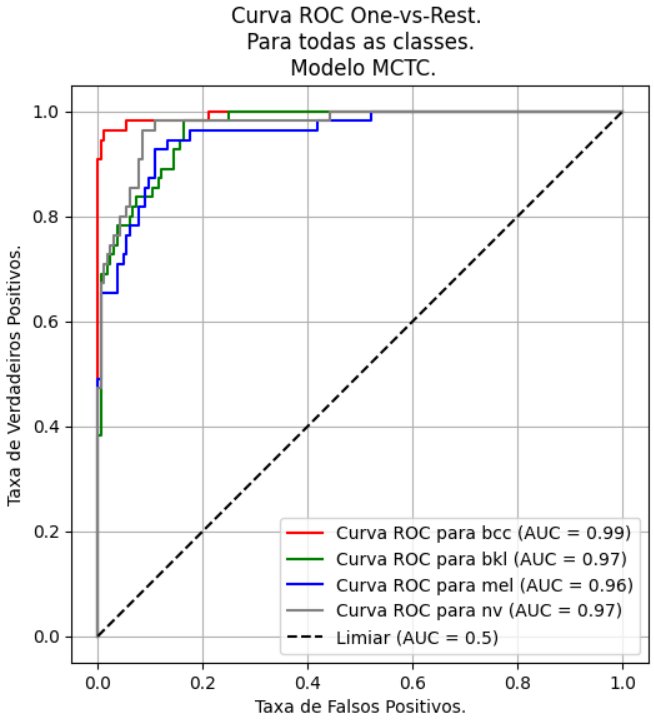
Já para diferenciação entre as classes Nevo Benigno e Melanoma, a mesma matriz aponta que 10,91% das imagens com lesões da classe Nevo Benigno foram classificadas como sendo da classe Melanoma. Além disso, 9,09% das imagens com lesões da classe Melanoma foram classificadas como lesões da classe Nevo Benigno.

Verifica-se também que o modelo MCTC apresentou uma leve dificuldade em diferenciar o Nevo Benigno da Queratose Seborreica, já que 9,09% das lesões do tipo Nevo Benigno foram classificadas como sendo Queratose Seborreica. Por outro lado, o contrário não ocorreu, isto é, o MCTC conseguiu diferenciar bem as lesões da classe Queratose Seborreica das lesões da classe Nevo Benigno, uma vez que apenas 1,82% das lesões do tipo Queratose Seborreica foram classificadas como Nevo Benigno.

Por fim, a Figura 4.5 apresenta as curvas ROC do MCTC, além dos respectivos valores de AUC, na classificação de cada uma das classes. O gráfico aponta que o menor valor de AUC, 0,96, ocorreu para a classificação do Melanoma, fato que se relaciona aos resultados apresentados na Matriz de confusão da Figura 4.3 (c). Ainda assim, o valor obtido para

AUC foi bastante alto. Cabe destacar também o excelente valor alcançado para a AUC da classificação do Carcinoma Basocelular, 0,99.

Figura 4.5 – Curva ROC para cada uma das classes, com seus respectivos valores de AUC, do teste do MCTC.



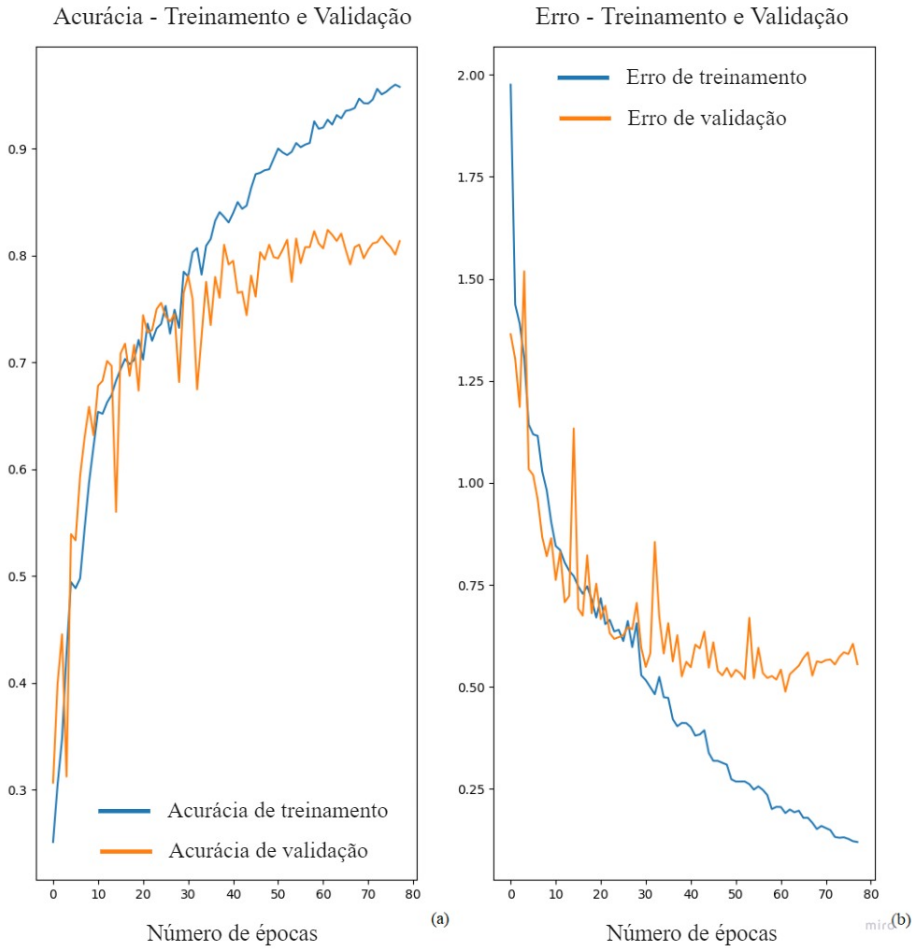
Fonte: O Autor, 2023.

4.3.2 Modelo VGG16

As curvas de aprendizagem do modelo VGG16 estão nos gráficos da Figura 4.6. À esquerda, na Figura 4.6 (a) encontram-se as curvas de acurácia da classificação em relação às bases de treinamento e validação. Estas curvas apontam que o modelo conseguiu alcançar uma acurácia de classificação em relação à base de validação ligeiramente superior a 80%, em torno da quinquagésima época de treinamento.

Já na Figura 4.6 (b) estão dispostas as curvas de erro de classificação também em relação às bases de treinamento e de validação. As curvas nesta Figura apontam a ocorrência de *overfitting* do modelo em torno da quinquagésima época de treinamento. Contudo, apesar da ocorrência deste fato, o método *ModelCheckPoint* permite salvar o modelo obtido que apresentou melhor desempenho em relação ao erro de validação, isto é, previamente

Figura 4.6 – Curvas de aprendizagem do modelo VGG16: Acurácia (a) e Erro (b) em relação às bases de treinamento e validação.



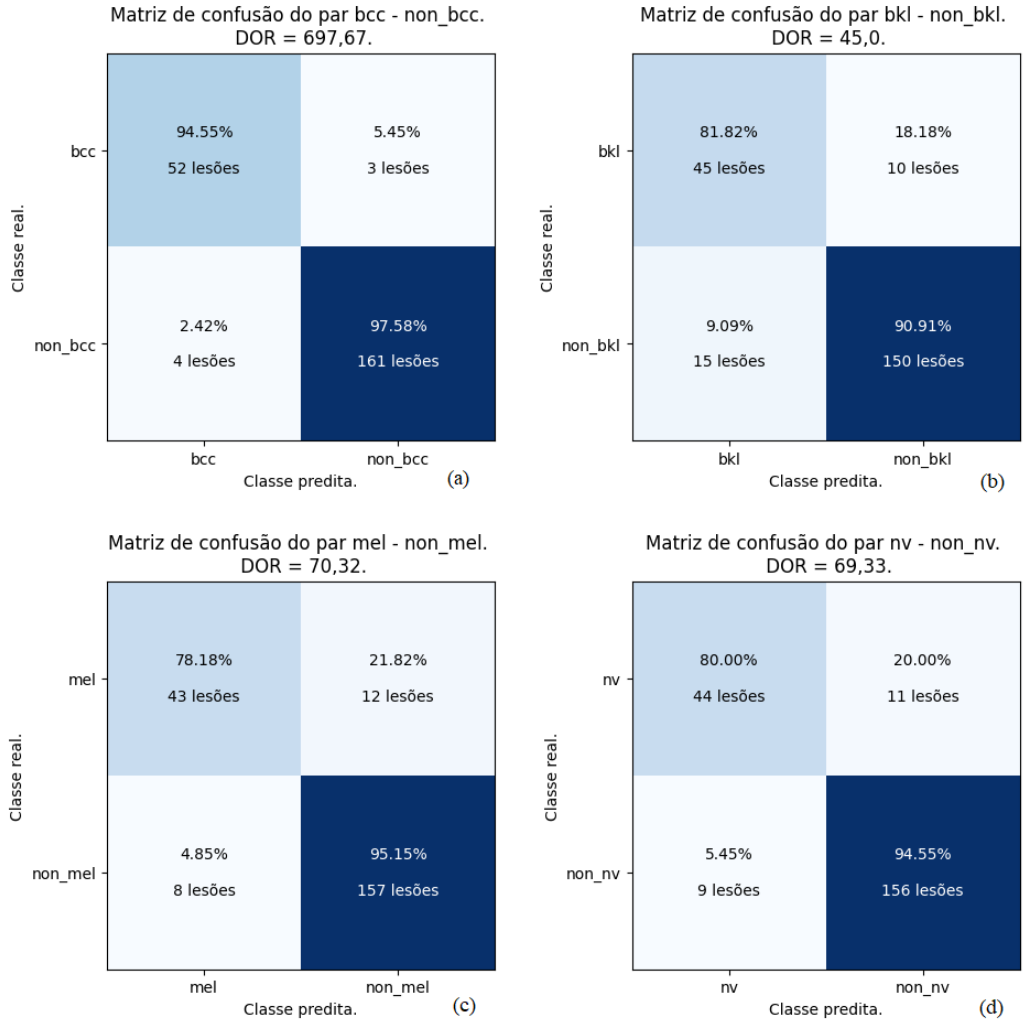
Fonte: O Autor, 2023.

ao aparecimento do *overfitting*.

A Figura 4.7 apresenta uma visão do desempenho do modelo VGG16, através de matrizes de confusão, na sua capacidade de identificar e diferenciar uma determinada classe de lesão em relação a todas as outras classes. As matrizes de confusão (a), (b), (c) e (d) da Figura 4.7 são, respectivamente, uma avaliação da capacidade do modelo de diferenciação das respectivas classes de lesões Carcinoma Basocelular, Queratose Seborreica, Melanoma e Nevo Benigno.

Como apresentado na Figura 4.7 (a), o VGG16 consegue identificar e classificar com performance satisfatória lesões do tipo Carcinoma Basocelular. A taxa de acerto de 94,55% é uma taxa bastante pertinente. Por outro lado, o VGG16 apresentou dificuldades na diferenciação do Melanoma em relação às outras classes de lesões. O percentual

Figura 4.7 – Matrizes de confusão, obtidas a partir da base de testes, da classificação pelo modelo VGG16 de um tipo de lesão em relação a todas as outras. o todo, foram usadas 220 imagens, sendo 55 por classe



Fonte: O Autor, 2023.

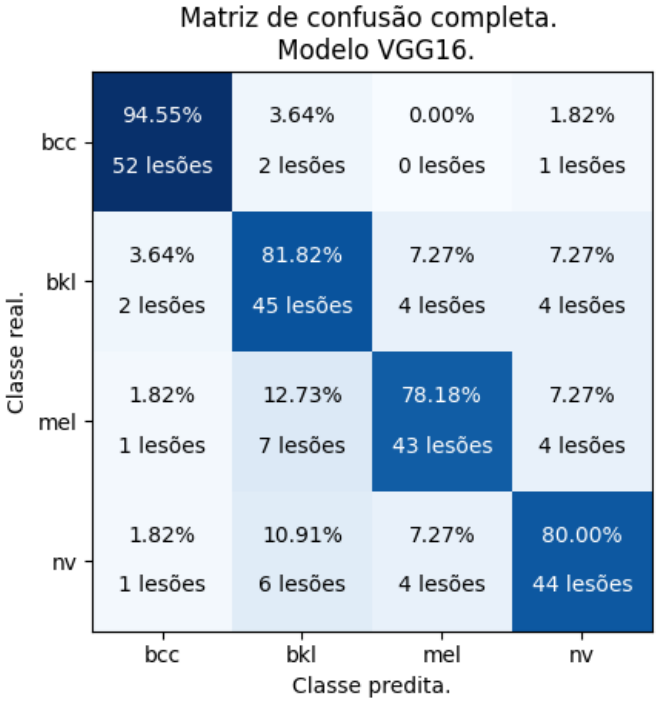
de acertos de 78,18% aponta essa dificuldade. Essa dificuldade acontece principalmente devido às características comuns que as lesões dos tipos Melanoma, Queratose Seborreica e Nevo Benigno possuem.

A Figura 4.8 apresenta a matriz de confusão completa para o modelo VGG16. Como mostra a Figura, o modelo VGG16 apresentou maior taxa de acertos na classificação do Carcinoma Basocelular, com uma taxa de 94,55%. Por outro lado, devido à semelhança das características entre as lesões Melanoma, Queratose Seborreica e Nevo Benigno, principalmente entre Melanoma e Queratose Seborreica, a taxa de acertos para o Melanoma

foi de apenas 78,18%.

Para o par de lesões Melanoma e Queratose Seborreica, a matriz aponta que 12,73% das imagens de lesões da classe Melanoma foram classificadas como Queratose Seborreica. Já para relação inversa, apenas 7,27% das imagens de lesões do tipo Queratose Seborreica foram classificadas como lesões da classe Melanoma.

Figura 4.8 – Matriz de confusão completa para o VGG16 obtida através da base de teste, com 220 imagens no total.

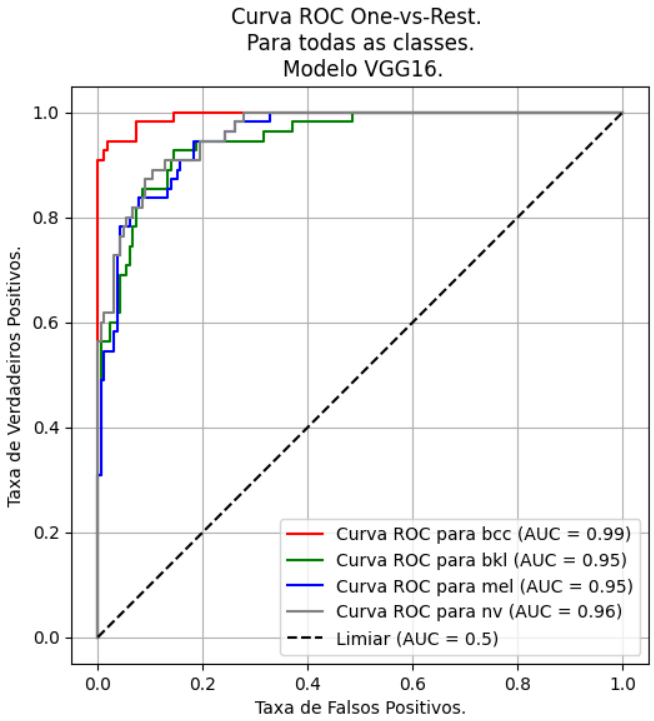


Fonte: O Autor, 2023.

A matriz de confusão da Figura 4.8 também aponta que 10,91% das imagens com lesões da classe Nevo Benigno foram classificadas como Queratose Seborreica. Para a comparação contrária, 7,27% das imagens de lesões do tipo Queratose Seborreica foram classificadas como Nevo Benigno.

Por fim, o gráfico da Figura 4.9 apresenta as curvas ROC, obtidas a partir da base de testes, para cada uma das classes de lesões classificadas pelo VGG16. Como mostra a Figura 4.9, o modelo VGG16 alcançou uma AUC de 0,99 para classificação do Carcinoma Basocelular, o maior valor obtido entre todos os valores de AUC para essa simulação. Por outro lado, ainda devido à semelhança de suas características, foi alcançada uma AUC de 0,95 para as curvas ROC das lesões Melanoma e Queratose Seborreica.

Figura 4.9 – Curva ROC para cada uma das classes, com seus respectivos valores de AUC, do teste do VGG16.



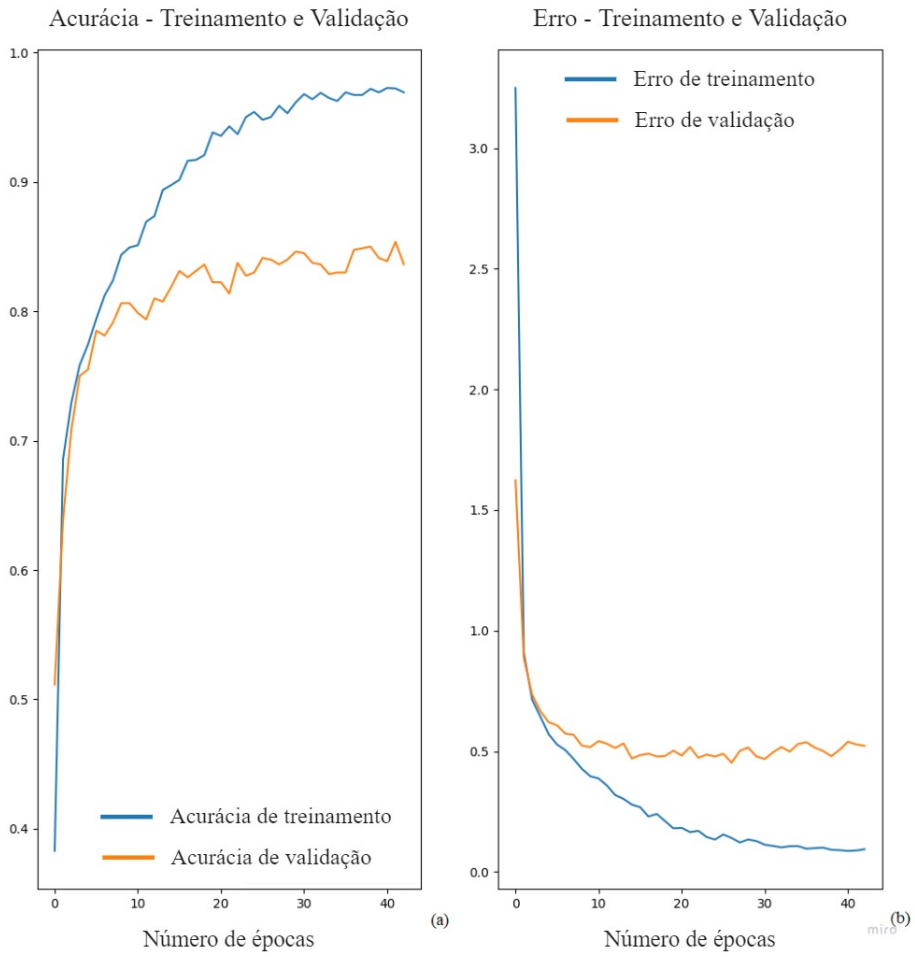
Fonte: O Autor, 2023.

4.3.3 Modelo Xception

As curvas de aprendizagem do modelo Xception estão nos gráficos da Figura 4.10. Na Figura 4.10 (a) encontram-se as curvas de acurácia da classificação em relação às bases de treinamento e validação. O gráfico indica que o modelo conseguiu alcançar uma acurácia de classificação em relação à base de validação superior a 80% logo nas primeiras épocas de treinamento, devido principalmente às características da arquitetura do modelo Xception descritas na seção 3.4.

O gráfico na Figura 4.10 (b), por outro lado, apresenta as curvas de erro do modelo em relação às bases de treinamento e validação. A curva de erro para base de validação aponta que o modelo conseguiu alcançar a convergência logo nas primeiras épocas de treinamento, ficando em torno de 0,5. Percebe-se também, a partir da Figura 4.10 (b), o surgimento de um pequeno *overfitting* em torno da trigésima época de treinamento. Contudo, através do uso do método *ModelCheckPoint* o modelo de melhor desempenho em relação ao erro de validação é salvo previamente à ocorrência deste fenômeno.

Figura 4.10 – Curvas de aprendizagem do modelo Xception: Acurácia (a) e Erro (b) em relação às bases de treinamento e validação.

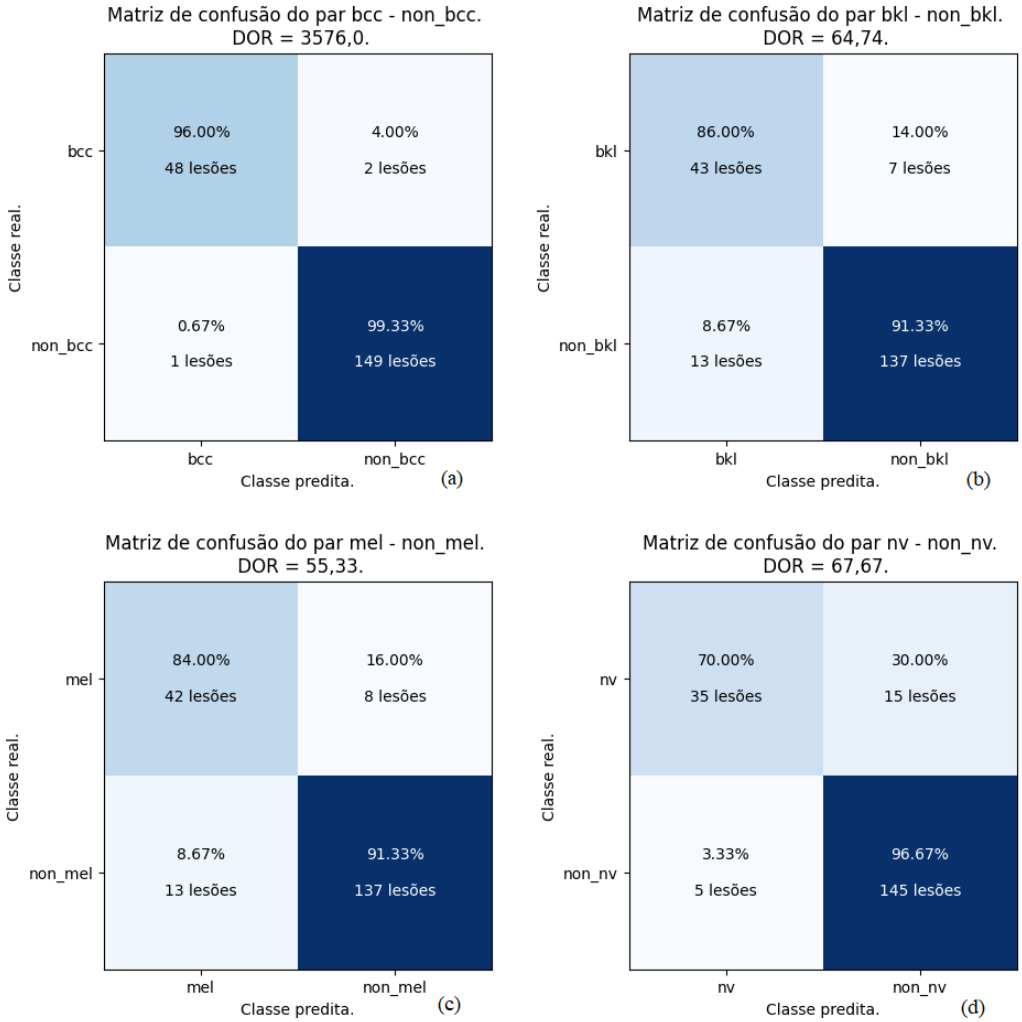


Fonte: O Autor, 2023.

A Figura 4.11 apresenta as matrizes de confusão que informam sobre a capacidade do modelo em identificar e classificar corretamente uma classe de lesão em relação a todas as outras. Na Figura 4.11 (a), verifica-se que o modelo Xception conseguiu alcançar uma taxa de acerto de 96,00% na classificação do Carcinoma Basocelular, com um DOR no valor de 3.576,00.

Por outro lado, como apontam as matrizes (b), (c) e (d) presentes na Figura 4.11, o modelo Xception encontrou maior dificuldade na identificação e diferenciação das lesões das classes Melanoma, Queratose Seborreica e Nevo Benigno em relação às lesões da classe Carcinoma Basocelular. Especificamente para as lesões da classe Nevo Benigno, o modelo Xception teve a menor taxa de acerto em relação às outras doenças, tendo alcançado um valor de 70,00%.

Figura 4.11 – Matrizes de confusão, obtidas a partir da base de testes, da classificação pelo modelo Xception de um tipo de lesão em relação a todas as outras. Ao todo, foram usadas 200 imagens, sendo 50 por classe.

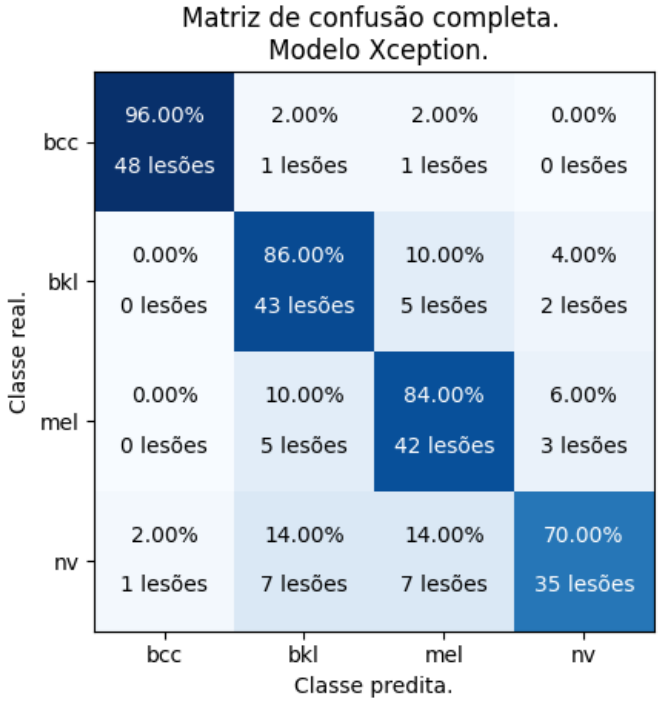


Fonte: O Autor, 2023.

Na Figura 4.12, a matriz de confusão completa, tem-se que os resultados da classificação do Nevo Benigno vistos na Figura 4.11 (d) ocorreram principalmente devido às semelhanças das características deste tipo de lesão com as características das lesões das classes Queratose Seborreica e Melanoma. Especificamente para lesões da classe Nevo Benigno, nota-se que 14,00% das imagens deste tipo de lesão foram classificadas como Melanoma, e que também 14,00% das imagens deste mesmo tipo de lesão foram classificadas como Queratose Seborreica.

Nesta mesma Figura, pode-se notar também que 10,00% das lesões do tipo Melanoma

Figura 4.12 – Matriz de confusão completa para o Xception obtida através da base de teste, com 200 imagens.



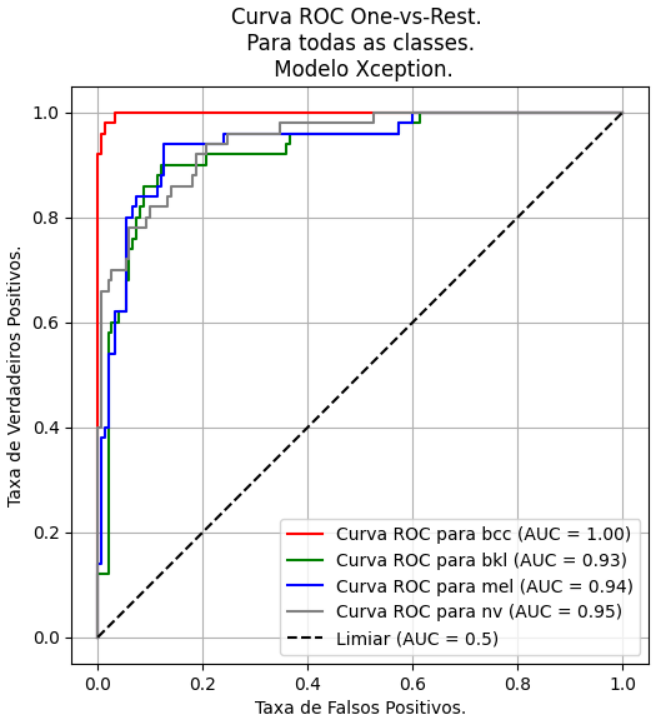
Fonte: O Autor, 2023.

foram classificadas como Queratose Seborreica, ao mesmo tempo em que 10,00% das lesões do tipo Queratose Seborreica foram classificadas como Melanoma. Mais uma vez, essa situação ocorre fruto das características semelhantes entre as lesões da classe Melanoma e da classe Queratose Seborreica.

Por fim, a Figura 4.13 apresenta as curvas ROC e os respectivos valores de AUC da classificação de cada uma das classes para o teste do modelo Xception. Cabe destacar o valor 1,00, valor máximo possível, assumido pela AUC na classificação do Carcinoma Basocelular, resultado este que pode ser correlacionado com os resultados dispostos nas matrizes de confusão (a) da Figura 4.11 e da Figura 4.12. Por outro lado, a classificação da Queratose Seborreica assumiu o menor valor de AUC, sendo 0,93.

É importante comentar o fato de, apesar de a taxa de acertos na classificação da Queratose Seborreica ser maior que a taxa de acertos na classificação do Nevo Benigno, a curva ROC da classificação da Queratose Seborreica apresentou uma AUC com o menor valor. Esse fato se deu pois, a AUC é uma medida da relação entre a Taxa de Verdadeiros Positivos e a Taxa de Falsos Positivos para diferentes limiares de classificação. Assim, ao

Figura 4.13 – Curva ROC para cada uma das classes, com seus respectivos valores de AUC, do teste do Xception.



Fonte: O Autor, 2023.

analisar as matrizes (b) e (d) da Figura 4.11, verifica-se de fato que a razão entre a Taxa de Verdadeiros Positivos e a Taxa de Falso Positivos é superior na classificação do Nevo Benigno quando comparado com a classificação da Queratose Seborreica.

4.4 Comparação entre as simulações de melhor performance

A Tabela 4.3 apresenta uma visão comparativa dos valores assumidos pelo DOR para cada uma das matrizes de confusão presentes nas Figuras 4.3, 4.7 e 4.11. A partir da Tabela, é possível verificar que o modelo Xception apresentou melhor desempenho que os outros modelos na identificação e classificação das lesões Carcinoma Basocelular e Queratose Seborreica. Por outro lado, na identificação e classificação do Melanoma, o VGG16 apresentou melhor performance, enquanto que na identificação e classificação do Nevo Benigno, o MCTC obteve o melhor desempenho.

Especificamente para o Carcinoma Basocelular, pode-se verificar que o valor do DOR para o modelo Xception é aproximadamente 5 vezes superior ao valor do DOR para o modelo VGG16. Já para o DOR do MCTC, também em relação ao mesmo tipo de lesão, percebe-se que seu valor é aproximadamente 4 vezes superior ao valor do DOR do modelo VGG16.

Tabela 4.3 – Valores da métrica DOR, obtidos a partir da base de testes, para classificação de cada um dos tipos de lesão em relação a todos os outros tipos nas simulações de melhor desempenho de todos os modelos.

Comparação dos valores do DOR para cada modelo			
Par de classes	DOR MCTC	DOR VGG16	DOR Xception
bcc-non_bcc	2.842,67	697,67	3.576,00
bkl-non_bkl	57,38	45,00	64,74
mel-non_mel	48,54	70,32	55,33
nv-non_nv	90,29	69,33	67,67

A Tabela 4.4 apresenta a Precisão dos modelos na classificação de cada uma das classes de câncer de pele. O modelo VGG16 apresentou Precisão superior aos outros modelos apenas na classificação do Melanoma. O modelo Xception apresentou Precisão superior na classificação do Nevo Benigno, enquanto que o MCTC apresentou melhor Precisão na classificação da Queratose Seborreica. Por fim, na classificação do Carcinoma Basocelular, ambos os modelos Xception e MCTC apresentaram a mesma Precisão, sendo 0,98.

Tabela 4.4 – Valores de Precisão, obtidos a partir da base de testes, da classificação de cada uma das classes pelos modelos.

Comparação da precisão da classificação dos modelos para cada classe			
Classe	MCTC	VGG16	Xception
Carcinoma Basocelular	0,98	0,93	0,98
Queratose Seborreica	0,79	0,75	0,77
Melanoma	0,76	0,84	0,76
Nevo Benigno	0,86	0,83	0,88
Média	0,85	0,84	0,85

Na Tabela 4.5 são apresentados os valores de Sensibilidade das classificações dos mo-

delos para cada uma das classes. O modelo Xception apresentou Sensibilidade superior na classificação de todas as classes, exceto na classificação do Nevo Benigno. Contudo, para o Nevo Benigno, os modelos VGG16 e MCTC apresentaram os maiores valores de Sensibilidade, ambos iguais, sendo 0,80.

Tabela 4.5 – Valores de Sensibilidade, obtidos a partir da base de testes, de classificação de cada uma das classes pelos modelos.

Comparação da sensibillidade da classificação dos modelos para cada classe			
Classe	MCTC	VGG16	Xception
Carcinoma Basocelular	0,95	0,95	0,96
Queratose Seborreica	0,82	0,82	0,86
Melanoma	0,82	0,78	0,84
Nevo Benigno	0,80	0,80	0,70
Média	0,85	0,84	0,84

Por fim, a Tabela 4.6 apresenta os valores de AUC das classificações de cada um dos tipos de lesões por todos os modelos. O modelo Xception apresentou o maior valor de AUC na classificação das lesões dos tipos Carcinoma Basocelular e Queratose Seborreica. Por outro lado, o MCTC teve AUC maior na classificação das lesões do tipo Melanoma e Nevo Benigno. Os valores máximos de AUC alcançados foram de 1,00 para o modelo Xception e de 0,99 para os modelos VGG16 e MCTC na classificação do Carcinoma Basocelular. Por outro lado, o menor valor de AUC obtido foi de 0,93, pelo modelo Xception, na classificação da Queratose Seborreica.

Tabela 4.6 – Valores de AUC, obtidos a partir da base de testes, de classificação de cada uma das classes pelos modelos.

Comparação da AUC da classificação dos modelos para cada classe			
Classe	MCTC	VGG16	Xception
Carcinoma Basocelular	0,99	0,99	1,00
Queratose Seborreica	0,97	0,95	0,93
Melanoma	0,96	0,95	0,94
Nevo Benigno	0,97	0,96	0,95
Média	0,973	0,963	0,955

Capítulo 5

Conclusão

A partir dos resultados das simulações descritos na Tabela 4.1, percebe-se que o modelo Xception apresentou o desempenho com a menor variabilidade de seus resultados. Para pequenos tamanhos de base, o modelo Xception conseguiu extrair de forma eficiente as características que identificam as classes de imagens dos tipos de câncer, obtendo acurácia de 75% e erro de 0,74 já na primeira simulação, com 400 imagens. Levando em conta todas as simulações, o modelo Xception apresentou uma acurácia média de 79,81% e um desvio padrão de 3,40% . Já para o erro, a média foi de 0,58 e o desvio padrão foi de 0,09.

Por outro lado, o modelo MCTC apresentou um comportamento oposto do ponto de vista da variabilidade de seus resultados, tendo esse comportamento mais acentuado quando comparado com os outros modelos. Também, o desempenho deste modelo foi bastante insatisfatório nas simulações iniciais, alcançando nas primeiras duas simulações, respectivamente, valores de acurária de 30% e 25% e valores de erro de 1,47 e 1,48. Apesar desse desempenho insatisfatório, o MCTC alcançou melhor desempenho entre todos os modelos na simulação final, levando em conta os valores de acurácia e erro, sendo, respectivamente, 85% e 0,47. A média das acurácias de todas as simulações foi de 67,27% e desvio padrão de 21,19%, o mais alto entre todos. Já para o erro, a média foi de 0,8 com desvio padrão de 0,36.

Por fim, o modelo VGG16 apresentou um desempenho bastante similar ao MCTC na última simulação, com acurácia de 84% e erro de 0,48. Além disso, a variabilidade de seus resultados ao longo das simulações não foi alta, ficando atrás apenas do modelo Xception nesse quesito. Considerando todas as simulações, a média de sua acurácia foi de 73,27%, com desvio padrão de 9,40%. Já para o erro, a média foi 0,63 e o desvio padrão de 0,19.

A Tabela 4.1 também aponta que os modelos MCTC e VGG16 alcançaram um relevante progresso em relação a acurácia e erro ao longo das simulações com bases de imagens cada vez maiores, obtendo resultados cada vez melhores. Por outro lado, apesar do excelente desempenho do modelo Xception já nas primeiras simulações, este apresentou uma certa estagnação em relação à melhora dos seus resultados ao longo das simulações, com a base de imagens cada vez maior.

Cabe destacar também, como aponta a tabela 4.2 e o gráfico na Figura 4.1, o modelo Xception apresentou a maior velocidade de convergência entre todos os modelos. Considerando um contexto em que a economia de recursos é relevante, o modelo Xception se destaca. A velocidade de convergência de resultados desse modelo é consequência direta de sua arquitetura. Como mencionado em [32], a operação de convolução separável em profundidade confere à arquitetura do Xception uma velocidade de convergência superior à arquitetura do InceptionV3, que é utilizado como Base de Extração de Características do MCTC. Graças a esse fato, a velocidade de treinamento do modelo Xception foi superior às velocidades dos outros modelos.

A partir da análise realizada na seção 4.3, pode-se dizer que o modelo proposto neste trabalho, o MCTC, apresentou o melhor desempenho entre todos os modelos na classificação das classes de câncer trabalhadas. Contudo, pode-se dizer que os resultados dos modelos Xception e VGG16 ficaram bastante próximos dos resultados do MCTC, tendo claro que o modelo VGG16 apresentou desempenho ligeiramente superior ao modelo Xception.

É importante destacar que todos os três modelos apresentaram bastante facilidade na classificação do Carcinoma Basocelular em comparação com a classificação dos outros três tipos de lesão. Este fato pode ser concluído a partir dos valores assumidos pela métrica DOR presentes na Tabela 4.3. Como aponta a primeira linha da Tabela, o DOR alcançou os valores de 2.842,67 para o MCTC, 697,67 para o VGG16 e 3.576,00 para o Xception na classificação do Carcinoma Basocelular. Os valores de precisão, sensibilidade e AUC presentes, respectivamente, nas tabelas 4.4, 4.5 e 4.6 para a classificação do Carcinoma servem para corroborar com a conclusão em questão.

Por outro lado, todos os três modelos apresentaram certa dificuldade na diferenciação entre as lesões do tipo Queratose Seborreica, Nevo Benigno e Melanoma. A dificuldade de diferenciação é consequência direta das características similares entre as três classes de

lesões [12], [35]. Além disso, os valores assumidos pelas métricas DOR, precisão, sensibilidade e AUC presentes, respectivamente, nas tabelas 4.3, 4.4, 4.5 e 4.6 para classificação desses três tipos de lesão atestam essa conclusão.

Dito isto, a partir dos resultados desse trabalho, pode-se dizer que os modelos de redes neurais convolucionais se apresentam como aliados de bastante potencial de especialistas para diagnóstico e classificação de tipos de câncer de pele. Diante do contexto em que o diagnóstico de câncer de pele ocorre, primariamente, de forma clínica, a partir da inspeção visual, as redes neurais convolucionais podem ser ferramentas capazes de ampliar de forma relevante a taxa de diagnóstico precisa de câncer de pele.

Trabalhos futuros

Como demonstrado no Capítulo 4, apenas o MCTC alcançou performance acima do objetivo mínimo de acurácia de 85%. Contudo, uma vez que estes modelos podem ser utilizados diretamente no desenvolvimento de ferramentas e técnicas que auxiliem especialistas no diagnóstico do câncer de pele, é importante e bastante relevante buscar aprimorar ainda mais os resultados obtidos.

Como abordado neste trabalho, um dos principais desafios encontrados se deu na diferenciação de tipos específicos de lesões de câncer devido a existência de um conjunto de características similares de lesões de classes distintas. Esse desafio tem como consequência direta a maior ocorrência de classificações como Falsos Positivos ou Falsos Negativos. Diante disso, uma potencial linha de trabalho futura consiste na aplicação de técnicas de processamento de imagens específicas para diferenciação das características únicas de cada tipo de lesão, a fim de aprimorar a acurácia de classificação dos modelos propostos.

Além disso, uma outra abordagem para evolução da performance do modelo pode ser da aplicação da técnica de *Fine Tuning*, como apresentada na subseção 2.3.6. Há também outras estratégias de refinamento que podem ser implementadas, tais como aumentar a base de dados para simulações, refinar os ajustes na taxa de aprendizagem ao longo do treinamento dos modelos, refinar as técnicas de *data augmentation* aplicadas, testar outros algoritmos de otimização, entre outros.

Para vislumbrar aplicações em situações reais, é relevante refinar o modelo para que este possa alcançar valores de acurácia com no mínimo 90%. Além disso, dada a diversidade de tipos de câncer de pele existentes, é oportuno e importante aprimorar a

arquitetura dos modelos, assim como as técnicas de pré-processamento dos dados de treinamento, para que seja possível identificar e classificar esses outros tipos de câncer.

Por fim, com o intuito de tornar a classificação de tipos de câncer de pele acessível, uma interessante linha de trabalho futura é a aplicação de modelos de classificação de câncer em sistemas embarcados. Nesse sentido, é importante considerar modelos que não exijam tanto do ponto de vista de memória e poder de processamento, uma vez que estes podem ser recursos limitados em hardwares embarcados.

Referências

[1] *Síntese de Resultados e Comentários — gov.br*. https://www.gov.br/inca/pt-br/assuntos/cancer/numeros/estimativa/sintese-de-resultados-e-comentarios. [Accessed 26-Jun-2023].

[2] *Câncer de pele — gov.br*. https://www.gov.br/saude/pt-br/assuntos/saude-de-a-a-z/c/cancer-de-pele. [Accessed 26-Jun-2023].

[3] *Pele Não Melanoma | A.C.Camargo Cancer Center — accamargo.org.br*. https://accamargo.org.br/sobre-o-cancer/tipos-de-cancer/pele-nao-melanoma. [Accessed 26-Jun-2023].

[4] The International Agency for Research on Cancer (IARC). *Global Cancer Observatory — gco.iarc.fr*. https://gco.iarc.fr/about-the-gco. [Accesso: 27/06/2023].

[5] *Cancer today — gco.iarc.fr*. [Accesso: 27/06/2023].

[6] Marceli de Oliveira Santos et al. "Estimativa de incidência de câncer no Brasil, 2023-2025". Em: *Revista Brasileira de Cancerologia* 69.1 (2023).

[7] SKIN MELANOMA IN BRAZIL. "Melanoma cutâneo no Brasil". Em: *Arquivos Catarinenses de Medicina* 38.Suplemento 01 (2009), p. 14.

[8] Khalid M Hosny, Mohamed A Kassem e Mohamed M Foaud. "Classification of skin lesions using transfer learning and augmentation with Alex-net". Em: *PloS one* 14.5 (2019), e0217293.

[9] Yanhui Guo et al. "Multiple convolutional neural network for skin dermoscopic image classification". Em: *2018 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*. IEEE. 2018, pp. 365–369.

[10] Spiros V Georgakopoulos et al. “Improving the performance of convolutional neural network for skin image classification using the response of image analysis filters”. Em: *Neural Computing and Applications* 31 (2019), pp. 1805–1822.

[11] Christiane Affonso De Donato Piazza e Sebastião de Almeida Prado Sampaio. “Queratose seborréica: estudo clínico, dermatoscópico e histopatológico”. Em: (2000).

[12] *Nevos - Distúrbios da pele - Manual MSD Versão Saúde para a Família — msdmanuals.com*. https://www.msdmanuals.com/pt-br/casa/distÃžrbios-da-pele/tumores-cutÃćneos-nÃčo-cancerosos/nevos. [Accesso: 20/07/2023].

[13] Caroline Salah Salmen e Marcos Wachowicz. “A atribuição da pessoa jurıédica à inteligência artificial: desafios e sua efetividade The attribution of the legal entity to artificial intelligence: challenges and its effectiveness”. Em: *Brazilian Journal of Development* 7.7 (2021), pp. 71438–71457.

[14] Pei Wang. “On defining artificial intelligence”. Em: *Journal of Artificial General Intelligence* 10.2 (2019), pp. 1–37.

[15] Dalvinder Singh Grewal. “A critical conceptual analysis of definitions of artificial intelligence as applicable to computer engineering”. Em: *IOSR Journal of Computer Engineering* 16.2 (2014), pp. 9–13.

[16] Simon Haykin. *Redes neurais: princıépios e prática*. Bookman Editora, 2001.

[17] Equipe DSA. *Capítulo 4 - O Neurônio, Biológico e Matemático - Deep Learning Book — deeplearningbook.com.br*. https://www.deeplearningbook.com.br/o-neuronio-biologico-e-matematico/. [Access: 24/01/2023].

[18] Buduma Nikhil e L Nicholas. *Fundamentals of deep learning*. 2017.

[19] Equipe DSA. *Capítulo 10 - As 10 Principais Arquiteturas de Redes Neurais - Deep Learning Book — deeplearningbook.com.br*. https://www.deeplearningbook.com.br/as-10-principais-arquiteturas-de-redes-neurais/. [Accessed 14-Jun-2023].

[20] Anirudha Ghosh et al. “Fundamental concepts of convolutional neural network”. Em: *Recent trends and advances in artificial intelligence and Internet of Things* (2020), pp. 519–567.

[21] Flávio HD Araújo et al. "Redes neurais convolucionais com tensorflow: Teoria e prática". Em: *SOCIEDADE BRASILEIRA DE COMPUTAÇÃO. III Escola Regional de Informática do Piauí. Livro Anais-Artigos e Minicursos* 1 (2017), pp. 382–406.

[22] *Maximum Pooling — kaggle.com*. https://www.kaggle.com/code/ryanholbrook/maximum-pooling. [Accesso: 21/07/2023].

[23] Sebastian Raschka. *Python machine learning*. Packt publishing ltd, 2015.

[24] Martin A Fischler e Oscar Firschein. *Intelligence: The eye, the brain, and the computer*. Addison-Wesley Longman Publishing Co., Inc., 1987.

[25] Zeferino Gomes da Silva Neto. "Curva ROC para comparação de modelos de predição para variáveis dicotômicas". Em: (2020).

[26] Afina S Glas et al. "The diagnostic odds ratio: a single indicator of test performance". Em: *Journal of clinical epidemiology* 56.11 (2003), pp. 1129–1135.

[27] ISIC. *Isic Archive*. URL: https://www.isic-archive.com/#!/topWithHeader/wideContentTop/main. acesso: 31.05.2023.

[28] TensorFlow. URL: https://www.tensorflow.org/api_docs/python/tf/keras/applications/xception/Xception. acesso: 05.06.2023.

[29] TensorFlow. URL: https://www.tensorflow.org/api_docs/python/tf/keras/applications/inception_v3/InceptionV3. acesso: 05.06.2023.

[30] TensorFlow. URL: https://www.tensorflow.org/api_docs/python/tf/keras/applications/vgg16/VGG16. acesso: 05.06.2023.

[31] Karen Simonyan e Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". Em: *arXiv preprint arXiv:1409.1556* (2014).

[32] François Chollet. "Xception: Deep learning with depthwise separable convolutions". Em: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1251–1258.

[33] Christian Szegedy et al. "Rethinking the inception architecture for computer vision". Em: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2818–2826.

[34] Keras Team. *Keras documentation: About keras*. URL: https://keras.io/about/. acesso: 25.05.2023.

[35] Alessandra Yoradjian, Natalia Cymrot Cymbalista e Francisco Macedo PaschoaL. "Queratose seborreica simuladora de melanoma". Em: *Surgical & Cosmetic Dermatology* 3.2 (2011), pp. 169–171.

Apêndice

Os códigos implementados para obtenção dos dados, definição da arquitetura do modelo e seus hiperparâmetros, assim como para treinamento, validação e testes dos modelos foram armazenados em repositório no GitHub. O repositório se encontra no link https://github.com/matheusvieirabs/TCC_classificacao_de_tipos_de_cancer_CNN.